Development and applications of machine learning frameworks for dynamic emulation of aerospace multiphysics problems and characterization of microstructure

by

Roberto Perera

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 4, 2024

Keywords: Machine Learning; Microstructural feature characterization; Graph Neural Networks; Deep Emulator Network SEarch (DENSE); Computational fracture mechanics; Microcracks coalescence; Adaptive Mesh Refinement; Mesh-Based Graph Neural Network; Transfer learning; Convolutional Encoder-Decoder; Convolutional Neural Networks; Deep Neural Networks; Brittle materials; Extended finite element method; Phase Field Model; Crack Propagation; Displacement Fields; Stress evolution

Approved by

Vinamra Agrawal, Chair, Assistant Professor of Aerospace Engineering
Davide Guzzetti, Assistant Professor of Aerospace Engineering
Russel Mailen, Assistant Professor of Aerospace Engineering
Suhasini Gururaja, Associate Professor of Aerospace Engineering
Nicholas Cummock, Research Scientist of U.S Naval Air Warfare Center Weapons Division

Abstract

Future challenges in aerospace problems, spanning space exploration and military aircraft, demand advancements in several areas, including in-space 3D printing, high-performance missile technology, and rapid structural failure modeling for aircraft and rockets. However, materials used in these applications, such as additively manufactured (AM) materials and Energetic Materials (EM), exhibit defects at atomistic and microstructural scales, impacting their structural integrity and failure behavior. Addressing these challenges requires improved computational models for material characterization and dynamic failure simulation. Machine learning (ML) methods offer a promising approach to develop such models and enhance data processing efficiency. In this study, we propose various ML frameworks and techniques to aid in the development of efficient computational models for characterizing and simulating failure response in heterogeneous 3D printed materials and EMs.

The first framework proposed is an autonomous ML model for fast characterization of pores, particles, grains and grain boundaries (GBs) from microstructural images of additively manufactured (AM) materials. To automate the process, the first ML model involves a classifier Convolutional Neural Network (CNN) to detect microstructures of pores or powder particles, versus GBs. For microstructures of pores or particles, a Convolutional Encoder-Decoder Network (CEDN) is used for generating binary segmentation images. Using an object detection ML network (YOLOv5), the particles' or pores' number, size and location are predicted with high accuracy. For GBs, Red-Green-Blue (RGB) segmentations are generated using an additional optimized CEDN. The Deep Emulator Network SEarch (DENSE) method (which employs the Covariance Matrix Adaptation - Evolution Strategy (CMA-ES)) is implemented to optimize the RGB CEDN in terms of computational speed. The characterization framework showed a significant improvement in analysis time when compared to conventional methods. The extracted defects can be used to rapidly estimate material properties in new unique heterogeneous AM

material configurations. Lastly, the predicted defects and estimated material properties can be used as input to computational models to simulate their failure dynamics.

Towards this effort, moving to simulation models of material failure, we then develop a Graph Neural Network (GNN) framework for simulating the failure response of brittle materials with multiple initial microcracks (5 to 19 microcracks) subjected to tension. First, a conventional eXtended Finite Element Model (XFEM-based) fracture model was used to generate training, validation, and test datasets. The number of cracks (5 to 19), their initial positions and orientations (0^o, 60^o, and 120^o) were varied. The graph representation involved vertices placed at each crack-tip and edges connecting each crack-tip to its neighboring crack-tips within a 750mm radius. To achieve high prediction accuracy, the framework architecture is established using a sequence of physics-informed GNN-based predictions. The first prediction stage determines Mode-I and Mode-II stress intensity factors (stress distribution), the second stage predicts which microcracks will propagate (quasi-statics), and the final stage propagates crack-tip positions to the next time instant. The trained GNN framework is capable of simulating crack propagation, coalescence and corresponding stress distribution with speed-ups 6x–25x faster compared to an XFEM-based simulator.

Next, while Microcrack-GNN was able to emulate crack propagation in problems involving multiple cracks with length of $300mm$, orientations of 0^o, 60^o, and 120^o, in a $2000mm \times 3000mm$ domain under tensile load, the framework did not consider other problem-specific inputs. For instance, problems involving shear loadings, arbitrary crack orientations, arbitrary crack lengths, and different domain sizes were not predicted. An important challenge in supervised ML applications is the need for large training datasets. Extending Microcrack-GNN to handle these varying problem-specific inputs using traditional approaches would require generating large datasets for each parameter change. Therefore, to circumvent the issue of needing large training datasets for new initial conditions and loading cases, we use Transfer Learning (TL) approaches from ML theory to extend Microcrack-GNN's capability. The new framework, ACCelerated Universal fRAcTure Emulator (ACCURATE), is generalized to a variety of crack problems using a sequence of TL update steps. The TL update steps are defined by sequentially training on significantly smaller datasets for: (i) arbitrary crack lengths,

(ii) arbitrary crack orientations, (iii) square domains, (iv) horizontal domains, and (v) shear loadings. Using significantly small training datasets (20 simulations for each TL update step), ACCURATE achieves high prediction accuracy in Mode-I and Mode-II stress intensity factors, and crack paths for these problems. A key addition of ACCURATE is its ability to predict crack growth and stress evolution with high accuracy for unseen cases involving the combination of new boundary sizes with arbitrary crack lengths and crack orientations, for both tensile and shear loading. Additionally, we demonstrate a significant acceleration in simulation time of up to 2 orders of magnitude faster (200x) compared to the XFEM-based fracture model. The ACCURATE framework provides a universal computational fracture mechanics model that can be easily modified or extended in future work.

Following this GNN framework along with TL, where the graph representation is formulated using vertices at each crack-tip, we then considered a mesh-based fracture simulator for phase field (PF) fracture models. As such, we develop a mesh-based GNN framework for emulating PF simulations of crack propagation. A key addition of this work is the introduction of Adaptive Mesh Refinement (AMR) to the graph representation itself. The framework (ADAPT-GNN) exploits the benefits of both ML methods and AMR by describing the graph representation at each time-step as the refined mesh itself. ADAPT-GNN is able to add nodes and edges dynamically as the mesh is refined. We predict the evolution of displacement fields (u,ν) and scalar damage field (or crack field, ϕ) with good accuracy compared to a conventional PF fracture model. The stress field (σ) is as also computed using the predicted displacements and PF parameter. In terms of computational efficiency improvement, ADAPT-GNN is 15-36x faster compared to serial execution of the PF model.

While ADAPT-GNN showed significant speed-up and overall good prediction accuracy, the framework involved limitations. Mesh-based GNNs such as ADAPT-GNN require a large number of message-passing (MP) steps and suffer from over-smoothing for problems involving very fine mesh. To mitigate challenges with conventional mesh-based GNNs such as ADAPT-GNN, we develop a multiscale mesh-based GNN framework mimicking a conventional iterative multigrid solver, coupled with adaptive mesh refinement (AMR). We use the framework to

accelerate PF fracture problems involving coupled partial differential equations with a near-singular operator due to near-zero modulus inside the crack. We define the initial graph representation using all mesh resolution levels. We perform a series of downsampling steps using Transformer MP GNNs to reach the coarsest graph followed by upsampling steps to reach the original graph. We use skip connectors from the generated embedding during coarsening to prevent over-smoothing. We use Transfer Learning (TL) to significantly reduce the size of training datasets needed to simulate different crack configurations and loading conditions. The trained framework showed accelerated simulation times, while maintaining high accuracy for all cases compared to physics-based PF fracture model. This work provides a new approach to accelerate a variety of mesh-based engineering multiphysics problems. In future efforts, the microstructure characterization framework can be used in conjunction with the developed mesh-based GNNs to accelerate computational failure models for heterogeneous AM materials with defects such as pores, particles, grains and GBs.

Lastly, in an effort to aid in the development of new high-performance missiles we also integrate ML methods for Heterogeneous Energetic Materials (HEM). In the realm of HEMs, where structural defects like pores are prevalent, predicting initiation metrics such as pressure, temperature, and particle velocity becomes complex due to the diverse arrangements of these defects. Current prediction methods rely heavily on experimental data and computational simulations, which are limited by the need for exhaustive testing across various pore configurations. To overcome this limitation, we introduce a novel ML framework to forecast critical velocities in PBX-9501 samples featuring multiple pores of varying sizes, quantities, and spatial distributions. In this framework, we employ the Computational Hydrocode (CTH) to simulate the shock response of each sample upon impact by a flyer plate, followed by the utilization of an automated bisection algorithm to compute critical velocities. We then develop two ML models, CNNs and GNNs, for predicting critical impact velocities. We perform rigorous evaluation of these models to assess their performance in predicting critical velocities across scenarios involving diverse spatial distributions, pore quantities, and pore sizes. The ultimate objective of this work is to develop ML-guided models capable of directly predicting critical velocities for unseen pore structures without the need for CTH simulations. By doing so, this framework lays

the groundwork for accelerated comprehension of how different pore configurations influence shock sensitivity in HEMs.

Acknowledgments

Above all, I want to thank God for illuminating my path each day to reach this accomplishment, and for allowing me to meet remarkably intelligent and supportive individuals along the way. Without His presence and blessings, none of my achievements would hold any significance.

I would like to express my heartfelt gratitude to the individuals who have been instrumental in supporting me throughout my Ph.D. First, I am deeply grateful for my academic advisor, Dr. Vinamra Agrawal, whose guidance, immense knowledge, and encouragement have been invaluable to me. Dr. Agrawal, thank you so much for being a beacon of inspiration and for guiding me on this journey of learning and self-discovery. I am truly fortunate to have had such an exceptional advisor who consistently pushes the boundaries of science and research.

I would also like to thank the rest of my thesis committee: Dr. Davide Guzzetti, Dr. Russell Mailen, Dr. Suhasini Gururaja, Dr. Nick Cummock, and Dr. Daniel Tauritz for their insightful comments, hard questions, and future research ideas. Specifically, I express my sincere appreciation to Dr. Davide Guzzetti for his exceptional mentorship and support during my initial year as a graduate student at Auburn University.

I extend my heartfelt gratitude to my supervisors and mentors at Naval Air Warfare Center Weapons Division (NAWCWD), Dr. Ephraim Washburn, Dr. Nick Cummock, Blake McCracken, Ann Obligacion, Dr. Gatwech Thich, Caleb Baumgart, and Emilio Bettez for their unwavering support and belief in my abilities throughout my summer internships. In particular, I want to extend my gratitude to Caleb Baumgart and Emilio Bettez for generously offering me rides to work every day. Furthermore, I wish to express my heartfelt appreciation to Dr. Nick Cummock and Blake McCracken for their dedicated guidance and enthusiastic support, ensuring that my internships at NAWCWD were both fruitful and enriching experiences.

During my time at Auburn University, I was very fortunate to develop long-lasting friendships with fellow classmates Alexander Davis, Manuel Indaco, Andrea Brandonisio, and Ryota Nakano. I want to express a special thanks to Alexander Davis, who has been like a brother

to me, offering guidance and spiritual support throughout my early years in graduate school. Special recognition goes to my Italian friends Manuel Indaco and Andrea Brandonisio for their companionship during numerous insightful conversations, coffee breaks, and road trips.

To my parents Ramon Perera and Consuelo Aguiar, I am profoundly grateful for your unconditional love, encouragement, and sacrifices, shaping me into the person I am today. I am forever indebted to you for instilling in me the values of hard work, perseverance, and determination. To my brothers and sister Ernest Perera, Javier Perera, and Aylet Perera, your presence has been a constant source of strength and motivation in both the highs and lows of my journey. Your camaraderie, humor, and shared experiences have provided invaluable moments of respite during challenging times. Finally, I want to dedicate this dissertation and all the work completed during my Ph.D. journey to my beautiful, smart, and sexy wife, Jennifer Cordovi. This achievement is as much hers as it is mine, and I am forever grateful for her loving and faithful presence by my side.

With respect to funding, this work was supported by Department of Defense (DOD) in conjunction with the Naval Air Warfare Center Weapons Division through the SMART Scholarship Program (SMART ID: 2021-17978). This work was supported in part by the DoD High Performance Computing Modernization Program at NAWCWD. Lastly, this work was supported in part by the High Performance and Parallel Computing Easley Cluster at Auburn University.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Over the past few decades, a manufacturing technique which has found applications towards aerospace rocket components, automotive parts, biomedical equipment, and infrastructural materials is additive manufacturing. Building on its success, current research endeavors focus on advancing additive manufacturing technologies for in-space 3D printing applications and developing higher-performance explosives. A notable characteristic of this technique is its dependence on a variety of adjustable input parameters, which can theoretically be optimized to achieve desired material properties, contributing to the development of stronger space structures and more efficient explosives. However, the relation between the manufacturing input parameters, the build process, and the resulting material properties is not well understood. This relation is defined as the process-structure-property of AM materials; understanding process-structure-property relations of AM materials is an ongoing effort in the materials science and solid mechanics fields [1].

A key observation made is that AM materials result in heterogeneous microstructures of various types of defects which are randomly distributed. Some of these microstructural defects are powder particles of different size distribution and morphology, microstructural pores, grain boundaries (GBs), and microcracks which are directly related to the manufacturing process itself. Multiple studies have shown that the interaction of these defects at various length scales is one of the leading cause of structural failure. For instance, microstructural pores and GBs are shown to affect the Young's Modulus, strength, fatigue life, and hardness of materials. As

such, the first step in the process-structure-property study is to understand the relation between the manufacturing parameters and the resulting microstructure (process-structure). Secondly, a connection between the AM material microstructure and its resulting mechanical properties and failure response (structure-property) must be established. Being able to comprehensively model the process-structure-property relation would grant a framework where optimal material microstructures can be tailored towards desired material properties or failure responses using specific manufacturing parameters (i.e., moving backwards through property-structure-process). In the proposed work, we focus on the second portion of the process-structure-property relation (i.e., structure-property).

First, let us address some of the challenges involved in current approaches used for microstructure characterization. Popular approaches used to extract meaningful information of the microstructure (e.g., defects' size distribution, locations, etc.,) include imaging techniques. Some of these techniques entail the development of image segmentation algorithms. For example, the watershed city-block distance (WCBD) method and the point-sampled intercept length method (PSILM), are used to generate binary segmentations of pores and particles, and semantic RGB (red-green-blue) image segmentations of GBs, respectively. Although these techniques have been useful, they require time-consuming pre-processing image operations, extensive computational resources for post-processing, and cannot handle more than one type of defect or microstructure at a time. Therefore, it is crucial to develop new computational microstructure characterization tools which are fast, autonomous, can handle various types of microstructure, and do not require pre/post-processing.

Next, the structure-property study requires for the extracted microstructural features to be incorporated into computational fracture mechanics models in order to simulate the failure behavior of the resulting defects. Towards this goal, popular computational fracture mechanics models used to emulate material failure include the eXtended finite element method (XFEM), cohesive zone model (CZM), and phase field (PF) technique. These high-fidelity fracture mechanics models, however, require solving complex systems of equations where computational costs increase quickly with problem complexity. For instance, a 3D microcrack coalescence problem involving a $1\ m^3$ domain can take several CPU-days to simulate. Therefore, the last

goal of the recent surge in understanding the linkage between AM material microstructure and resulting material failure behavior is the development of computationally efficient fracture mechanics models for heterogeneous microstructures.

Lastly, another notable material characterized by microscale and macroscale defects involves Energetic Materials (EMs). EMs date back to 6^{th} century old China when the combustion process of black powder was introduced. In principle, EMs undergo a rapid and powerful release of chemical energy in the form of heat and gas upon the application of loads. This explosive property has made them the prime candidates for military applications. While much work has been done in the past to characterize EMs' macroscale behavior using experimental approaches, a knowledge gap was introduced when considering Heterogeneous Energetic Materials (HEMs) (i.e., EMs involving material defects). Typical defects observed in HEMs are particles, microcracks, voids and grain interfaces. These defects can be located at the binder, at the energetic crystals and at the interfaces between the binder and the energetic crystals. Shock properties such as detonation initiation, pressure distribution, temperature distribution, particle velocity and shock velocity vary on a sample-to-sample basis due to the unique arrangements of material defects. Over the past few decades, multiple works have investigated the effects of microstructure on the resulting shock behavior of HEMs. These works have found that defects cause significantly greater sensitivity to shock compared to homogeneous HEMs (or pure crystals). As a result, understanding how various configurations of macroscale and microscale material heterogeneities, and impact or detonation affect the resulting shock response and macroscale behavior of HEMs has become a crucial subject of research. While computational models for simulating the shock response of HEMs have shown to produce faster and cheaper results compared to experiments, using computational models to study various HEM configurations and account for each possible arrangement of defects would require an extensive number of computational simulations, making them unfeasible for this problem.

1.2 Integration of ML into mechanics and materials science

1.2.1 Microstructure characterization

ML techniques offer a way to dramatically improve data processing efficiency of these problems and shorten the turnaround time. For microstructure characterization tasks and extraction of defects in 3D printed materials, tools such as Convolutional Neural Networks (CNNs) have been used for defect type detection or classification in microstructural images. Convolutional Encoder-Decoder Networks (CEDN) have also been implemented in image segmentation tasks. While current works introduced ML models capable of outperforming human expert defect quantification and conventional image processing tools, they were restricted by extracting a single type of defect for a single material, high GPU memory, prolonged training time, and complex network architecture depth. As a result, to mitigate these challenges we developed a fast, autonomous and optimized ML framework for microstructure characterization in AM materials with multiple types of defects (pores, particles, and grain boundaries) without requiring image pre-processing tools and prior-user information. First, the framework includes a ML classifier CNN for recognition of defect type (e.g., pores, particles, or GBs) given the initial microstructure, thus, making the framework independent of defect type. We then developed a simplified CEDN capable of generating binary segmentations for pores and particles to reduce the network's complexity (depth), GPU usage, and required training time. A key addition to the framework is the use of the Deep Emulator Network SEarch (DENSE) algorithm introduced in 2020 for ML neural network optimization. We used DENSE to optimize the simplified CEDN which further reduced training time and GPU usage by orders of magnitude. The framework then uses an object detection model to extract the location, size, and quantity of pores and particles. Additionally, for characterization of grains, we used the optimized CEDN for generating RGB segmentations of the GBs. We then developed a regression CNN to predict histograms of the grain size distribution. In this thesis, we present the framework's prediction for multiple types of materials involving various types of defects, and accelerated time performance analysis compared to conventional image processing algorithms.

1.2.2 Simulating fracture mechanics

Furthermore, an important aspect of the microstructure characterization framework outlined in Section 1.2.1 is its capability to extract defects information swiftly, enabling rapid estimation of material properties in new heterogeneous AM material configurations. This information on predicted defects and estimated material properties can then be employed to inform the modeling of failure dynamics across a range of diverse material configurations. However, incorporating the unique defect arrangements into existing physics-based simulation models would require solving highly intricate systems of equations, leading to computationally demanding models. As a result, ML methods have also been recently explored towards accelerating fracture mechanics problems. For instance, a graph-theory-inspired artificial neural network was recently introduced for predicting coalescence of cracks in systems with 19 microcracks and their material failure time. However, the developed models resulted in high prediction errors for some cases, and did not predict material failure dynamically at each time-step. Similarly, 2D crack growth in graphene was recently predicted using R-CNNs. This work was then extended using a convolutional long-short term memory (ConvLSTM) model to simulate crack propagation dynamically from molecular dynamic simulations. Although these works demonstrated promising results in using ML to simulate fracture dynamics, their ML models relied on binary image-based inputs, and their output consisted of binary images depicting the crack paths at future time-steps, failing to capture additional physics-based quantities throughout the material domain such as displacements and stresses. Therefore, new ML techniques capable of predicting both defect and stress evolution throughout the material domain in higher-complexity systems involving large number of initial defects have not been presented.

Towards this effort, in recent advancements, ML has introduced dynamic Graph Neural Networks (GNNs) for simulating physics problems with notably faster performance than traditional models. This innovative approach integrates graph theory with ML by representing problem configurations using nodes and edges, enabling the transfer and learning of physics through artificial neural networks. Recently, this method was applied to accelerate particle dynamics simulations involving fluids and deformable materials interacting with rigid bodies.

Additionally, it has been utilized in finite element simulations of plate bending, airflow over airfoils and cylinders, and flag dynamics, demonstrating high accuracy compared to conventional models. As a result, in an effort to explore higher-complexity problems involving multiple initial material defects, this thesis introduces various ML GNN techniques such as crack-tip-based GNNs, mesh-based GNNs, Multiscale GNNs, and Transfer Learning (TL) for simulating fracture dynamics with faster performance compared to classic computational fracture models.

The first framework, Microcrack-GNN, simulates crack propagation and coalescence in brittle materials involving 5 to 19 microcracks based on the XFEM-based approach. The structure of the Microcrack-GNN framework includes four GNNs: (i-ii) K_I-*GNN* and K_{II}-*GNN*, to predict Mode-I and Mode-II stress intensity factors respectively, (iii) *Class-GNN* to predict propagating versus non-propagating cracks, and (iv) *CProp-GNN* to predict future crack-tip positions. Using this approach, Microcrack-GNN not only predicts crack coalescence and propagation with high accuracy, but also the stress evolution.

Next, while Microcrack-GNN showed promising results to crack propagation and coalescence with faster performance, its accuracy sufferred when including new problem-specific inputs (e.g., loading type, domain size, etc.,) unknown to the training data. These cases require generating new large training datasets to retrain new individual models for each possible problem-specific input which is not a feasible approach. As possible solution for large data challenges we made use of TL methods. TL methods allow the transfer of learned information through the pre-trained weights from a baseline ML model. TL has been shown to significantly reduce the size of the training dataset, thus, reducing training time while achieving high accuracies. As a result, we applied TL to Microcrack-GNN, in order to develop the second framework - a generalized GNN framework - ACCelerated Universal fRAcTure Emulator (ACCURATE). The ACCURATE framework is able to predict crack propagation and stress evolution for various fracture mechanics problems involving new problem-specific inputs. ACCURATE handles the problem parameters of arbitrary crack lengths, crack orientations, domain size, and shear loading effects. The framework not only required significantly smaller training datasets for each problem-specific input, but it is also capable of predicting new cases unseen to the training datasets. Through the use of TL in ACCURATE, we demonstrate the efficacy of TL approaches

in using significantly smaller training samples, thus, reducing training time. The ACCURATE framework shows accelerated performance in simulation times with approximately 200x speedup compared to the XFEM-based surrogate model.

The third framework developed is an ADAPTive mesh-based GNN (*ADAPT*-GNN) for emulating PF models of single-edge crack propagation. Although in the Microcrack-GNN and ACCURATE frameworks the graph representation was formulated using vertices at each crack-tip, the ADAPT-GNN framework was inspired by recent works where mesh-based GNNs were introduced using the mesh points and edges from FEM simulations to directly represent the graph architecture itself. In ADAPT-GNN, we also leveraged the computational efficiency of Adaptive Mesh Refinement (AMR) for mesh-based approaches by dynamically adding/removing graph nodes and edges to mimic the refined mesh at each time-step of the PF simulation. The framework predicts displacements, (u, ν), and a scalar damage field field, ϕ, for each point in the adaptive mesh at each time-step with up to 36x faster performance compared to a conventional PF model.

Lastly, while the ADAPT-GNN framework demonstrated impressive capabilities in simulating complex multiphysics problems with significantly reduced computational times, it faced challenges such as requiring a high number of message-passing (MP) steps and susceptibility to over-smoothing. To address these issues, the fourth framework introduced a multiscale mesh-based GNN framework inspired by iterative multigrid solvers and integrated with AMR. This framework aimed to alleviate the limitations of conventional mesh-based GNNs such as ADAPT-GNN. In this framework, the initial graph representation encompassed all mesh resolution levels, and a series of downsampling steps utilizing Transformer MP networks were employed to reach the coarsest graph resolution. Subsequently, upsampling steps were executed to revert to the original graph resolution. Skip connectors were incorporated from the generated embedding during coarsening to mitigate the risk of over-smoothing. Additionally, TL techniques were leveraged to drastically reduce the size of training datasets required for simulating various crack configurations and loading conditions. The trained framework exhibited accelerated simulation times while maintaining high accuracy across all cases compared to physics-based PF fracture

models. Ultimately, this work presents a novel approach to expedite a wide range of mesh-based engineering multiphysics problems, offering significant advancements in computational efficiency without compromising predictive accuracy.

1.2.3 Predicting sensitivity of heterogeneous energetic materials

In recent years, ML approaches have emerged as promising tools for HEMs. Various works have implemented ML for predicting chemical properties of HEMs based on their molecular configurations. CNNs have also been successfully applied to predict material properties of HEMs with crystal microstructures. However, the effects of macroscale defects on these materials have not been explored using ML techniques in recent works.

To address this gap, our study focuses on developing an ML framework for predicting critical velocities in PBX-9501 samples with multiple macroscale pores characterized by varying quantity, size, and spatial distribution. Initially, we employ CTH simulations to emulate the shock response of each sample upon impact by a flyer plate. Subsequently, we utilize an automated bisection algorithm to compute the resulting critical velocities. Two distinct ML models are employed in our framework: a CNN and a GNN. We thoroughly evaluate the performance of each model in predicting critical velocities and assess their robustness in scenarios involving new spatial distributions, pore quantities, and pore sizes. Our objective is to develop models capable of directly predicting critical velocities for unseen pore structures without the need for executing CTH simulations. Ultimately, our work aims to pave the way for ML-guided models that elucidate how different pore structures influence shock sensitivity in HEMs. By leveraging ML techniques, we seek to enhance our understanding of the complex interplay between macroscale configurations and material properties in HEMs, thereby facilitating the development of more efficient and tailored energetic materials for various applications.

1.2.4 Contributions

The development of aerospace structures and materials for space exploration and military applications requires new manufacturing techniques such as additive manufacturing. However, AM materials involve unique arrangements of defects at multiple length scales which affect the

resulting material properties and structural failure. While these defects can negatively affect structural integrity, specific heterogeneous arrangements can also be tailored to obtain desired material properties and behavior. Currently, high-fidelity computational models employed to simulate the failure dynamics in complex heterogeneous materials are computationally expensive. Therefore, the main contributions of thesis focus on the development of novel and accelerated ML models for material characterization and accelerated dynamic failure simulation of aerospace structures with multiple microscale and macroscale defects.

- In Chapter 2, we present an optimized ML framework designed to autonomously and efficiently characterize pores, particles, grains, and grain boundaries in AM materials. The framework comprises Classifier and Regression CNNs, Semantic Segmentation Encoder-Decoder Networks, and an object detection model. To enhance the performance of the Semantic Segmentation Encoder-Decoder Networks, we implemented the DENSE optimization algorithm for Neural Architecture Search (NAS). Overall, our framework achieved a remarkable speed-up in material characterization time compared to conventional methods. A published version of this work can be found in [2]: R. Perera, D. Guzzetti, and V. Agrawal, "Optimized and autonomous machine learning framework for characterizing pores, particles, grains and grain boundaries in microstructural images," *Computational Materials Science*, vol. 196, p. 110524, 2021.

- In Chapter 3, we develop GNN based framework tailored for simulating structural failure in brittle materials with multiple initial microcracks. The trained GNN framework effectively captures crack propagation, coalescence, and stress evolution across a broad spectrum of initial microcrack configurations, ranging from 5 to 19 microcracks, without requiring further modification. Compared to an XFEM-based fracture simulator, our framework demonstrates high prediction accuracy. Furthermore, the simulation time of our framework exhibits notable speed-ups, ranging from 6x to 25x faster than conventional XFEM models. A published version of this work can be found in [3]: R. Perera, D. Guzzetti, and V. Agrawal, "Graph neural networks for simulating crack coalescence and propagation in

brittle materials," *Computer Methods in Applied Mechanics and Engineering*, vol. 395, p. 115021, 2022.

- In Chapter 4, we employ TL approaches to extend and enhance the previously developed GNN framework tailored for simulating structural failure in brittle materials with multiple initial microcracks. By leveraging a series of TL update steps, we effectively avoid the need for retraining with large datasets (only 20 simulations were needed). Our results demonstrate the newfound capability of the framework to accurately simulate crack growth and stress evolution in unseen scenarios, encompassing various boundary dimensions, arbitrary crack lengths, and orientations, under both tensile and shear loading conditions. This generalized GNN framework, dubbed ACCURATE, yields significantly accelerated simulation times, achieving up to 200x faster performance compared to the XFEM-based fracture model. A published version of this work can be found in [4]: R. Perera and V. Agrawal, "A generalized machine learning framework for brittle crack problems using transfer learning and graph neural networks," *Mechanics of Materials*, vol. 181, p. 104639, 2023.

- In Chapter 5, we proceed to develop a mesh-based GNN framework tailored for emulating phase field simulations of crack propagation, augmented with Adaptive Mesh Refinement (AMR) capabilities to accommodate various crack configurations. Dubbed ADAPT-GNN, this framework capitalizes on the strengths of both machine learning methods and AMR by representing the graph at each time-step as the refined mesh itself. ADAPT-GNN exhibits remarkable accuracy in predicting the evolution of displacement fields, scalar damage field, and stress field, showcasing accuracy on par with conventional phase field fracture models. Moreover, this mesh-based GNN framework yields substantial simulation speed-ups, ranging from 15 to 36 times faster compared to the serial execution of the phase field model. A published version of this work can be found in [5]: R. Perera and V. Agrawal, "Dynamic and adaptive mesh-based graph neural network framework for simulating displacement and crack fields in phase field models," *Mechanics of Materials*, vol. 186, p. 104789, 2023.

- In Chapter 6, we focus on mitigating current challenges of large scale mesh-based GNNs which require a substantial number of message-passing steps and suffer from over-smoothing. To address this issue, we introduce a novel multiscale mesh-based GNN framework, inspired by conventional iterative multigrid solvers, coupled with AMR capabilities. This framework is tailored to accelerate simulation times for multiphysics problems, such as phase field fracture models entailing coupled partial differential equations, which involve a near-singular operator arising from near-zero modulus regions within the crack. Our approach pioneers a unique downsampling and upsampling strategy, dynamically adding/removing the finest mesh resolution level at each time step within the refined mesh. Moreover, leveraging TL techniques, we notably reduce the size of required training datasets, thereby broadening the framework's applicability to encompass diverse crack configurations and loading scenarios. Our multiscale GNN framework not only demonstrates accelerated simulation times but also upholds high prediction accuracy across all tested cases when compared with traditional physics-based phase field fracture models. A published version of this work can be found in [6]: R. Perera and V. Agrawal, "Multiscale graph neural networks with adaptive mesh refinement for accelerating mesh-based simulations," *arXiv preprint arXiv:2402.08863*, 2024. (*under review*)

- In Chapter 7, we consider the effects of macroscale defects such as pores on the resulting shock sensitivity of explosives, and focus on the development of new ML approaches for predicting sensitivity faster than current computational models. To this end, we introduce a novel ML framework tailored for predicting critical velocities in PBX-9501 samples characterized by multiple macroscale pores exhibiting varying quantities, sizes, and spatial distributions. Leveraging both CNNs and GNNs architectures, our framework attains remarkable accuracies exceeding 95%. These results show the importance of relying on ML models to rapidly predict material sensitivity in the future without the need for a hydrocode.

Chapter 2

Autonomous and Optimized Machine Learning Framework for Microstructure Characterization of Additively Manufactured Materials

2.1 Introduction and motivation

In recent years, additively manufactured metals (AM) have been utilized towards biomedical parts [7], automobile components [8], rocket equipment [9], and infrastructural elements [10]. However, the mechanical properties and failure behavior of these materials is affected by their resulting microstructure from the manufacturing process itself. For example, metallic materials manufactured using powder bed fusion (PBF) result in powder particles with varying morphology and size distribution. The resulting configurations of powder particles significantly affect the material hardness, energy absorption, strength, surface roughness, and thermal conductivity [11, 12]. Another important microstructural defect in AM metals which affects material properties is pores. Microstructural pores occur during the solidification process where inert gases become trapped within the material. These microstructural pores create additional stress concentrations that can cause crack initiation and nucleation [13]. Additionally, microstructural pores have been found to decrease mechanical properties such as strength, Young's Modulus, and fatigue life [14, 15, 16]. The build direction used in the manufacturing process also leads to the generation of microstructural columnar grains [17]. These microstructural defects defined as grain boundaries (GBs) have been found to affect the stress-strain behavior, hardness, strength, and fatigue life of AM materials [18]. As a result, a crucial step in determining the mechanical properties and failure behavior of AM materials involves the characterization of their microstructural features.

In the past, numerous imaging techniques have been established towards this effort. For instance, scanning electron microscopy, electron back scattering diffraction, helium pycnometry,

energy dispersive spectroscopy, X-Ray computed tomography, and laser and X-Ray diffraction [19]. However, imaging techniques require computational time for post-processing which quickly becomes expensive due to the large amount of data these methods generate. Another computational tool used to characterize GBs, particles and pores is image segmentation. Popular image segmentation algorithms used to identify pores include city-block distance function + watershed segmentation [20] and assisted threshold binarization procedure [21]. Another widely used image segmentation algorithm used for GB characterization includes the point-sampled intercept length method (PSILM) [22, 23]. The PSILM algorithm first employs a standard edge detection algorithm to position a fixed number of points over the edges detected. Various lines are then generated for each point-pair along the following four directions: $0°$, $90°$, $45°$, and $-45°$. Lastly, the PSILM algorithm develops RGB segmentations and histograms of the resulting grain size using the resulting intersecting lines.

While the WCBD algorithm has found applications in characterization of pores and particles, and the PSILM algorithm has been useful for grain boundaries extraction, these methods still require additional computationally expensive and time consuming pre-processing image operations. For instance, morphological operators, edge detection, gamma corrections, interpolation functions, and threshold value selection [20, 21, 22, 23]. Another drawback of these conventional image segmentation algorithms is that they can only be implemented to one type of AM material and microstructural feature (pores, particles, or GBs) independently. As a result, to improve computational time and costs of current microstructure characterization approaches it is critical to develop new autonomous frameworks with faster turnaround time.

A tool that offers a route to mitigate these challenges and develop such frameworks is Machine Learning (ML). ML tools have shown significant data processing efficiency with fast performance [24] while providing comparable accuracy to current state-of-the-art methods. Additionally, an autonomous ML framework would not require prior information from the operator for the input material or the specific microstructural feature (pores, particles, or GBs). ML techniques have been employed in recent years for material microstructure characterization. For example, for predicting vertical or horizontal microstructural dendrites [25], predicting coarse-grained dislocation of different dislocation density field variables [26], and predicting

the AM powder feedstocks of varying particle size distribution (PSD) [27]. These works, however, were only able to predict one detail of the microstructure. For example, the effects of grain clustering and grain spatial distribution which is linked to the resulting hardness of the material [28, 29] were not predicted. Further integration of ML techniques along with image processing tools have successfully shown to characterize additional microstructural features. In [30], the authors developed a ML and image processing framework for predicting the size, shape, and location of line and loop dislocations in irradiated steel. The framework involved a 15-layer Convolutional Neural Network (CNN) along with conventional image processing tools (watershed flood algorithm, MATLAB built-in regionprops tool, and cascade detector). In [31], the authors developed an autonomous workflow to detect helium bubbles in microstructures of Irradiated X-750. The authors were able to predict bounding boxes for each defect in Irradiated X-750 samples with orders of magnitude speedup using the open-source LabelImg program along with a ML model (Region Proposal Faster R-CNN). While these works showed the benefits of using ML for material characterization, they were still restricted in extracting a single type of microstructural feature (particles, pores, GBs, etc).

In the past, ML models such as CNNs and encoder-decoder networks have also been developed towards binary and semantic segmentation of material microstructure. For instance, in [32] the state-of-the-art encoder-decoder model, U-Net, was implemented to generate binary segmentations of aluminum alloys. In [33, 34], the authors developed Residual Neural Networks (RNNs) and Fully Convolutional Neural Networks (FCNNs) to first predict microstructural shapes (i.e., ”lamellar”, “duplex”, or “acicular”) and then generate binary segmentations of the resulting microstructure. Additionally, [35] used SEM images of steel to generate binary segmentations for the voids, precipitates and line dislocations using a Convolutional Encoder-Decoder Network (CEDN) called DefectSegNet. The DefectSegNet framework outperformed defect characterization of human expert with significantly faster performance. While these works show the advantages of ML image segmentation tools, current CEDNs involve highly complex and large architectures which lead to prolonged training times and high GPU requirements. For instance, in Sections 2.4.1 and 2.4.4, we show the state-of-the-art U-Net CEDN model to

require a training time of 1:55 hours with GPU usage of 998.28 MBs to generate accurate RGB segmentations of GBs.

In the recent years, the challenge of high GPU usage in ML studies has led to the development of new NVIDIA GPUs aimed to improve their storage capacity and performance speed. However, as network architecture becomes more complex the development of new faster and smaller models may be crucial, especially for problems involving large dataset. An existing GPU resource which provides free usage (12 GBs for 12 hours) is Google Colab. Therefore, using these free-to-use resource-bound machines will require the development of new computationally efficient ML models which produce results with similar accuracy as state-of-the-art ML models.

This challenge may be solved through the Deep Emulator Network SEarch (DENSE) framework [36]. DENSE was introduced in 2020 to optimize the architecture of existing ML models in terms of their GPU usage and training time, while maintaining good accuracy. As a result, we use DENSE in this work to optimize the GPU memory and training time requirements of current state-of-the-art semantic segmentation networks such as U-Net. Using DENSE, we are able to significantly reduce the architecture complexity of U-Net while achieving high-accuracy binary and RGB semantic segmentations for pores, particles, and GBs, respectively. To gather a large training dataset of particles and pores, we make use of the standard city-block distance function along with the watershed segmentation algorithm presented in [20]. For GBs, we gather a large training dataset using the standard PSILM algorithm in [37, 38]. We then develop a ML classifier CNN for predicting which type of defect is present in each microstructure (i.e., particles, pores, or GBs). Using the initial architecture of the stat-of-the-art U-Net model, we significantly simplify its complexity by removing various convolution and transpose convolution layers. This results in a simplified network with reduced training time and GPU usage. We also optimize the resulting simplified U-Net CEDN model by employing DENSE for RGB segmentations of GBs. For GBs, we develop an additional regression CNN for predicting a histogram of the resulting grain size. Lastly, the integration of the Classifier CNN, binary and RGB CEDNs, and Regression CNNs, provide an autonomous microstructure characterization framework for AM materials involving particles, pores, and GBs without prior user input of the individual microstructural features.

2.2 ML methods and optimization

2.2.1 Convolutional Neural Networks for classification and regression tasks

A widely used supervised neural network involves Classification CNNs. Classification CNNs have found applications in prediction of spam emails, handwritten digits, clothing, real-time recognition of objects, and gender [39, 40, 41, 42, 43]. These models are typically used for predicting binary outputs (single classification tasks), as well as predicting multiple classes (i.e., up to thousands) using the activation function "Softmax" [44]. Unlike Classification CNNs where the predictions are discrete numbers, another widely used supervised neural network involves Regression CNNs which focus on the prediction of continuous numbers. Example applications of Regression CNNs include stock market prediction, facial landmark detection, and human-pose landmark detection [45, 46, 47].

In this work, we develop a classifier CNN to automate the detection process for the initial microstructure type. In essence, the classifier CNN is used to predict whether the input microstructure involves pores, particles, or GBs. This model allows the user to input a mixed database of microstructural images involving varying defect types without time-consuming separation of the images prior to extracting their features. Lastly, we generate histograms depicting the grain size distribution in images involving GBs using two regression CNNs for predicting the grains radii and their frequencies (i.e., distribution).

2.2.2 Convolutional Encoder-Decoder Networks for semantic segmentation

In 2015, the CEDN SegNet model was developed for image semantic segmentation [48]. The SegNet model was initially aimed at real-time semantic segmentation of nature, animals, people, and objects (e.g., cars) for applications such as scene understanding and self-driving vehicles. CEDNs were then developed following the SegNet methodology for image segmentation of higher complexity tasks. For instance, the CEDN U-Net, also developed in 2015, followed the SegNet approach with the addition of multiple feedforward and fully connected convolutional layers at the center of the encoder and decoder [49]. However, the initial purpose of the U-Net

was specifically for the task semantic segmentation of biomedical images involving microscopic neural structures.

The U-Net model has been extended in recent years for multiple tasks by integrating additional state-of-the-art ML methods. A common example of an extension to the U-Net model is the Res-UNet. The Res-UNet made use of Residual Networks (ResNet) for replacing the fully connected convolution and feedforward layers between the encoder and decoder section [50]. The use of ResNet was shown to mitigate exploding or vanishing gradients with the use of skip connectors and identity layers [51]. The Res-UNet has been successfully applied towards biomedical image segmentation of brain tumors [52], retina vessels [53], photovoltaic panels [54], and photoacoustic panels [55]. As a result, in this work we optimize the architectures of conventional U-Net and Res-UNet models towards accurate binary and RGB image segmentations of microstructural images involving particles, pores, and GBs.

2.2.3 Deep Emulator Network SEarch (DENSE) for optimization of ML models

In Januray 2020, M.F. Kasim introduced a framework called Deep Emulator Network SEarch (DENSE) for rapid optimization of ML models' architecture [36]. The DENSE framework initially aimed at fast emulation of large-scale systems using ML. Current predictive models for emulation of large-scale systems such as climate change, high energy density physics, seismology, astrophysics, and biogeochemistry are extremely computationally expensive. Although ML models have demonstrated significant speed up compared to conventional approaches, determining the optimal architecture (e.g., channels, filter dimensions, stride and padding size, etc.) for each problem requires significant computational processing time. Therefore, the purpose of DENSE was to dynamically search for optimal architecture parameters in ML emulators. To generate the search space of network architectures and optimize them iteratively, the DENSE framework integrated CNNs along with the evolution algorithm of Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) [56]. The authors showed that the DENSE framework was capable of speeding up the architecture optimization dynamically, thus, producing faster emulation with maintained high accuracy. As a result, in this work we make use of the DENSE

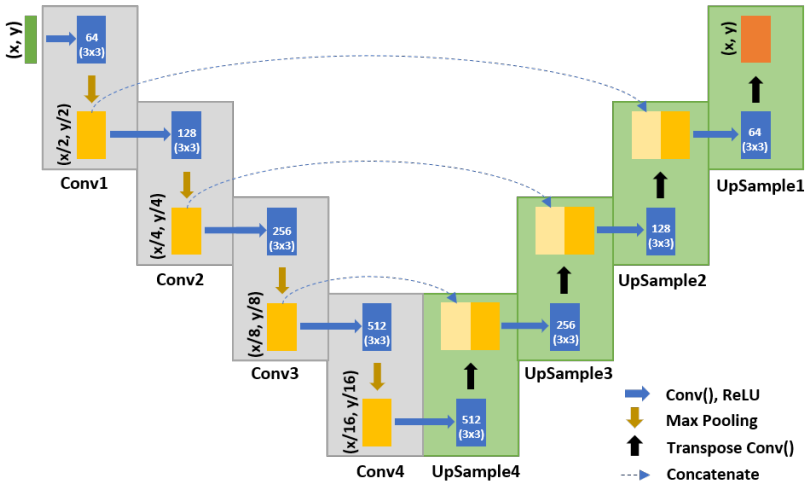


Figure 2.1: Simple-UNet: The architecture of the simplified U-Net encoder-decoder network.

framework for optimizing the CEDNs for binary and RGB image segmentation, as well as the Regression CNNs.

2.2.4 Development of Simple - UNet, and DENSE - UNet for image segmentation

In order to develop the CEDNs for binary segmentation of particles and pores, we used the conventional U-Net model and simplified its architecture to obtain the Simple U-Net model. The Simple U-Net model (shown in Figure 2.1) removed four convolution and four transpose convolution layers from U-Net, as well as the feedforward neural network between the encoder and decoder sections. By removing these layers, the architecture size, GPU requirements, and training time of Simple U-Net were significantly reduced compared to the conventional U-Net. The Simple-UNet model also provided accurate binary segmentations for microstructures of particles and pores.

Lastly, we further optimized the architecture of Simple-UNet towards RGB segmentations of microstructural images involving GBs. We made use of DENSE and optimized Simple U-Net in terms of the filter dimensions: (i) 1x1, (ii) 3x3, (iii) 5x5, (iv) 7x7, and (v) 9x9. We call the optimized CEDN for RGB segmentation of GBs DENSE-UNet. We note that the number of channels for DENSE-UNet were kept fixed as in Simple-UNet shown in Figure 2.1.

2.2.5 YOLOv5

Region-based Convolutional Neural Network (R-CNN) gave rise to ML techniques for object detection [57]. The R-CNN approach was significantly improved shortly after its introduction for the task of object detection by the YOLO (You Only Look Once) model. A key contribution of YOLO to the R-CNN approach was the use of a single regression CNN for predicting the best detection box size (i.e., window size) for the given image [58]. Since its introduction, the YOLO model has been refined. For instance, YOLO9000 increased the number of recognizable objects in a single image [59], YOLOv3 improved the prediction accuracy [60], YOLOv4 increased the speed of real-time prediction [61], and YOLOv5 leveraged previous high GPU memory requirements [62]. We note that the YOLO approach continues to be improved each year for faster, more accurate, and higher complexity real-time object detection tasks [63, 64]. Therefore, in order to detect particles and pores from the generated binary segmentation images, we made use of the YOLOv5 network in this work due to its low GPU memory requirements, ease of implementation, and portability.

2.2.6 Google Colab and Pytorch for training and optimization of ML models

Google Research provides a free and easy-to-use online programming platform called Google Colab. Google Colab is hosted by Jupyter Notebooks and grants users with up to 12 GBs of NVIDIA Tesla k80 GPU memory for up to 12 hours [65]. Additionally, one of the most recent and widely used libraries for programming ML projects is Pytorch [66]. Pytorch is formulated in Python language through the use of dynamic computation graphs. The Pytorch library provides a class of “tensors” (i.e., similar to “NumPy” arrays) However, these Pytorch “tensors” have CUDA capabilities in order for GPU operations which accelerate tensor computations and required training times [67]. Therefore, in this work we make use of the Pytorch library and the Google Colab platform to train and optimize the ML framework.

2.3 Microstructural feature characterization framework

2.3.1 Training data and augmentation procedure

In order to train the ML models in the framework, we generated two datasets involving (i) powder particles, and (ii) GBs. First, for powder particles we gathered an open-source dataset from [68] (i.e., through Mendeley Data) consisting of 2048 synthetic microstructure images. We note that the 2048 synthetic images were categorized into 8 groups of varying PSD. To generate the resulting labels of binary segmentations, we made use of the watershed and city-block distance (WCBD) method from [20]. The WCBD method also provided the bounding boxes defining each particle's size and position for training the YOLOv5 model. We emphasize that to extend the WCBD method from powder particles to pores the open-source WCBD algorithm from [20] was modified by inverting the binary values. In essence, we set the locations of the defects using black pixels, and the locations of the surrounding material using white pixels. For the training labels of the YOLOv5 model, we extracted the centroid and diameter of each powder particle. Ultimately, to develop the training dataset for YOLOv5 consisting of bounding boxes, we made use of the free online tool, "Roboflow.ai". Next, we gathered an additional dataset consisting of GBs from [69]. The dataset contained 59 (2048x1532) high-resolution microstructural images obtained through selective laser melting (SLM) of 316L stainless steel. To obtain a large training dataset consisting of 3776 GBs images, we first resized each high-resolution image to 2048x2048 dimensions, and then cropped each resized image to 256x256 separate smaller images. Lastly, we implemented the PSILM algorithm from [37, 38] to each 256x256 microstructural image and obtained their resulting RGB image segmentations, along with their histogram for grain size distribution.

Next we trained the Classifier CNN described in Section 2.2.1 using 4000 images of powder particles and GBs (i.e., 2000 images of powder particles, and 2000 images of GBs). We distributed the training, validation, and test datasets for GBs using 3800, 100, and 100 images, respectively. Similar to the cropping approach used for GBs, we cropped each of the 512x512 high-resolution synthetic images of powder particles into 256x256 smaller separate images. This resulted in a total of 8192 training images for powder particles. We then distributed the

training, validation, and test datasets for powder particles using 7850, 512, and 100 images, respectively. Additionally, we note that YOLOv5 requires significantly smaller datasets in order to reach high prediction accuracies [62]. Therefore, we set the training, validation, and test sets for the YOLOv5 model as 800, 100, and 100 images, respectively. Finally, we distributed the training, validation, and test sets used to train the DENSE-UNet for RGB segmentations, and the Regression CNNs for grain size distribution using 3560, 108, and 108 images, respectively.

2.3.2 Overview of framework architecture

A flowchart depicting the architecture of the developed framework is shown in Figure 2.2. As mentioned in previous sections, a key advantage of the framework is that users are not required to input additional specifications or prior knowledge of the defects type present for each image. To accomplish this autonomy, the first ML network involved a "Classifier CNN". The objective of the Classifier CNN was to predict whether the input microstructural image involved particles, pores, or GBs. Depending on which type of defect was predicted by the Classifier CNN, the framework involved two separate paths.

The first path (i.e., top path in Figure 2.2) was for particles and pores. The path for particles and pores included the Simple U-Net CEDN described in Section 2.2.4 for generating binary segmentations. The segmented images for particles and pores described the locations of the defects using black pixels, and the surrounding bulk material using white pixels as shown in Figure 2.2. The next ML model for particles and pores was the YOLOv5 network described in Section 2.2.5. The YOLOv5 network was used to predict the centroid location of each particle or pore, and a bounding box for their size (i.e., height and width). We note that the YOLOv5 was also capable of predicting centroid locations and bounding boxes for overlapping particles or pores.

The second path following the Classifier CNN (i.e., bottom path in Figure 2.2) was for grains and GBs. The first ML model for the bottom path of GBs involved the DENSE-UNet CEDN described in Section 2.2.4. The DENSE-UNet CEDN used the input microstructural image to generate RGB segmentations of the grains as shown in Figure 2.2). We note that particles and pores are typically scattered and nicely separated throughout the microstructure

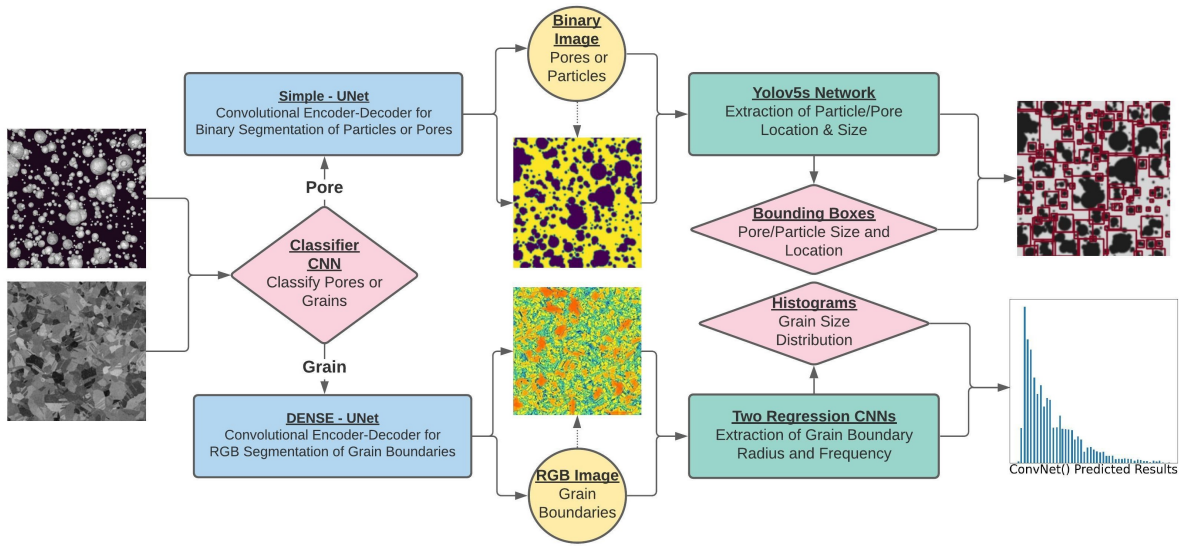


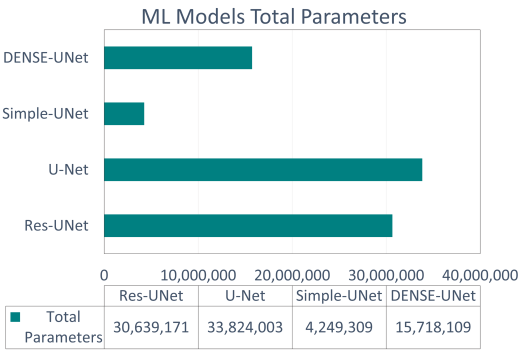
Figure 2.2: Diagram illustrating the structure of the machine learning framework for microstructural feature characterization

allowing the use of YOLOv5 to predict separate bounding boxes. However, gains and GBs are typically continuous in the material's microstructure. Therefore, while for particles/pores the YOLOv5 network was sufficient to extract the locations of each defect along with their size (i.e., bounding box), we needed to implement a different approach for grains and GBs. Instead, we made use of the color variations from the generated RGB segmentations. For instance, from Figure Figure 2.2 it can be seen that regions of darker red color depict the larger sized grains, while the regions of darker blue color depict the smaller sized grains. We took advantage of these color variations provided by the PSILM algorithm and implemented two Regression CNNs for extracting a histogram of the grain size distribution (i.e., radii and their frequencies).

2.4 Results and discussion

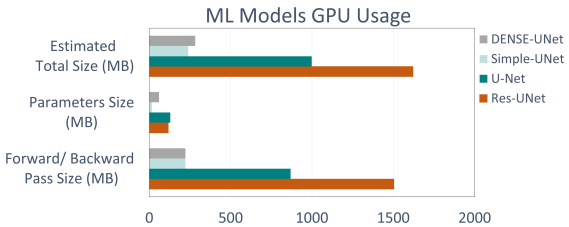
2.4.1 Number of training parameters and GPU requirements

To compare the computational costs of each CEDN model used towards the development of the framework (i.e., Simple-UNet, DENDE-UNet, U-Net, and Res-UNet), we computed the number of training parameters and their GPU MBs usage. The obtained number of training parameters and GPU usage of each model are shown in Figures 2.3a and 2.3b, respectively. For total number of training parameters, it can be seen that the state-of-the-art models U-Net and Res-UNet have



	Res-UNet	U-Net	Simple-UNet	DENSE-UNet
Total Parameters	30,639,171	33,824,003	4,249,309	15,718,109

(a) Number of trainable parameters



	Forward/ Backward Pass Size (MB)	Parameters Size (MB)	Estimated Total Size (MB)
DENSE-UNet	222.12	59.96	282.83
Simple-UNet	222.12	16.21	239.08
U-Net	868.5	129.03	998.28
Res-UNet	1504	116.88	1621.63

(b) GPU-Usage (MB)

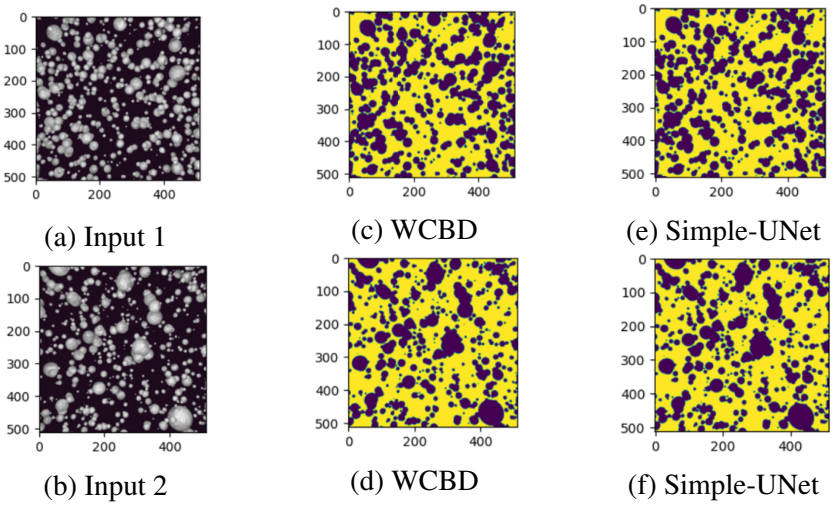
Figure 2.3: a) Total number of trainable parameters for each model and b) Total GPU-usage of each model.

the highest requirements of 33,824,003, and 30,639,171 parameters, respectively. As expected, the CEDN with the lowest required number of training parameters was the simplified Simple-UNet model (i.e., designed for binary segmentation of particles/pores) with only 4,249,309 (approximately 8x smaller than U-Net). Additionally, the second CEDN model with the lowest required number of training parameters was the DENSE (i.e., designed for RGB segmentation of GBs) with only 15,718,109 parameters (approximately 2x smaller than U-Net)

Next, we compare the GPU requirements of each CEDN model in Figure 2.3b. An interesting observation to make is that although the U-Net model involved the largest number of training parameters, the Res-UNet model shows higher GPU usage of 1621.63 MBs compared to the U-Net model of 998.28 MBs. We note that this higher GPU requirement of the Res-UNet is due to its higher forward/backward pass of 1504 MBs, compared to the U-Net of 868.5 MBs. Comparing the DENSE-UNet and Simple-UNet, both model show a significant decrease in GPU usage compared to the state-of-the-art CEDNs. For instance, the Simple-UNet shows the lowest GPU usage of 239.8 MBs (approximately 4x smaller compared to Res-UNet), followed by DENSE-UNet of 282.83 MBs (approximately 3.5x smaller compared to Res-UNet).

2.4.2 Predictions for powder particles and pores

First, we compare the prediction accuracy of the standard WCBD method against the developed Simple-Net model to generate binary segmentations of synthetic powder particles. The resulting binary segmentations using both methods are shown in Figure 2.4. From Figure 2.4, we see



(a) Input 1 (c) WCBD (e) Simple-UNet

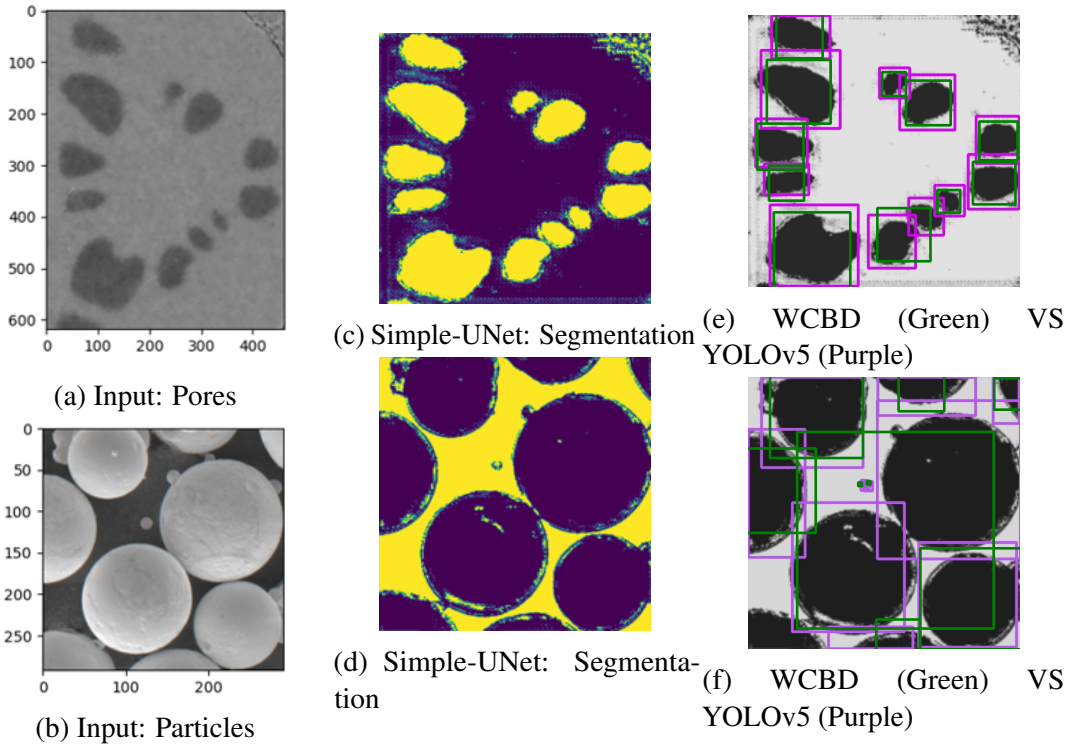
(b) Input 2 (d) WCBD (f) Simple-UNet

Figure 2.4: Comparison of particle binary segmentation obtained through the watershed and city-block distance segmentation methods versus the ML Simple-UNet.

that the predicted binary segmentations by the Simple-UNet model are virtually identical to the segmentations produced by the WCBD method.

To verify these observations, we performed a quantitative analysis by computing the average pixel error and the maximum pixel error from the Simple-UNet against the WCBD method following the approach presented in [49]. For this quantitative analysis, we first gathered 10 randomly chosen synthetic particle images and obtained their binary segmentations. For the predicted "black pixels" (i.e., 0 pixels) we obtained a maximum error of 0.65 $\%$ and average error of 0.15 $\%$. For the predicted "white pixels" (i.e., 1 pixels) we obtained a maximum error of 0.19 $\%$ and average error of 0.06 $\%$. These results all show maximum percent errors below 1%, therefore, the Simple-UNet model predicted binary segmentations for synthetic particles with high accuracy.

To further validate the framework's prediction accuracy, we gathered additional microstructural images from different distributions involving particles and pores. As shown in Figure 2.5, we show two microstructures of (i) austenite involving pores, and (ii) Ti-6Al-4v involving powder particles. We then used the Simple-UNet CEDN to generate their binary segmentations, and used them as input to the YOLOv5 network to predict the size and location of each defect. The predicted bounding boxes for pores are shown in Figure 2.5e, and the predicted bounding boxes for powder particles in Figure 2.5f using purple boxes. The bounding boxes generated



(a) Input: Pores

(b) Input: Particles

(c) Simple-UNet: Segmentation

(d) Simple-UNet: Segmentation

(e) WCBD (Green) VS YOLOv5 (Purple)

(f) WCBD (Green) VS YOLOv5 (Purple)

Figure 2.5: Results for the ML framework when tested on various distributions of microstructural images depicting pores and particles.

by the WCBD method are shown using green boxes in Figures 2.5e and 2.5f. It can be seen that the bounding boxes generated by the WCBD method are usually smaller with square shape, compared to the bounding boxes predicted by the YOLOv5 network. We note that a limitation of the WCBD method is that it generates a single diameter value for each defect, while the YOLOv5 network is able to predict the height and width of each detected defect.

For additional quantitative validation of these results, we generated Tables 2.1, 2.2 and 2.3. To gather the ground truth bounding boxes for the austenite and Ti-6Al-4V microstructures we made use ”CVAT.org”. ”CVAT.org” is an open-source online labeling tool for object detection problems. In Table 2.1 we compare the total number of detected defects from WCBD and YOLOv5, against the ground truth obtained using ”CVAT.org”. From Figure 2.5e and Table 2.1, the austenite microstructure involves 12 pores. We note that the YOLOv5 network successfully predicted a total of 12 pores for austenite as seen in Table 2.1. However, the WCBD algorithm only detected 11 pores. For particles it can be observed from Figure 2.5f that the Ti-6Al-4V sample includes a total of 10 particles. The YOLOv5 network and WCBD method did not capture the particle located towards the bottom left of the microstructure. Additionally, a key

Number of Predicted Defects		
	Number of Particles	Number of Pores
Ground Truth	10	12
WCBD Method	8	11
YOLOv5 Method	9	12

Table 2.1: Comparison of the number of detected bounding boxes for powder particles in Ti-6Al-4V, and pores in Austenite, using the YOLOv5 network and conventional WCBD method versus the ground truth.

Particles Microstructure - Ti-6Al-4V				
Bounding Boxes Error				
	Center (X)	Center (Y)	Width	Height
WCBD $\%$ Error (Avg. $\pm$ Std. Dev.)	5.45 ± 7.88	13.29 ± 28.43	44.76 ± 28.22	51.19 ± 34.20
YOLO $\%$ Error (Avg. $\pm$ Std. Dev.)	3.34 ± 4.97	10.15 ± 28.80	17.42 ± 30.46	19.59 ± 29.94

Table 2.2: Percentage error for the predicted bounding boxes parameters (centroid locations, height, and width) in powder particles of Ti-6Al-4V, using the YOLOv5 network and the conventional WCBD method compared to the ground truth.

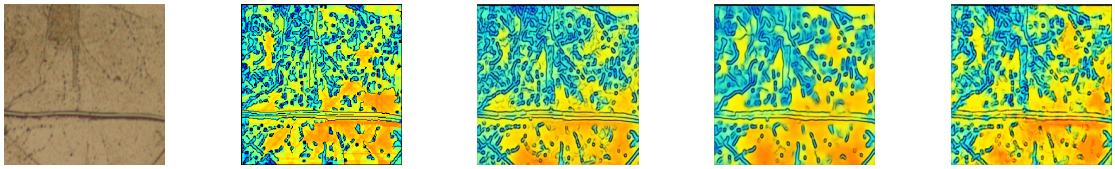
observation to make is that the WCBD method generated a single bounding box for the two largest overlapping particles located at the center of the microstructure. However, the YOLOv5 network accurately predicted a single bounding box for each of particle.

To further analyze these results, we show the resulting percent errors for the YOLOv5 network and WCBD method compared to the ground truth in Tables 2.2 and 2.3. Here, we computed the difference in distance between the predicted and the ground truth centroid, height, and width of each defect. As shown in Table 2.2, because the WCBD method typically generated larger bounding boxes compared to YOLOv5 for powder particles in Ti-6Al-4V, the percent errors of WCBD were higher for the centroid, height, and width parameters. While the percent errors for YOLOv5 are significantly lower than the WCBD method for Ti-6Al-4V, it also showed high errors of approximately 17 $\%$ for the width, and 19$\%$ for the height. We note that the errors in Table 2.2 account for errors from the particle located at the bottom left of the microstructure not being detected, thus, significantly increasing the errors.

Next, the percent errors for the YOLOv5 network and WCBD method in the predicted centroid, height, and width of each pore in austenite are shown in 2.3. Here, we computed the

Pores Microstructure - Austenite				
Bounding Boxes Error				
	Center (X)	Center (Y)	Width	Height
WCBD % Error (Avg. $\pm$ Std. Dev.)	6.06 ± 18.58	6.79 ± 21.31	27.46 ± 25.05	18.82 ± 27.30
YOLO % Error (Avg. $\pm$ Std. Dev.)	0.59 ± 0.59	0.78 ± 0.58	11.74 ± 7.39	22.52 ± 12.15

Table 2.3: Percentage error for the predicted bounding boxes parameters (centroid locations, height, and width) in pores of Austenite, using the YOLOv5 network and the conventional WCBD method compared to the ground truth.



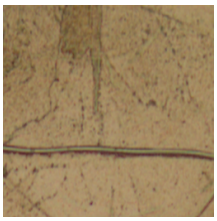
(a) Input (b) PSILM (c) Res-UNet (d) U-Net (e) DENSE-UNet

Figure 2.6: Comparison of grain boundary segmentation results using ASTM's PSILM, Res-UNet, U-Net, and DENSE-UNet models.

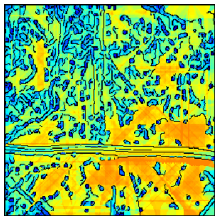
difference in distance between the Here, the WCBD method also resulted in higher percent errors compared to the YOLOv5 network for the centroid coordinates, and width parameters. However, the YOLOv5 network shows higher error for the height prediction compared to the WCBD method. The highest accuracy is shown by YOLOv5 for predicting the center coordinates at approximately 3.2 % error, while the the WCBD method resulted in approximately 10.5 % error. Ultimately, these results show that the YOLOv5 predicted a higher number of defects for both austenite and Ti-6Al-4V samples, and predicted the center, height, and width with overall higher precision compared to the WCBD method.

2.4.3 Predictions for grain boundaries

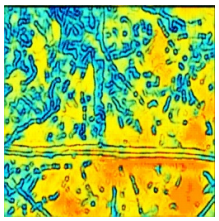
We then analyzed the prediction accuracy of state-of-the-art CEDNs (i.e., Res-UNet, U-Net) and the developed optimized CEDN (i.e., DENSE-UNet) to generate RGB segmentations for GBs. We also compared the prediction accuracy of these models versus the conventional PSILM technique. The generated input microstructure of GBs, and the resulting RGB segmentations of each model are shown in Figure 2.6.



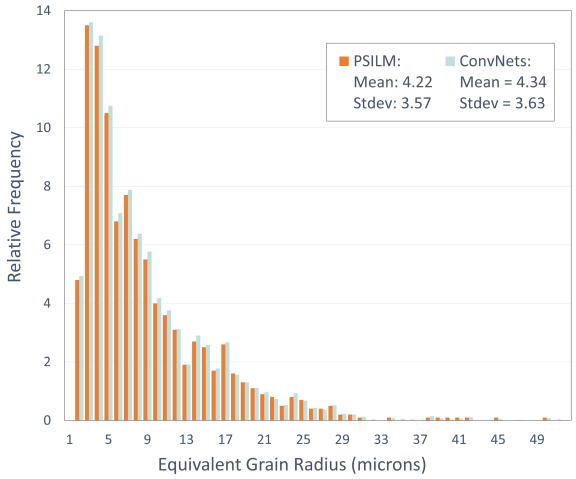
(a) Grain Boundary



(b) PSILM Output



(c) DENSE-UNet Output



(d) PSILM vs ML Framework: Grain boundary size distribution histograms

Figure 2.7: Comparison of grain boundary size distribution histograms obtained from both the PSILM method and the ML framework.

From Figure 2.6, it can be seen that all the CEDNs accurately predicted the grain sizes as darker red for larger grains, and darker blue for smaller grains. However, Figure 2.6b shows that the PSIM did not fully capture various larger sized grains. The PSILM algorithm relies on conventional image-processing edge detection tools. Once the PSILM performs edge-detection, a number of random points are positioned over the detected edges. The line intercept method [38, 37] is then implemented to compute the horizontal, vertical, and diagonal lengths between neighboring points. Finally, the computed lengths are interpolated in order to formulate the grain sizes and generate the resulting color ranges (i.e., RGB segmentation). The errors shown in Figure 2.6b for the PSILM algorithm may originate from conventional edge detection tools. Conventional edge detection tools are known to incorrectly label lighter shadows in the images as small grains [70], thus, omitting various larger sized grains. Additionally, the line intercept method does not account for larger sized grains positioned around the edges of the microstructure, thus, incorrectly labeling them as smaller sized grains.

Next we compared the grain size distribution histograms generated by the PSILM algorithm versus the framework in Figure 2.7. The PSILM resulted in average grain size (radius) of 4.22 ± 3.75, while the framework predicted grain size of 4.34 ± 3.63. We note that the ML framework predicted overall grain size close to the PSILM approach, it predicted higher frequencies

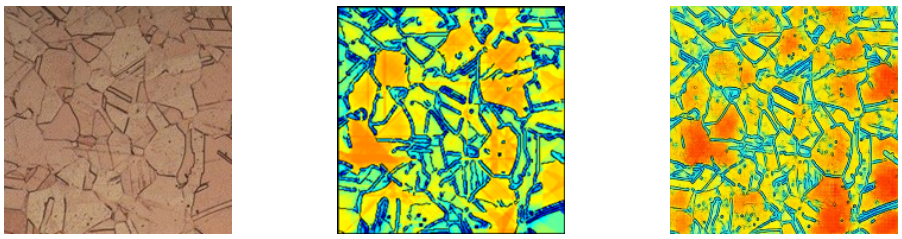
compared to the PSILM histogram. This may be due to the conventional edge detection tool from the PSILM not capturing some of the larger sized grains as mentioned previously. In contrast, it can be seen from Figure 2.7c that the DENSE-UNet model captures a higher number of larger sized grain sections.

For further validation of the DENSE-UNet model, we analyzed its prediction accuracy on microstructural images of GBs obtained from a different distribution. We gathered four GB microstructures for (i) Copper Alloy in Figure 2.8a, (ii) Titanium in Figure 2.8b, (iii) an Inconel Alloy in Figure 2.8c, and (iv) Steel in Figure 2.8d. We then generated their resulting RGB segmentations using the PSILM algorithm as shown in Figures 2.8e, 2.8f, 2.8g, and 2.8h, and the DENSE-UNet as shown in Figures 2.8i, 2.8j, 2.8k, and 2.8l. Similar to Figure 2.7b, the PSILM struggles to capture some of the larger sized grains, especially near the edges of the microstructures.

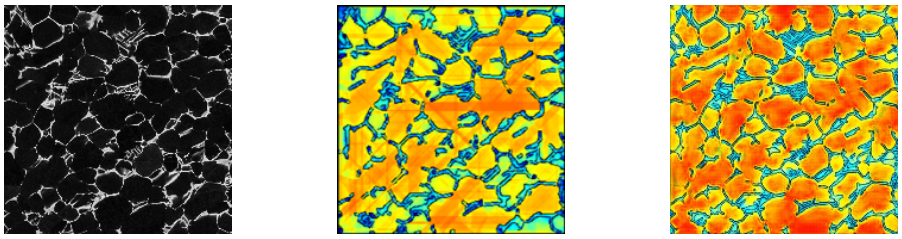
A clear example of this error is shown for steel by comparing the PSILM RGB segmentation versus the DENSE-UNet model in Figures 2.8h and 2.8l. Here, multiple larger sized grains are captured as smaller grains by the PSILM. However, the DENSE-UNet model accurately captured the larger grains, thus, outperforming the PSILM for steel. Therefore, these results show that the optimized CEDN model, DENSE-UNet, was able to generate accurate RGB segmentations. The DENSE-UNet model was able to distinguish larger grains and label them using darker red colors, and smaller grains using darker blue colors.

2.4.4 Training time and accuracy of CEDNs for RGB segmentation

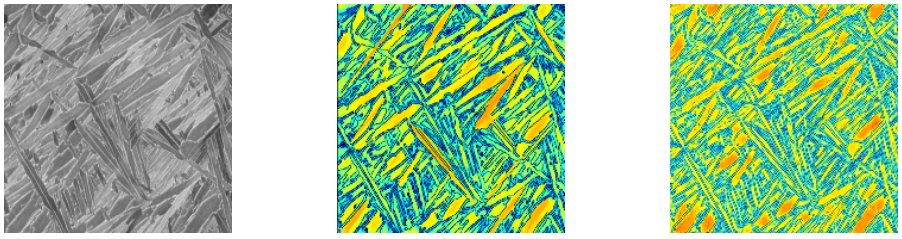
We obtained training time required for each CEDN model used for RGB segmentations of GBs as shown in Figure 2.9a. From Figure 2.9a, we note that the longest training time was obtained for the Res-UNet model at 5:34 hours. As shown in Section 2.4.1 and Figure 2.3b, the Res-UNet also resulted in the highest GPU usage. The next model showing the longest training time was U-Net requiring 3:55 hours. Additionally, from Figure 2.3b we recall that that the CEDN for RGB segmentation with the lowest GPU usage was DENSE-UNet, thus, resulting in the lowest training time of 55 minutes. We note a significant decrease of training time compared to the state-of-the-art Res-UNet (approximately 6x faster).



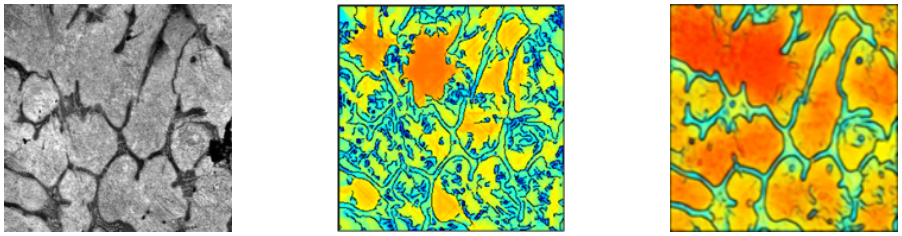
(a) Input: Copper (e) PSILM: Copper (i) DENSE-UNet: Copper



(b) Input: Titanium (f) PSILM: Titanium (j) DENSE-UNet: Titanium

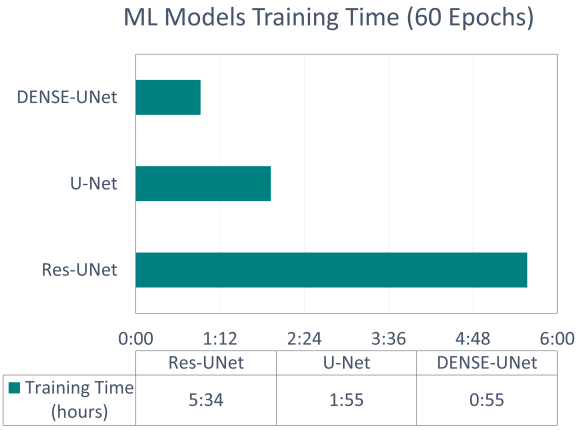


(c) Input: Inconel Alloy (g) PSILM: Inconel (k) DENSE-UNet: Inconel

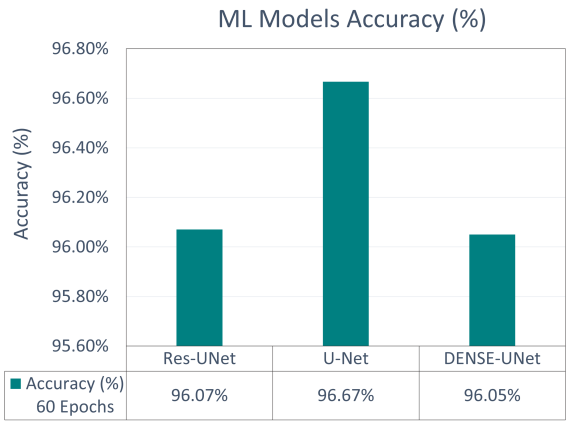


(d) Input: Steel (h) PSILM: Steel (l) DENSE-UNet: Steel

Figure 2.8: ML framework evaluated across various distributions of microstructural images depicting grain boundaries.



(a) Training time (min:sec) tested for 60 Epochs



(b) Accuracy (%) tested for 60 epochs

Figure 2.9: Comparisons of training time and accuracy across different ML encoder-decoder networks.

Furthermore, we performed a quantitative analysis for the accuracy of each CEDN model to generate RGB segmentations of GBs (Figure 2.9b). From Figure 2.9b, we observe the highest accuracy for U-Net at approximately 96.60%, while the most computationally demanding CEDN Res-UNet resulted in lower accuracy of approximately 96.05% While the Res-UNet requires higher GPU-Usage, we recall from Figure 2.3a that the U-Net model involved a larger number of training parameters which may aid towards higher prediction accuracy. For the optimized DENSE-UNet, the accuracy is nearly parallel to the Res-UNet model at approximately 96.05%. However, a significant advantage of DENSE-UNet is its reduced training time of 55 minutes compared to Res-UNet of 5:34 hours. Therefore, we conclude that the optimized DENSE-UNet is the most efficient model for RGB segmentation of GBs while achieving similar high accuracy compared to state-of-the-art U-Net and Res-UNet.

2.4.5 Time performance of the entire framework versus PSILM

Lastly, in Figure 2.10 we analyzed the time performance of the developed framework to characterize 10 images of GBs microstructures and compared it to the PSILM algorithm. We note that this analysis did not include the required training time due to ML models allowing us to save pretrained models. Therefore, we first accounted for the times required to load the pretrained weights of the Classifier CNN, the developed CEDNs (e.g., Simple-UNet, and DENSE-UNet),

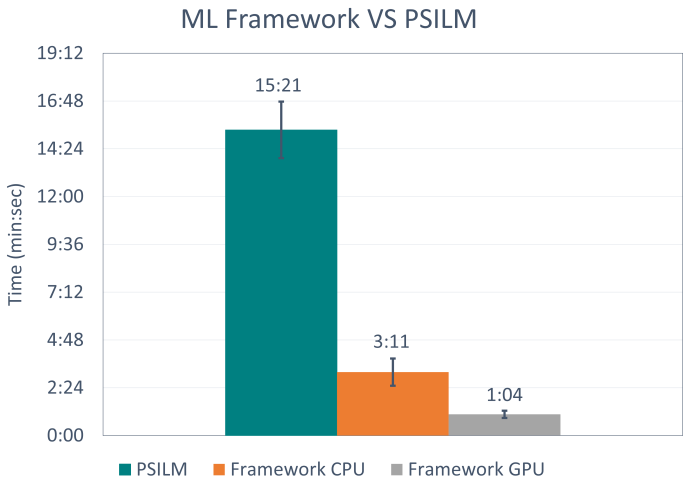


Figure 2.10: Analysis time comparison between the ML Framework and PSILM for 10 microstructural grain boundary images.

and the Regression CNNs. We then recorded the time from initial loading of the pretrained weights to the generation of the histograms of grain size distribution. Using this approach we recorder the total average time using both CPU and GPU. Following this method, we computed the analysis time of the PSILM algorithm from the initial input image to the final RGB segmentation and histogram generation. As shown in Figure 2.10, the PSILM required 15:21 minutes to fully characterize 10 microstructures (i.e., approximately 1:32 minutes per image). In contract, the framework resulted in only 3:11 minutes (i.e., approximately 19 seconds per image) to characterize the same 10 microstructures when using CPU. Using GPU, the proposed ML framework performed significantly faster with only 1:04 minutes total (i.e., 6.4 seconds per image).

Considering the required training times of the ML framework, the PSILM may prove to be a better option for characterizing smaller datasets. For instance, to characterize a dataset involving 36 microstructures using the PSILM approach a total 55 minutes would be required. In contrast, to train the framework from scratch and perform the same analysis, would require more than 55 minutes. As a result, we emphasize that when training is required, the framework is only faster than the PSILM approach for datasets involving more than 36 microstructural images. As an additional comparison, to characterize 100 microstructures the PSILM approach requires around 2:33 hours. However, the ML framework using CPU only would require approximately

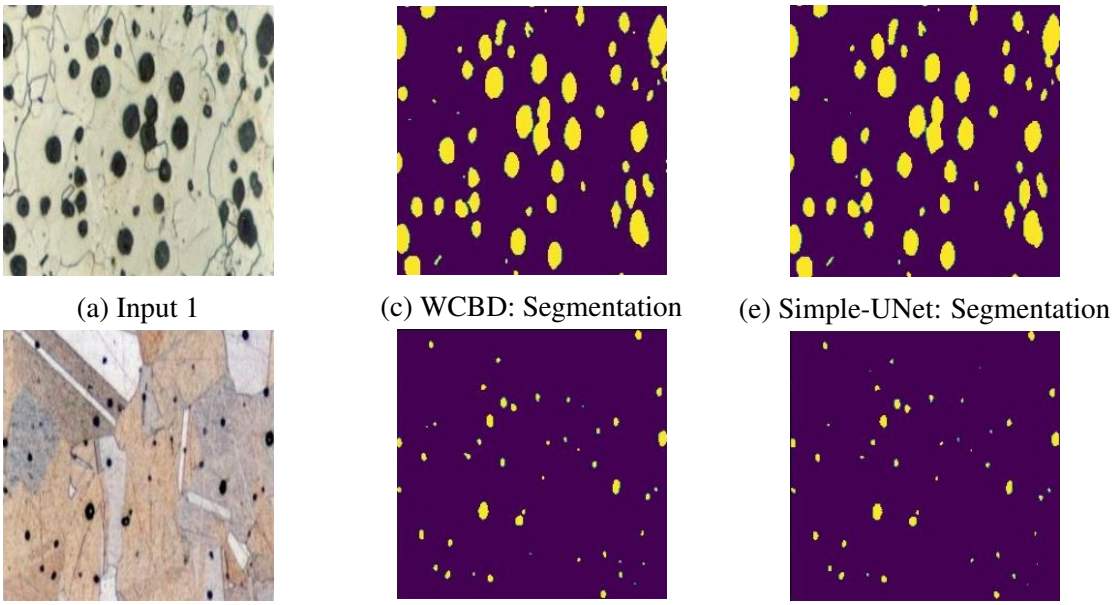
44 minutes to train, followed by 32 minutes to characterize 100 microstructures. This shows a performance improvement of 1:16 hours by the framework.

Ultimately, we emphasize that the framework does not need prior user information for the types of microstructural defects in each image. For datasets involving mixed microstructures of pores, particles, and GBs the developed framework is capable of autonomously separating each microstructure prior to characterizing each defect. This advantage of the proposed ML framework would also save significant time for analysts and material scientists. While the proposed ML framework is not a replacement for conventional methods such as PSILM, it is a useful and computationally efficient tool for autonomous characterization of large datasets involving different materials and microstructural features.

2.4.6 Case of microstructures involving multiple defects

While the current ML framework performs accurate characterization of microstructures with a single type of defect (i.e., particles, pores, or GBs), it does not consider single microstructures with multiple types of defects (e.g., pores and GBs). We address this challenge by testing the framework's capability to characterize datasets of microstructures involving both pores and GBs. We obtain a dataset involving 354 microstructures of materials with pores and GBs from the DoITPoMS - Micrograph Library. We resize the low resolution images to 512x512 and the high resolution images to 1024x1024. We then crop each image using dimensions of 256x256, and perform random rotation, and vertical and horizontal flipping for data augmentation. The final mixed microstructure dataset resulted in 1416 images. Using the approach described in Section 2.3.1, we obtained the binary segmentations and bounding boxes for the pores, and then obtained the RGB segmentations for the GBs. We also implemented transfer learning to the CEDNs (i.e. Simple-UNet and DENSE-UNet) and the object detection YOLOv5 network for the new dataset of mixed microstructures. We emphasize that an advantage of the PSILM method is its ability to segment pores as black pixels, thus, removing them from the final grain size measurements

In Figure 2.11, we show the resulting binary segmentations from Simple-UNet. The Simple-UNet qualitatively shows virtually indistinguishable segmentations compared to the WCBD method. Next, we compared the bounding boxes generated by the YOLOv5 network versus



(a) Input 1 (c) WCBD: Segmentation (e) Simple-UNet: Segmentation

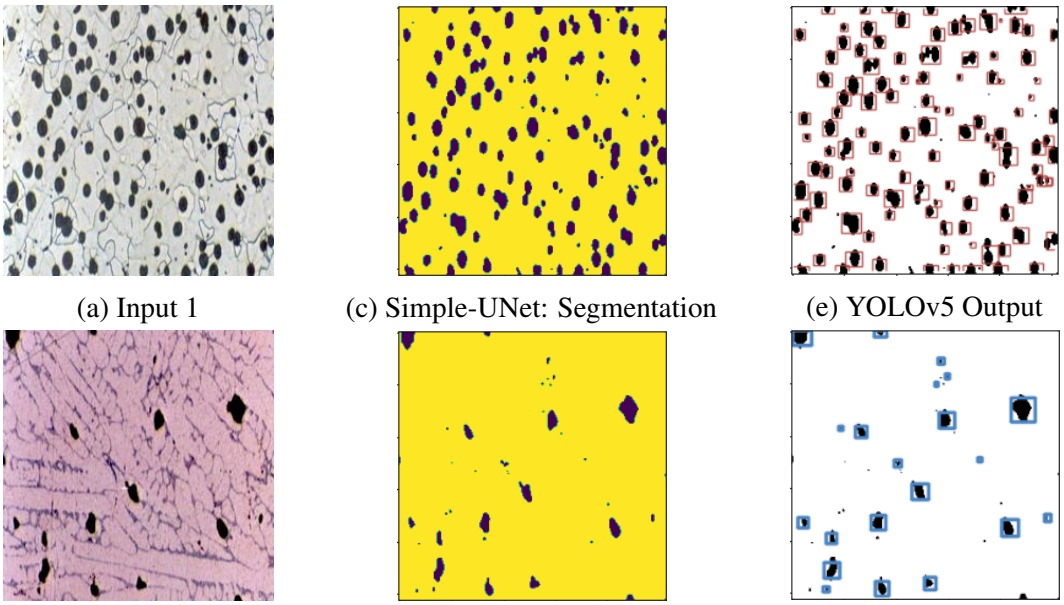
(b) Input 2 (d) WCBD: Segmentation (f) Simple-UNet: Segmentation

Figure 2.11: Comparison of binary segmentation utilizing the WCBD method against Simple-UNet for microstructural images containing grain boundaries and pores.

the WCBD method in Figure 2.12. The YOLOv5 network predicted the bounding boxes with high accuracy compared to the WCBD bounding boxes, thus, demonstrating the framework's capability to extract pore size and location in microstructures with multiple defects.

Next, Figure 2.13 first demonstrate the advantage of the PSILM approach to segment pores using black pixels, and the GBs using color variations. Additionally, it can be seen that the framework generated RGB segmentations for the pores and GBs with good accuracy compared to the PSILM algorithm. These results show that the proposed ML framework may be extended in future work for simultaneous characterization of microstructures with multiple defects.

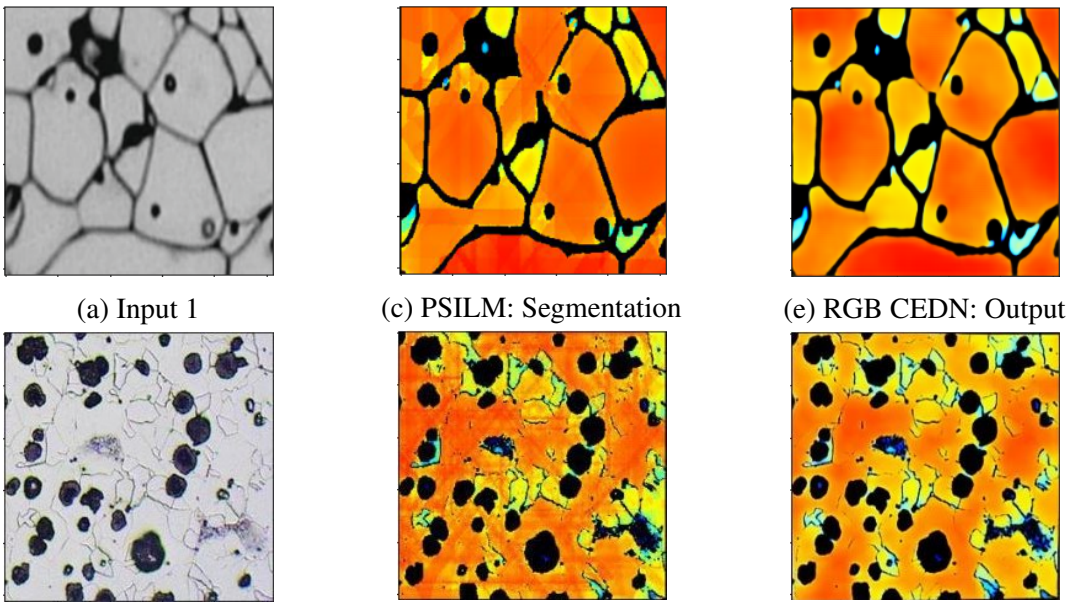
However, there are several limitations in the current work which may need to be further studied. First, the current ML framework is autonomous for microstructures involving a single type of defect. We recall from Sections 2.3.1 and 2.3.2 that the Classifier CNN was trained using 2000 images of pores and 2000 images pf GBs. However, the dataset gathered of microstructures with both GBs and pores consist of only 1239 images. Therefore, developing a Classifier CNN capable of predicting microstructures involving two different types of defects will require compiling larger training datasets.



(a) Input 1 (c) Simple-UNet: Segmentation (e) YOLOv5 Output

(b) Input 2 (d) Simple-UNet: Segmentation (f) YOLOv5 Output

Figure 2.12: Evaluation of YOLOv5's performance on microstructures containing grain boundaries and pores.



(a) Input 1 (c) PSILM: Segmentation (e) RGB CEDN: Output

(b) Input 2 (d) PSILM: Segmentation (f) RGB CEDN: Output

Figure 2.13: Performance evaluation of the Encoder-Decoder network for RGB segmentation on microstructures containing grain boundaries and pores.

2.5 Conclusion

The developed ML framework resulted in improvements compared to conventional microstructure characterization methods based in image-processing, as well as existing state-of-the-art CEDNs for image segmentation. We developed an autonomous and complete feature extraction process for microstructures involving particles, pores, and GBs through the integration of multiple ML models. Standard computational tools such as the PSILM algorithm and the WCBD method were crucial in order to gather the training dataset. However, the final trained framework can be used autonomously in future work without the need for additional training. The first step of the framework towards autonomy involves a Classifier CNN for predicting the types of defects pertaining to each microstructure (i.e., particles, pores, or GBs). Depending on the predicted defect from the Classifier CNN, we then developed two separate CEDNs (i) Simple-UNet for binary segmentation of particles or pores, and (ii) DENSE-UNet for RGB segmentation of GBs. Finally, the generated binary segmentations are used as input to the object detection YOLOV5 network for predicting the particles/pores size and location, while the generated RGB segmentations are used as input to two Regression CNNs for predicting histograms of the grain size distribution.

We emphasize that a time-consuming and tedious challenge in supervised ML methods involves the generation of the training datasets. Recent online tools and resources aimed at ML have significantly reduce the need for expensive GPUs. For example, the free online tool Google Colab grants users with up to 12 GBs of GPU memory for 12 hours. This tool can be used to train state-of-the-art complex ML models like Res-UNet and U-Net without the need to purchase expensive GPUs. We note that although these tools are useful for ML applications, in order to achieve high prediction accuracy some models may require large training datasets, thus, limiting the GPU usage time. Additionally, the development of ML frameworks with exportability capabilities requires further reduction of training time and GPU requirements. This is a limitation in current state-of-the-art complex ML models due to their high GPU usage and prolonged training times.

Therefore, in this work aimed at optimizing state-of-the-art models such as U-Net and Res-UNet for RGB segmentation of GBs using the DENSE algorithm. Using DENSE, we reduced the GPU usage of these models by 716 MBs (approximately 4x lower) compared to U-Net. Additionally, we reduced the required training time to 55 minutes (approximately 6x faster) compared to Res-UNet at 5:34 hours, while maintaining high accuracy. We then showed good prediction by the developed DENSE-UNet to generate RGB segmentation of GBs in titanium, copper alloy, steel, and Inconel alloy. For pores and particles, the Simple-UNet model showed high prediction accuracy for predicting binary segmentations similar to the standard WCBD method. Lastly, the framework involved the object detection YOLOv5 network which predicted the size and location of particles and pores with high accuracy in austenite and Ti-6Al-4V.

Although the autonomous framework showed promising results for microstructure characterization problems, there were certain limitations. While the framework showed faster performance compared to WCBD and PSILM methods, the model may fall short in performance when training the framework from scratch is required. If training is required, the framework outperforms the PSILM method for datasets involving more than 36 images. Another limitation is that the framework is currently designed to characterized microstructures involving a single type of defect (i.e., particles, pores, or GBs). We demonstrated that the framework could be extended to handle microstructures involving both GBs and pores. For instance, the Simple-UNet and DENSE-UNet predicted binary and RGB segmentations for these microstructures with good accuracy. Also, the YOLOv5 network predicted bounding boxes for the pores locations accurately. However, the Classifier CNN model did not include cases with two types of microstructural defects. In order to establish a new autonomous framework capable of predicting microstructures with more than one type of defect, a larger dataset for these cases is required in future work. Lastly, the framework currently handles processed images rather than raw experimental images or data. For example, data collected using SEM requires additional processing in order to generate the microstructural used as input in the developed framework. In order to handle inputs of raw data, new ML algorithms must be developed and integrated into the framework.

2.6 Data availability statement

Supplementary material and code available at:
https://github.com/rperera12/MachineLearning_Framework_

Microstructure_Characterization

Chapter 3

Graph neural networks for simulating crack coalescence and propagation in brittle materials

3.1 Introduction

Understanding the interaction, initiation, and propagation of microcracks in engineering materials is critical in order to evaluate material performance and structural longevity. Computational fracture mechanics models play a significant role in modeling material failure due to microcrack propagation and coalescence. Widely used state-of-the-art computational fracture mechanics models used to simulate crack behavior in materials include meshfree methods [71], cohesive zone modeling (CZM) [72, 73, 74, 75], scaled boundary finite element method (SBFEM) [76, 77, 78, 79], phase-field modeling (PFM) [80, 81, 82], and extended finite element methods (XFEM) [83, 84, 85]. An extensive study and comparative overview for these approaches can be found in [86, 87]. While these methods have shown success across various applications, they quickly become computationally expensive for higher-complexity problems (e.g., number of cracks, type of material, and loading configuration). For example, simulating microstructural-scale fracture in materials with high number of microcracks (i.e., multiple microcrack propagation and coalescence) is computationally demanding. In [88], it was demonstrated that modeling fracture due to multiple microcracks in a realistic three-dimensional domain of $1\ m^3$, may require several CPU days.

Reduced-order modeling techniques offer a promising approach to mitigate these challenges and reduce computational costs [89, 90, 88]. Over the past years, a reduced-order modeling technique which has shown significant attention across the solid mechanics and materials science community involves Machine Learning (ML) methods[91, 92, 93, 94, 95, 96, 97, 98, 99, 2, 100,

101, 102, 103, 104, 105, 106, 107, 108, 109]. Recently, ML models have been developed for predicting the stress and strain fields in composites [110, 111], stress hotspots for different crystal configurations [112, 113], behavior of nonlinear materials [114, 110], and tensor decomposition of complex materials [115]. Specific to fracture mechanics problems, ML models have also been implemented for various studies [116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127]. In [128] the authors used image-based ML techniques to simulate crack growth in graphene materials. Other studies have successfully implemented ML approaches to predict displacement of notched plates [129], and predict structural response in materials due to crack growth [130]. Recently, in [75, 88] a graph-theory-inspired artificial neural network was developed capable of predicting the final crack coalescence and time of failure in brittle materials. However, the developed ML model involved considerable errors compared to the high-fidelity model Hybrid Optimization Software Suite (HOSS) [131, 132], and was only able to predict the final configurations of microcrack coalescence. Additionally, in [133] the authors introduced ConvLSTM - a convolutional long-short term memory network - for modeling crack growth in brittle materials with similar results to a high-fidelity molecular dynamic (MD) simulation model. Here, the initial crack configurations were depicted as image-based inputs using binary images. The binary images defined the bulk material using white pixels and the cracks using black pixels. While these studies have integrated ML techniques to fracture mechanics problems, simulating the dynamic evolution of stresses and microcrack propagation for higher-complexity cases involving a large number of initial microcracks has not been explored.

A recent ML technique that has demonstrated significant speedup and accuracy to simulate various complex physics phenomena is Graph Neural Networks (GNNs). GNNs integrate Multi-Layer Perceptron (MLP) networks from ML approaches, and graph theory from mathematics. In typical artificial neural networks, the trainable weights are constructed using the architecture of the model (i.e., hidden layers, number of nodes, etc.). The resulting weights are then updated through multiple operations of stochastic backward gradient descent [134, 135, 136]. However, in the GNN approach the trainable weights are constructed using the graph representation itself [137] (i.e, node and edge connections). In 2020, [138] presented a dynamic GNN framework able to simulate the interaction of fluids (e.g., water, sand, etc.) and rigid bodies. The framework

defined the graph using up to 20k nodes for the fluid particles, and edges connecting the neighboring nodes. The edges in the graph included information for the node interactions (e.g., gravitational force, momentum, and/or energy). Message-passing graph convolutions were then used to generate latent space embeddings from the graphs of previous time-steps and the particles accelerations and positions were predicted at future time-steps. The graph-based implementation of GNNs (i.e., connecting nodes and edges) has also shown to be a suitable candidate for engineering problems. For instance, GNNs were recently studied for inverse material design of glass structures, to extract Perovskite synthesizability, and predict material properties [139, 140, 141, 142, 143, 144, 145, 146, 147]. Additionally, GNNs have been implemented for simulating various engineering problems. In [148], a GNN model was introduced to simulate the responses of homogenized polycrystals. The graph representation consisted of nodes for each crystal within the polycrystal structure, and edges to connect the crystals with sharing faces. The final GNN model was trained on phase-field fracture models to simulate the anisotropic response, and predict the resulting energy functional. Other recent works that have implemented GNNs for simulating various engineering problems include prediction of stresses in anisotropic elastoplastic materials, molecular properties in inorganic materials, grain-scale toughness in meso-scale experiments, craquelure fracture patterns, and metal organic CO2 adsorption properties [149, 150, 151, 152, 153, 154]. Moreover, GNNs have also been developed to forecast molecular dynamics of rotationally-covariant and translationally-invariant local atomic environments [155], finite element (FE) convergence of elastostatic problems [156], the dynamics of solid mechanics and fluid mechanics problems [148, 157, 3, 158, 159], and the graph structures of 3D boundaries for granular flow processes [160].

Therefore, in this chapter we develop a GNN-based framework called *Microcrack-GNN* to simulate the stress evolution and microcrack propagation in brittle material with multiple microcracks. We design the Microcrack-GNN framework to handle higher-complexity systems involving varying number of microcracks (i.e., from 5 to 19). As shown in Figure 3.1, the architecture of Microcrack-GNN includes four separate GNNs to model the underlying physics of material fracture due to multiple microcracks. From the XFEM methodology, the Mode-I and Mode-II effects are superimposed inn order to compute the stress distribution in the domain

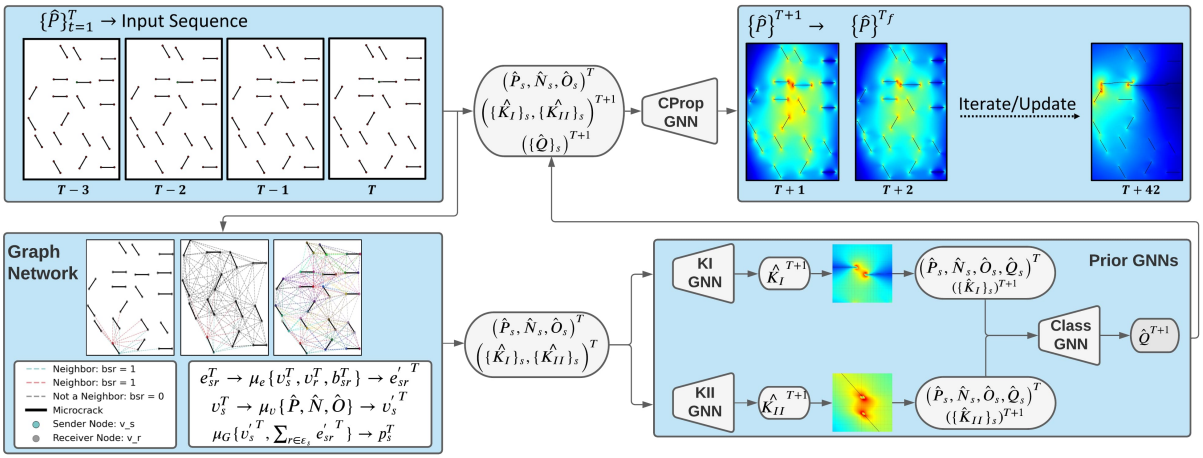


Figure 3.1: Structure diagram of the Microcrack-GNN framework

[161]. The resulting stress distribution is then used to determine the propagating crack-tips (i.e., cracks-tips with highest stresses). To capture this relation, we include the first two GNNs of the framework, (i) K_I-*GNN* to predict the Mode-I stress intensity factors, and (ii) K_{II}-*GNN* to predict the Mode-II stress intensity factors. We then include the third GNN of the framework, (iii) *Class-GNN* for predicting propagating and non-propagating crack-tips. We note that *Class-GNN* aims to capture the quasi-static nature of the problem. Lastly, the fourth GNN of the framework, (iv) *CProp-GNN*, predicts the future crack-tip positions using the predictions from the K_I-*GNN*, K_{II}-*GNN*, and *Class-GNN* models as input. A key feature of Microcrack-GNN is its ability to predict the evolution of microcrack propagation and coalescence, as well as the stress distribution using K_I-*GNN* and K_{II}-*GNN*. The framework provides a new data-driven ML approach for modeling higher-complexity fracture problems involving multiple microcracks with reduced computational costs and simulation times.

3.2 Methods

3.2.1 High-fidelity XFEM simulator

In order to simulate various fracture mechanics cases involving multiple microcracks in brittle materials, we make use of the open-source XFEM-based simulator from [162, 163, 164]. The XFEM-based model is written in MATLAB to model fracture in two-dimensional brittle material domains with multiple microcracks of arbitrary length and orientation. The XFEM

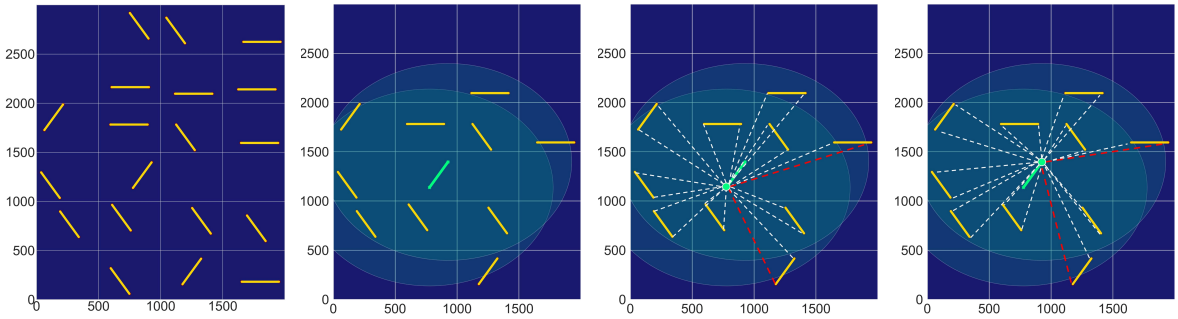
model allows the use of different crack growth criteria, including (i) maximum hoop stress, (ii) minimum total energy, and (iii) symmetric localization criteria. The high-fidelity model also considers changes in fracture topology to calculate important parameters such as the stiffness matrices, the timewise force vector, and crack-tips mesh-enrichment. Accounting for fracture topology to obtain compute these parameters helps reduce computational cost, thus, speeding up computations. Two additional key contributions of the XFEM model are the use of the resulting surface pressure and residual stress/strain of each crack to compute the stress intensity factors (i.e., K_I and K_{II}), and the use of the domain-form interaction integral approach [165, 166]. Therefore, to generate a large training dataset of two-dimensional simulations for fracture in brittle materials with multiple microcracks, we use the high-fidelity XFEM model.

3.2.2 Graph representation

For the graph representation of Microcrack-GNN, we define the graph as $\langle \mathbf{V}, \mathbf{E} \rangle$, where $\mathbf{V}$ includes all crack-tips as vertices/nodes, and $\mathbf{E}$ includes all the edges in the system. $\mathbf{E}$ involves edges connecting each crack-tip $v_s \in \mathbf{V}$ to their neighboring crack-tips within a zone of influence, as well as edges connecting each crack-tip $v_s \in \mathbf{V}$ to the remaining non-connecting crack-tips outside the zone of influence. In essence, $s : \{1, 2, \ldots, 2C\}$ for all positive integers, where C defines the total number of microcracks in the system) We show the defined graph representation in Figures 3.2a - 3.2b.

For a sequence of previous time-steps $\hat{\mathbf{T}} := \{T-3,\ T-2,\ T-1,\ T\}$, we define the crack-tip vertices using their positions, $\hat{P}_s$, their nearest-neighbors $\hat{N}_s$, and their orientation at the initial time-step, $\hat{O}_s$ (i.e., $\{\theta_s\} = 0^o$, 60^o, or 120^o).

$$
\begin{aligned}
\hat{P}_s &= \{(x_s, y_s)\}_{t=T-3}^{T} \qquad \{s \in \mathbf{V}\}, \\
\hat{N}_s &= \{N_s\}_{t=T-3}^{T} \qquad \{s \in \mathbf{V}\}, \\
\hat{O}_s &= \{\theta_s\}_{t=T-3}^{T} \qquad \{s \in \mathbf{V}\}, \\
\{v_s\}^{\hat{T}} &= \left(\hat{P}_s, \hat{N}_s, \hat{O}_s\right) \qquad \{s \in \mathbf{V}\}.
\end{aligned} \tag{3.1}
$$



(a) Initial Configuration (b) Crack-9 Neighbors (c) Vertex-1 Neighbors (d) Vertex-2 Neighbors

Figure 3.2: Diagram illustrating the connections between the current crack tip (vertex) and its neighboring crack tips (edges) for a single crack.

Here, (x_s, y_s) are the Cartesian coordinate positions, and θ_s the orientations (in radians) for node $s \in \mathbf{V}$. At each discrete time-step, we define the edges in the system as $(s, r, b_{sr}) \in \mathbf{E}$, where s denotes the index of the "sender" node, r denotes the index of the "receiver" node (i.e., $r : \{1, 2, \ldots, 2C\}$ for all positive integers), and $(b \in \{0, 1\})$ denotes a binary number to inform the graph whether the "sender" node and the "receiver" node belong to the same pairwise neighbors. We emphasize that $b_{sr} = 1$ for "sender" and "receiver" nodes where $s = r$. We use this graph representation and define a sequence of neighbors for each microcrack in the time sequence $\hat{\mathbf{T}}$ as

$$e_{sr}^t = \left(v_s^t, v_r^t, b_{sr}^t\right) \qquad \{t \in \hat{\mathbf{T}}\} \;\; ; \{(s, r, b_{sr}) \in \mathbf{E}\}. \tag{3.2}$$

In equation (3.2), v_s^t denotes the "sender" crack-tip for time t, v_r^t defines the neighboring 'or 'receiver" crack-tip for time t, and b_{sr}^t denotes the binary value defining if v_s^t and v_r^t belong to the same neighborhood for time t.

3.2.3 Formulation of Nearest-neighbors

As shown in Figures 3.2c and 3.2d, we define the zone of influence using radius, $r_c = 750\ mm$, and obtain all microcracks existing within the zone of influence (i.e., shown as white-dashed lines). We note that if a "receiver" crack-tip exists within the zone of influence of a "sender" crack-tip (i.e., $e_{sr} = (v_s, v_r, b_{sr} = 1)$), the remaining crack-tip of the "receiver" node is automatically included as a neighboring node of the "sender" crack tip (i.e., $e_{sr} = (v_s, v_r, b_{sr} = 1)$).

We show this special scenario using red-dashed lines in Figures 3.2c and 3.2dd. Similarly, for crack-tips which have coalesced we share their neighbors with the connecting crack-tips.

3.2.4 Spatial Message-Passing

A key component of GNN models is the introduction of message-passing graph convolutions [167]. Spatial message-passing networks help the GNN model to learn the nodes, edges, and node-edge interactions in the latent space by transferring information between the edges within the local neighborhoods [168]. We implemented three message-passing processes in the Microcrack-GNN framework in order to learn the nodes, edges, and node-edge interactions dynamically.

For the first message-passing process, we update the node interactions defined in equation (3.1) using an MLP encoder. We denote the node MLP encoder as "v-MLP" in Figure 3.3, and μ_v in equation (3.3a). The output from equation (3.3a) denotes the encoded node embedding in the latent space, $\{v_s^{'}\}^{\hat{T}}$. Next, the second message-passing process aims at learning the edge interactions through a second MLP encoder defined as 'e-MLP" in Figure 3.3 and μ_e in equation (3.3b). The output from equation (3.3b) denotes the encoded edge embedding in the latent space, $\{e_{sr}^{'}\}^{\hat{T}}$.

$$\begin{aligned} \{v_s^{'}\}^{\hat{T}} &\longleftarrow \mu_v\left(\hat{P}_s, \hat{N}_s, \hat{O}_s\right) \qquad \{s \in \mathbf{V}\} \\ \{e_{sr}^{'}\}^{\hat{T}} &\longleftarrow \mu_e\left(\{v_s\}^{\hat{T}}, \{v_r\}^{\hat{T}}, \{b_{sr}\}^{\hat{T}}\right) \qquad \{(s,r) \in \mathbf{E}\} \end{aligned} \tag{3.3}$$

Lastly, the third message passing process aims to learn the node-edge-node interactions using the generated latent space embeddings of the nodes and edges. The third message passing process is denoted by "G-MLP" in Figure 3.3 and μ_G in equation (3.4). The inputs to "G-MLP" are defined by concatenating the latent space node embeddings and edge embeddings from equations (3.3a) and (3.3b), respectively. The one-hot encoded feature vector describing the

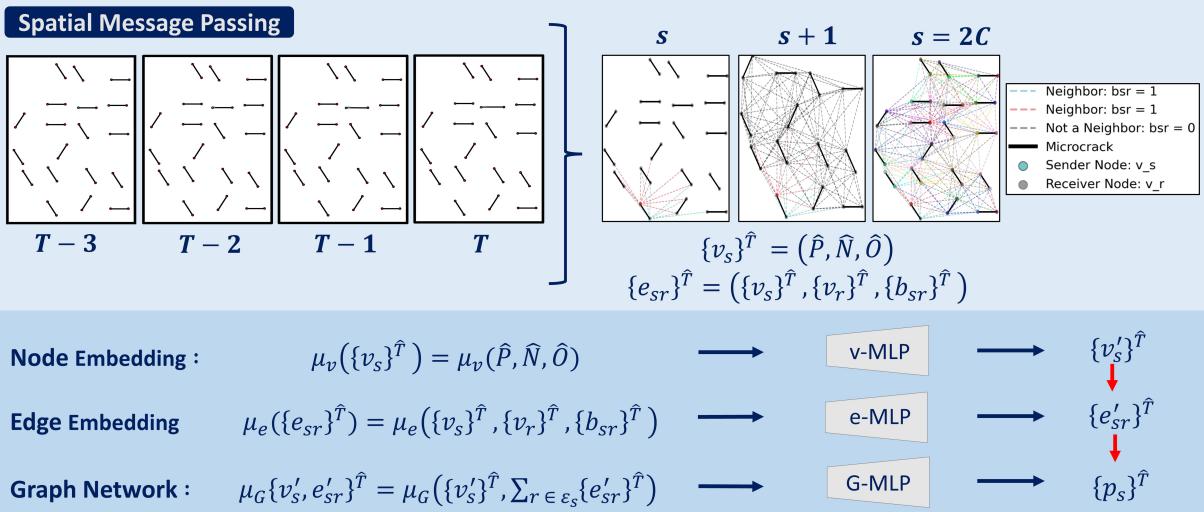


Figure 3.3: Spatial Message-Passing Process: The graph network representation is established and leveraged with two initial encoder MLPs, namely "v-MLP" and "e-MLP", to produce embeddings for the vertices and edges in the latent space. During the message-passing phase, the latent space embeddings of vertices and edges are combined and fed into the message-passing MLP network "G-MLP" to propagate interactions between nodes and edges through a series of update steps, denoted by M. The resulting output is a one-hot encoded feature vector that describes the interaction of the node-edge-node system in the latent space.

resulting latent space node-edge-node interactions, $\{p_s\}^{\hat{T}}$, is obtained as

$$\{p_s\}^{\hat{T}} \longleftarrow \mu_G\left(\{v_s'\}^{\hat{T}}, \sum_{r\in\mathcal{E}_s}\{e_{rs}'\}^{\hat{T}}\right) \qquad \{s \in \mathcal{E}\}. \tag{3.4}$$

Where μ_G defines the third message-passing MLP, $\mathcal{E}$ defines the nearest-neighbors of all microcracks in the system, and $\mathcal{E}_s$ defines the nearest-neighbors of microcrack v_s (i.e., at $b_{sr} = 1$). We note that a number of message-passing steps, M, are then implemented in order to transfer information across the graph's nodes and edges. The number of message-passing steps required for each problem varies depending its complexity, and the depth and size of its local neighborhoods. We follow the same definition from [169, 170, 171] and use $M = 4$ for the initial development of Microcrack-GNN. However, we optimize the model using cross validation in Section 3.5 and obtain the optimal number of message-passing steps as $M = 6$.

3.3 Configuration and set-up of simulations

3.3.1 Training, validation, and testing datasets

We made use of the high-fidelity XFEM mode described in Section 3.2.1 to gather the training, validation, and testing datasets. Following a similar approach as in [88], we defined the domain using $2000\ mm$ by $3000\ mm$ dimension and considered simulations involving up to 19 initial microcracks The material was perfectly brittle, homogeneous and isotropic with $E = 22.6\ GPa$ for the Young's Modulus, $\nu = 0.242$ for the Poisson's ratio, and $K_{crt} = 1.08\ MPa \cdot \sqrt{m}$ for the material toughness. We restrict the problem to quasi-static loading in order to avoid simultaneous propagation of multiple crack-tips as well as crack-tip bifurcation.

The boundary conditions include (i) a fixed amplitude tensile load of $0.01\ m$ (i.e., positive y-direction) at the top edge of the domain, and (ii) fixed bottom edge. We then used the XFEM-based model and developed a function, $GenCrack\_Rand$, for generating unique microcrack configurations involving (i) varying number of microcracks (i.e., defined by the user as $5 \leq C \leq 19$), (ii) randomly generated initial positions, and (iii) randomly generated orientations of 0^o, 60^o, and 120^o without overlapping microcracks. Using the function, $GenCrack\_Rand$, we generate a total of 64 unique configurations for each number of cracks considered, $C : 5 \leq C \leq 19$, and simulate their fracture. The final generated datasets involved a total of $N_{sim} = 64 \times 15 = 960$ unique microcrack simulations. Each simulation included a total of up to $N_{steps} = 46$ time-steps until failure. From [172, 170, 173, 174], we obtained the sequence of previous time-steps as $N_{seq} = 4$ (i.e., $\hat{\mathbf{T}} := \{T-3,\ T-2,\ T-1,\ T\}$)

The purpose of this approach, was to ensure that the framework used the previous four configurations to predict the future time-step. For instance, a single training iteration involved $N_{seq} + 1$ time-steps randomly chosen across each all simulations. The input for each training iteration involved $1, \ldots, N_{seq}$ time-steps, in order to predict the future time-step at $N_{seq}+1$. This approach ensures that the GNN operates on a random time sequence from a random simulation during each training simulation. Thus, learning to predict future time-step configurations given any instance in time for any simulation. However, we did not use random time sequences for

testing the framework. The testing procedure used the initial N_{seq} from the XFEM simulations and then predicted the future time-steps.

Next, we separated the training and validation datasets using 90%, and 10% of the total generated dataset involving $N_{sim} = 960$ simulations, respectively. In essence, the training dataset involved 864 simulations, and the validation dataset involved 96 simulations. Taking into account that each simulation included $N_{steps} = 46$ time-steps, the total number of inputs were calculated using $N_{input} = N_{sim} \times (N_{steps} - N_{seq})$. This resulted in a training dataset of $36,288$ input sequences, and validation dataset of 4032 sequences. We used batch size of 32 for training, and 1 for validation

To generate a testing dataset, we followed the same approach described here and generated 15 simulations for each number of microcracks (i.e., $5 \leq C \leq 19$). Therefore, the testing dataset was comprised of 225 simulations with 50-100 number of time-steps to failure, resulting in up to 22,500 input sequences.

3.3.2 Varying the number of initial microcracks

We emphasize a key feature of the developed framework to simulate microcrack propagation in systems with varying number of microcracks (i.e., $5 \leq C \leq 19$). To design the GNNs capable of predicting varying input-to-output size, we restricted the prediction output to the maximum number of allowed cracks, $C_{max} = 19$, for all cases. This approach involved an extra function which first counted the number of microcracks from the given initial input, and then set the remaining inputs to zero values for cases where $C < C_{max}$ (i.e., padding the output arrays). We emphasize that the GNNs can be easily trained for lower or higher number of initial microcracks in future work.

3.4 Framework architecture

The architecture of the Microcrack-GNN framework involved three initial GNNs: (i) K_I-GNN to predict the Mode-I stress intensity factors, (ii) K_{II}-GNN to predict the Mode-II stress intensity factors, and (iii) Class-GNN to predict the propagating and non-propagating crack-tips. The framework then uses the predictions from the initial GNNs in order to predict the future crack-tip

positions (i.e. predict crack propagation) using the final GNN, CProp-GNN. We show the architecture of the Microcrack-GNN framework in Figure 3.1 and describe each GNN model in the following sections.

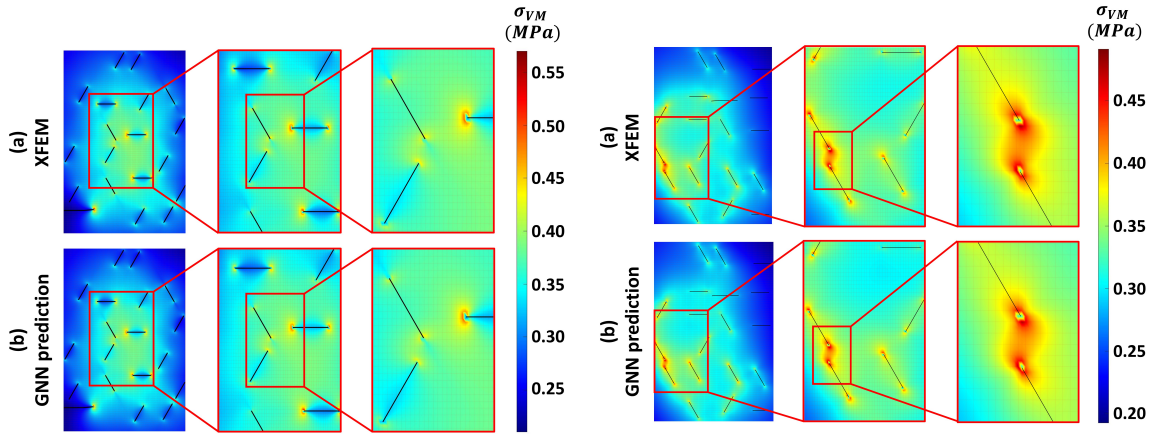
3.4.1 K_I - GNN

The K_I-GNN model was used predict the Mode-I stress intensity factors for each crack-tip in the domain at the future time-step. As shown in Figure 3.1, the first step of the framework was to generate the graph representation input defined in Sections 3.2.2, 3.2.3, and the resulting one-hot encoded feature embedding for the node-edge-node interactions defined in Section 3.2.4. The one-hot encoded feature vector was then input to the K_I-GNN model. This operation for K_I-GNN model is defined as

$$
\begin{aligned}
(\hat{K_I})^{T+1} &\longleftarrow \psi_{K_I}(\hat{P}, \hat{N}, \hat{O}, \hat{K_I}), \\
(\hat{K_I})_s^{T+1} &\longleftarrow \psi_{K_I}\left[\{p_s\}_{t=T-3}^{T} | \hat{N}_s, \hat{O}_s, \{(\hat{K_I})_s\}_{t=T-4}^{T}\right] \qquad \{s \in \mathbf{V}\}.
\end{aligned} \tag{3.5}
$$

The first input sets for K_I-GNN denote the one-hot encoded node-edge-node feature embedding defined in equation (3.4) and its nearest-neighbors in the time sequence $\hat{\mathbf{T}}$. K_I-GNN also includes the initial orientations of the microcracks, $\hat{O}$, as well as the Mode-I stress intensity factors of the crack-tips, $\hat{K_I}$, for the four previous time-steps (i.e., for time sequence $\hat{\mathbf{T}}$). ψ_{K_I} defines the MLP network used for training K_I-GNN. The trained K_I-GNN model then predicted the Mode-I stress intensity factors for each crack-tip at future time-steps (i.e., $(\hat{K_I})_s^{T+1}$).

We note that the predicted stress intensity factors, $\hat{K_I})_s^{T+1}$, were used to generate the stress distribution in the system from Linear Elastic Fracture Mechanics (LEFM) [175]. To compute the stresses, we developed a mesh of 201x201 points throughout the domain and used the principle of superposition from LEFM to obtain the von Mises stress with the predicted Mode-I stress intensity factors. We compare the von Mises stress from the XFEM model versus the von Mises stress from the K_I-GNN model for a simulation from the test dataset in Figure 3.4a.



(a) Predictions of σ_{VM} from K_I-GNN (b) Predictions of σ_{VM} from K_{II}-GNN

Figure 3.4: von Mises stress distributions comparing a) Predicted Mode-I stress intensity factors from the K_I-GNN model and b) Predicted Mode-II stress intensity factors from the K_{II}-GNN model.

3.4.2 K_{II} - GNN

Following a similar approach as K_I-GNN, we define the K_{II}-GNN mode as

$$
\begin{aligned}
(\hat{K}_{II})^{T+1} &\longleftarrow \psi_{K_{II}}(\hat{P}, \hat{N}, \hat{O}, \hat{K}_{II}) \\
(\hat{K}_{II})_s^{T+1} &\longleftarrow \psi_{K_{II}}\left[\{p_s\}_{t=T-3}^{T} | \hat{N}_s, \hat{O}_s, \{(\hat{K}_{II})_s\}_{t=T-4}^{T}\right] \qquad \{s \in \mathbf{V}\}
\end{aligned} \tag{3.6}
$$

Here, $\psi_{K_{II}}$ denotes the MLP network for training the framework on Mode-II stress intensity factors (i.e., $(\hat{K}_{II})_s^{T+1}$). In Figure 3.4b, we show the resulting von Mises stress comparison for the predicted Mode-II stress intensity factors versus the XFEM model.

3.4.3 Classifier - GNN

We note an important assumption of the XFEM model where a single crack-tip is allowed to propagate at each time-step. Therefore, the next GNN model integrated in the framework was Class-GNN for predicting a binary array, $\hat{Q} \in \{0, 1\}$, classifying the propagating crack-tips as $\hat{Q}_s = 1$, and non-propagating crack-tips as $\hat{Q}_s = 0$. By predicting propagating and non-propagating crack-tips the Microcrack-GNN framework aims to capture the quasi-static nature of this problem. Additionally, Class-GNN further simplifies the future time-step predictions for crack propagation by restricting the predictions to a single crack-tip during each time-step.

The first inputs to the Class-GNN model consisted of the binary array of non-propagating and propagating crack-tips during the previous time-steps $(\hat{Q})^T$. Additionally, another key feature of Class-GNN, was that the predicted Mode-I and Mode-II stress intensity factors ($(\hat{K}_I)_s^{T+1}$ and $(\hat{K}_{II})_s^{T+1}$) were used as part of its inputs. In essence, the predicted stress intensity factors allow the model to predict the non-propagating and propagating crack-tips by introducing future information of the stress evolution in the material such as the energy release rate. We define the Class-GNN model as

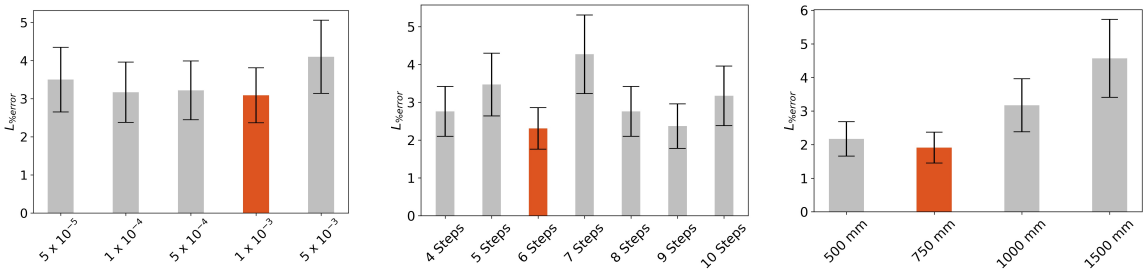
$$
\begin{aligned}
(\hat{Q})^{T+1} &\longleftarrow \psi_{CLASS}\left[(\hat{P},\hat{N},\hat{O}),\hat{Q}^T,(\hat{K}_I,\hat{K}_{II})^{T+1}\right], \\
(\hat{Q})_s^{T+1} &\longleftarrow \psi_{CLASS}\left[\{p_s\}_{t=T-3}^{T}|\hat{N}_s,\hat{O}_s,\{\hat{Q}_s\}^T,\{(\hat{K}_I)_s,(\hat{K}_{II})_s\}^{T+1}\right] \qquad \{s\in\mathbf{V}\}. \quad (3.7)
\end{aligned}
$$

Where ψ_{CLASS} denotes the MLP network used for training the Class-GNN. We note that Class-GNN was integrated in this framework due its quasi-static assumption of the XFEM model. Recently, other works have also developed GNNs to predict of dynamic simulations where all the nodes in the system are allowed to move (i.e., change their position) at each time-step [172, 169, 176, 171][172, 169, 176, 171]. However, this is not the case for quasi-static problems. We emphasize that the Class-GNN model may not be used for cases where multiple crack-tips are allowed to propagate simultaneously at a given time-step. The Class-GNN model may also be extended towards multi-class or multi-label classification [177] for these cases to predict more than one propagating crack-tip

3.4.4 Propagator - GNN

Lastly, the Microcrack-GNN frameworks includes an additional model, CProp-GNN, for predicting the future crack-tip positions. CProp-GNN uses the predicted Mode-I and Mode-II stress intensity factors, and the non-propagating and propagating crack-tips from the previous models as part of its input as shown in equation (3.8).

$$
\begin{aligned}
(\hat{P})^{T+1} &\longleftarrow \psi_{CProp}\left[(\hat{P},\hat{N},\hat{O}),(\hat{K}_I,\hat{K}_{II},\hat{Q})^{T+1}\right] \\
(\hat{P})_s^{T+1} &\longleftarrow \psi_{CProp}\left[\{p_s\}_{t=T-3}^{T}|\hat{N}_s,\hat{O}_s,\{(\hat{K}_I)_s,(\hat{K}_{II})_s,\hat{Q}_s,\}^{T+1}\right] \qquad \{s\in\mathbf{V}\} \quad (3.8)
\end{aligned}
$$



(a) Learning rates (b) Message-passing steps (c) Zone of influence radii

Figure 3.5: Cross-validation results for: a) Learning rates 5×10^{-5}, 1×10^{-4}, 5×10^{-4}, and 5×10^{-3} are shown in gray, with our model's learning rate 1×10^{-3} highlighted in red. b) Message-passing steps of 4, 5, 7, 8, 9, and 10 are shown in gray, with our model's message-passing steps of 6 highlighted in red. c) Zone of influence radii 500mm, 1000mm, and 1500mm are shown in gray, with our model's zone of influence radius of 750mm highlighted in red.

3.5 Cross-Validation of Microcrack-GNN

We implemented cross-validation [178] to the Microcrack-GNN framework for additional tuning. We tuned the framework in terms of the learning rate, message-passing steps, and zone of influence size. Following the *k*-fold cross-validation methodology, for each training parameter, we trained the framework for 5 epochs. We randomly selected 12 cases from the validation dataset for each training sequence (i.e., every 5 epochs). After training was completed, we computed the average of the maximum percent errors for the predicted length of the cracks. Figure 3.5 shows the obtained errors for the learning rates, message-passing steps, and zones of influence.

For the learning rates, as shown in Figure 3.5a we evaluated values of 5×10^{-5}, 1×10^{-4}, 5×10^{-4}, 1×10^{-3}, and 5×10^{-3}. While the smallest learning rate of 1×10^{-4} resulted in error of $3.50 \pm 0.85\%$, the lowest error was obtained for 1×10^{-3} (shown in red) of $3.09 \pm 0.72\%$ Therefore, for further training of the Microcrack-GNN framework we used 1×10^{-3} for the learning rate.

The errors for the message-passing steps are shown in Figure 3.5b. We evaluated message-passing values of 4-10. We recall from Section 3.2.4 that for initial training of the Microcrack-GNN framework a value of $M = 4$ was used. However, Figure 3.5b shows that the lowest error was obtained for $M = 6$ (shown in red) of $2.31 \pm 0.55\%$ compared to $M = 4$ of $2.76 \pm 0.66\%$. Therefore, we used $M = 6$ for further training of the Microcrack-GNN framework. We note

from [174, 179] that higher number of message-passing steps increases training time, simulation time, and GPU requirements. We also note that the difference in errors for $M = 4$ and $M = 6$ was not significant.

Lastly, as shown in Figure 3.5c we evaluated the zone of influence radii of 500mm, 1000mm, 750mm, and 1500mm. While one may think that increasing the size of local neighborhoods provides additional information for the node and edge interactions to increase its accuracy, this is not the case for this problem. For instance, the highest error in the zones of influence was obtained for radius of 1500mm at $4.57 \pm 1.16\%$. Here, larger connectivity radii may result in unnecessary connections (i.e., long edges) between crack-tips which far-away from each other. These interactions between far-away crack-tips may cause oversampling in high-resolution regions of the domain, where the nearest neighbors have higher influence in propagating microcracks compared to far-away neighbors [180]. From Figure 3.5c, the optimal zone of influence was obtained for radius of 750mm at $1.91 \pm 0.46\%$ compared to the smallest zone of influence of 500mm with $2.17 \pm 0.51\%$ error. Therefore, we chose 750mm to further optmize training of the framework.

3.6 Results

The testing dataset for Microcrack-GNN (described in Section 3.3.1) involved 225 simulations consisting of 15 simulations for each number of varying initial microcracks. We use the test dataset to evaluate the prediction accuracy of Microcrack-GNN to simulate microcrack propagation and coalescence. For each testing simulation, we then compute the errors for (i) the predicted microcracks length as a function of time, (ii) the final predicted cracks path, (iii) the resulting effective stress intensity factors, K_{eff}, and (iv) the generated von Mises stresses. We then compare the performance of Microcrack-GNN against two additional baseline models. Lastly, we compare the computational costs of Microcrack-GNN versus XFEM using their required simulation time.

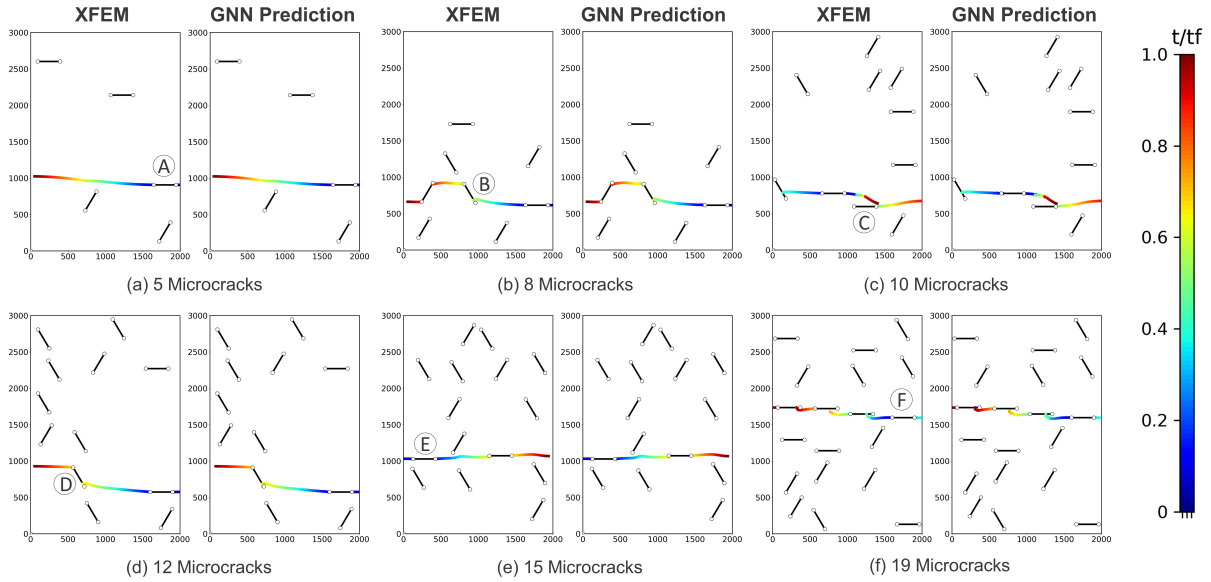


Figure 3.6: XFEM simulations compared to GNN predictions of crack propagation and coalescence for cases involving: a) 5 b) 8 c) 10 d) 12 e) 15, and (f) 19 microcracks. The crack paths are colored based on t/t_f, where t_f represents the final simulation time for a given case.

3.6.1 Prediction of microcracks propagation and coalescence

For this analysis, we randomly selected 6 cases from the test dataset for varying number of initial microcracks of $C = 5, 8, 10, 12, 15,$ and 19. We then generated the crack evolution for each case using Microcrack-GNN and compare the predicted microcracks paths versus the XFEM model as shown in Figure 3.6. From Figure 3.6a, ($C = 5$) it can be seen that the predicted fracture path of a single microcrack was nearly indistinguishable from the XFEM model. Figure 3.6d and Figure 3.6e depict the resulting microcracks paths for $C = 12$ and $C = 15$, respectively. For both cases, two microcracks can be seen to propagate and coalesce around 60% of final material failure. Similar to 3.6a, for $C = 12$ and $C = 15$ the predicted microcracks paths are nearly identical to the XFEM predictions. These cases show that framework was able to predict crack propagation for higher-complexity cases involving 12-15 microcracks with good accuracy.

Two marginally more complex crack paths were obtained for $C = 8$ and $C = 10$ shown in Figures 3.6b and 3.6c, respectively. For both cases, three microcracks propagated and coalesced in order to reach complete material failure. The case involving $C = 8$ showed high prediction accuracy (i.e., qualitatively) for the crack paths compared to XFEM. However, for $C = 10$ shown in Figure 3.6c we can see some prediction error for the microcracks paths. Here, two cracks

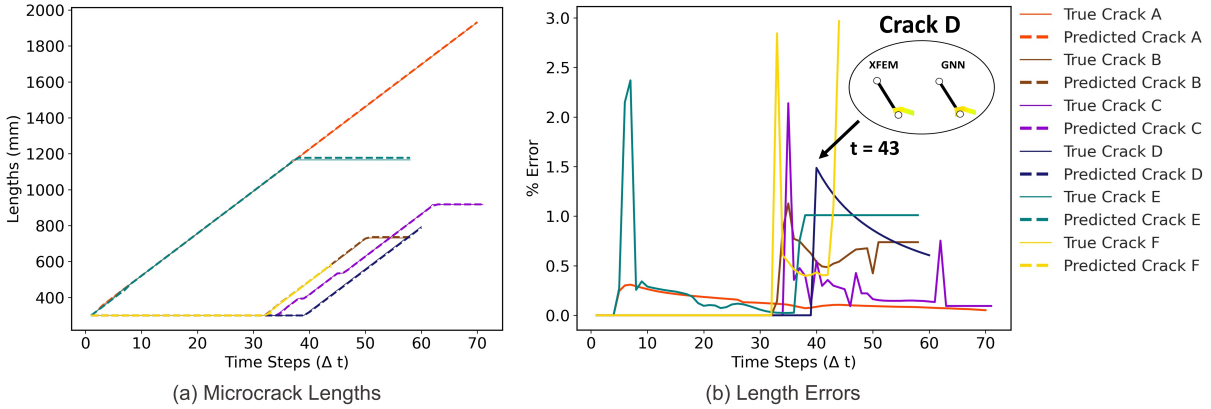


Figure 3.7: a) Evolution of crack length versus time for high-fidelity simulations and Microcrack-GNN for cracks $A, B, \ldots, F$ labeled in Figure 3.6. b) Crack lengths relative error between Microcrack-GNN and XFEM.

(i.e., left-most microcrack and middle microcrack) connected close to 40% of total material failure. At this time-step (i.e., $t = 0.4t_f$), a switch in the propagating crack-tip can be observed. This switch of propagating crack-tips caused the new propagating crack-tip to move towards the opposite direction from the actual path, thus, resulting in an overlapping microcrack.

Additionally, Figure 3.6f presents a more interesting fracture scenario for $C = 19$ Here, four microcracks propagated in the material to reach complete fracture. We denote the right-most microcrack (i.e., first microcrack to propagate) as "F" in Figure 3.6f. The simulation then follows a sequential propagation from microcrack "F" in the left direction. Although this case involves the highest amount of initial microcracks (i.e., higher complexity), the predicted fracture paths are relatively close to the XFEM simulation with few small deviations. For instance, a slight kink in the predicted path can be seen when the microcrack "F" connects with the second propagating crack (around $t = 0.4t_f$). A similar kink is also observed for the second microcrack during coalescence with the third propagating microcrack (around $0.9t_f$). While these kinks caused roughness in fracture path at these time-steps, the final predicted path of the Microcrack-GNN framework is virtually indistinguishable to the XFEM simulation. Considering these small errors, the framework was capable of simulating microcrack propagation and coalescence with good prediction accuracy for cases involving varying number of initial microcracks.

3.6.2 Microcrack length growth

Subsequently, we analyze the evolution of crack lengths over time to further validate the Microcrack-GNN's capability to accurately predict microcrack propagation and coalescence. The analysis involves calculating the growth of microcrack lengths with respect to time, as illustrated in Figure 3.7. In each simulation scenario, we utilize a propagating microcrack to monitor length variations, as indicated in Figure 3.6 by labels A, B, C, D, E, and F. For instance, microcrack A exhibits a linear increase in length, starting at the initial length of 300mm and extending to approximately 1900mm. Comparing the predicted microcracks paths and lengths obtained from XFEM and Microcrack-GNN (as depicted in Figures 3.6 and 3.7), we observe close alignment with the ground truth. Moreover, in cases involving $C = 8$ and $C = 10$ with three microcracks coalescing, the length prediction maintains high accuracy with slight intermediate errors during the coalescence events. Figure 3.7 further illustrates the predicted lengths of microcracks B and C to overlap for both XFEM and Microcrack-GNN, throughout the entire simulation duration.

For the case of $C = 12$ involving crack D, the crack length remains fixed at the initial measurement of $300\ mm$ for the majority of the simulation, until around time-step 43. As crack D intersects with the already propagating right-most microcrack during this time-step, we observe a linear increase in length over time, coupled with a sudden spike of around 1.5% in relative error. To analyze the origin of this relative error spike, we provide a close-up view of the time frame when Crack D (shown in Figure 3.7b) converges with the pre-existing propagating microcrack for both the XFEM and GNN models. Although the XFEM model smoothly connects both cracks, the GNN model shows a downward jump upon coalescence, resulting in an additional spike in error for crack D. Comparing the predicted crack length by Microcrack-GNN to XFEM, we observe that Microcrack-GNN predicts a slightly longer final crack length than XFEM. The relative error for the predicted crack length is approximately 0.55% at $t = t_f$). Similarly, for the $C = 15$ case of crack E, the length increases linearly until reaching approximately $1200\ mm$ by time-step 38. Like crack D, the Microcrack-GNN's predicted a slightly longer final length for crack E compared to XFEM (showing relative error

of 0.9% at $t = t_f$). These findings obtained for cases of $C = 12$ and $C = 15$ suggest that Microcrack-GNN may predict slightly faster crack growth, i.e., faster simulation, compared to XFEM.

Next, analyzing the more intricate scenario of $C = 19$ in in Figure 3.7, the right-most crack (F) is seen to propagate first. Figure 3.7 demonstrates a linear progression in crack length for crack F. Throughout this phase, the Microcrack-GNN's crack length closely mirrors XFEM, maintaining near-identical values until approximately time-step 13. During this time-step, a slight deviation is observed between the predicted crack length and the XFEM crack length. As previously noted, upon comparing the length increase of crack F with the crack paths depicted in Figure 3.6f, it becomes evident that the minor variance in predicted length is associated with deviations in crack paths at the junctures of crack coalescence. These results imply that the framework's accuracy for predicting crack paths and lengths may not be contingent upon the initial quantity of microcracks within the system. Consequently, suggesting that discrepancies in the predictions and the resulting small errors could be influenced by the initial configuration of the system instead (e.g., the initial positions and orientations of the cracks).

3.6.3 Errors in final crack path

We recall from Section 3.6.2 the time deviance error analysis performed for the predicted crack lengths. From this analysis shown in Figure 3.6, although the Microcrack-GNN framework may predict either faster or slower crack propagation in certain instances, the final cracks path were almost identical to XFEM. Nonetheless, . Within this section, we additionally conduct a spatial deviation analysis to characterize the errors in the final predicted crack paths. For this analysis, the maximum percent error in distance between the XFEM and Microcrack-GNN crack paths was computed.

In this section, to evaluate the errors in the final predicted crack paths, we conduct a spatial deviation analysis. For this purpose, we calculate the maximum percent error in distance between the crack paths from the Microcrack-GNN and XFEM models. In Figure 3.8, we illustrate the computed errors for crack path from the entire test dataset consisting of 225 cases. It is worth noting that in comparison to the time-wise errors of crack length errors (shown in Figure 3.9),

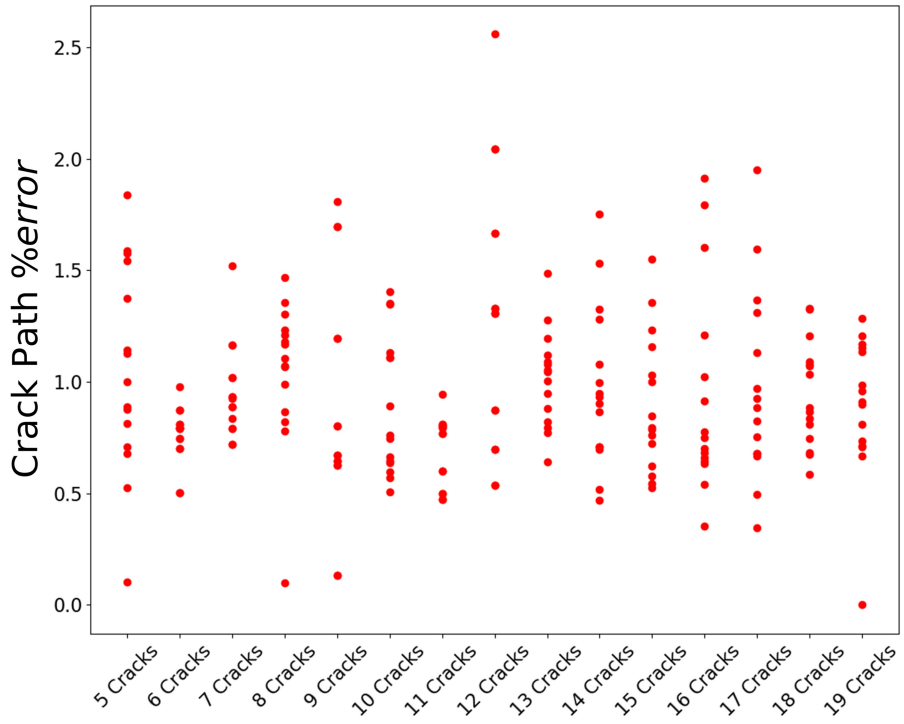


Figure 3.8: The maximum percentage errors for crack path predictions across all test cases (225 test cases).

the errors for the final predicted fracture paths are notably lower. The maximum error of 2.53% was observed for 12 microcracks, whereas the minimum error of 0.004% was obtained for 19 microcracks. Consequently, although the time deviation errors for the predicted crack lengths exhibited higher errors, Figure 3.8 demonstrates that the framework was capable of predicting crack paths with accuracies lower than 2.53%.

3.6.4 Crack length errors of entire test dataset

For every test simulation, across each C value, and at every time-step, we determined the maximum percent error of the predicted crack lengths. Subsequently, we calculated the average error across time for each test simulation to gather 15 error points pertaining to each C value The resulting average errors are shown in Figure 3.9. Notably, we observe the highest error in predicted length registering at approximately 5.85% for test case 11 involving 8 microcracks.

We examine Figures 3.10 and 3.11 in order to investigate the root cause of this error. In Figure 3.10 we provide a qualitative comparison for the predicted propagation of cracks A and B during time-steps 1 and 22. Additionally, we show the XFEM and Microcrack-GNN generated

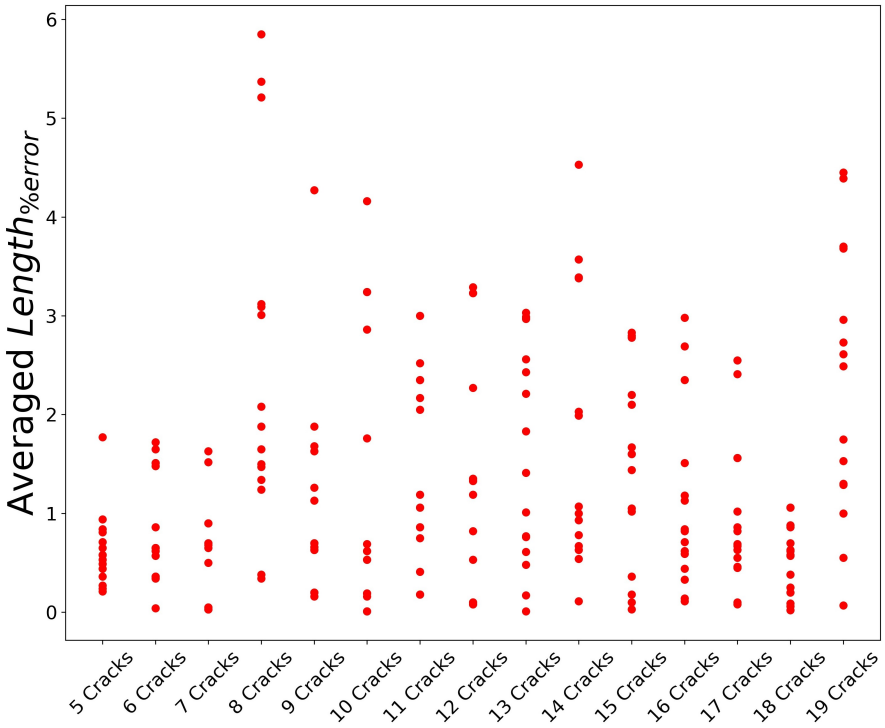
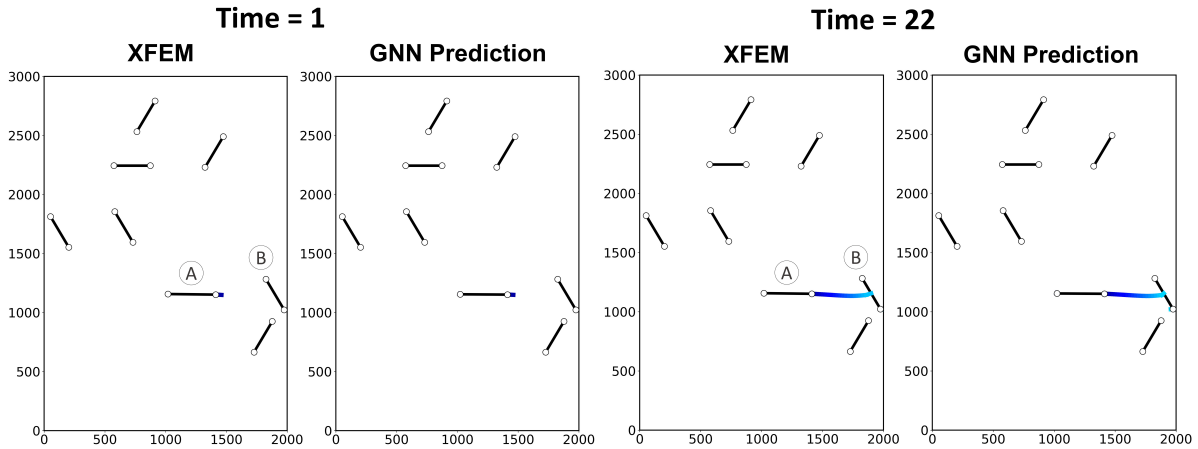


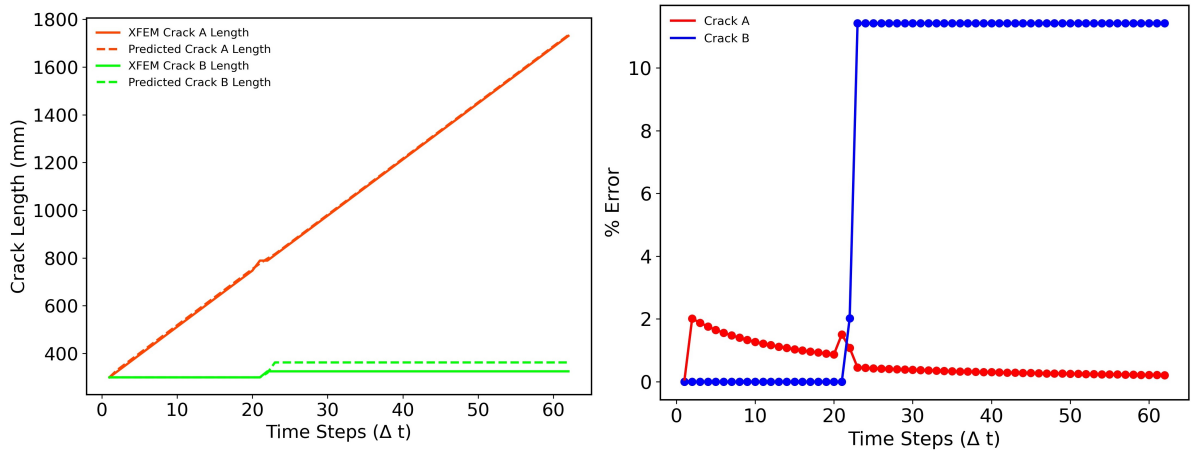
Figure 3.9: The averaged maximum percentage errors of crack length predictions across all test cases (225 test cases).



(a) Time-step 1 of 8 cracks

(b) Time-step 22 of 8 cracks

Figure 3.10: Crack propagation for test case 11 involving 8 microcracks, during a) Time-step = 1, and b) Time-step = 22



(a) Microcrack Length growth

(b) Microcrack Length Error

Figure 3.11: a) Evolution of crack length versus time for the high-fidelity XFEM model and Microcrack-GNN, for test case 11 involving 8 microcracks. b) Crack lengths relative error between the high-fidelity XFEM model and Microcrack-GNN, for test case 11 involving 8 microcracks

crack lengths for cracks A and B as a function of time, along with their errors in Figures 3.11a and 3.11b. Upon inspection of 3.11a, we observe an initial error spike of approximately 2.1% at time-step 1. From Figure 3.10a, this substantial error surge stems from the slightly divergent propagation direction of predicted crack A towards the negative y-direction, situated to the right of the domain. However, the propagation of crack A for the XFEM model is directed to the right following a straight path. Consequently, as also illustrated in Figure 3.11a, this results in a slightly larger predicted crack length than the XFEM crack length.

Furthermore, as illustrated in Figures 3.11a and 3.10, crack B shows the most significant error spike of approximately 10.8% during time-step 22. Examining Figures 3.10b and 3.11a, it is apparent that by time-step 22, crack A has already undergone propagation in previous time-steps. During this time frame, the right tip of crack B propagates in the right direction, eventually reaching the material's right. Observing Figure 3.10b, it's evident that at time-step 22, Microcrack-GNN predicts crack propagation in the negative x-direction (left) while the XFEM crack propagation direction aligns towards the positive x-direction (right) This erroneous predicted crack path leads to the highest percent error of 5.85% observed in the test dataset due to a larger predicted crack size in the subsequent time-steps.

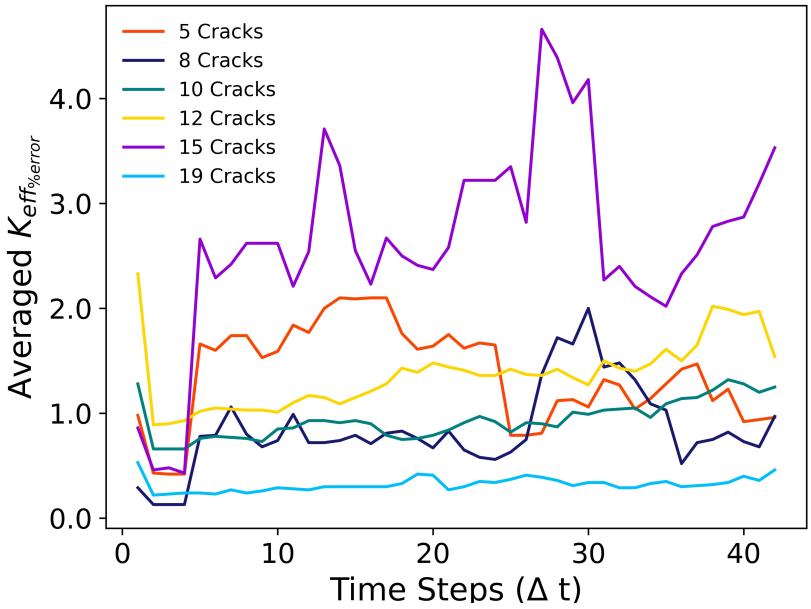


Figure 3.12: Averaged maximum percentage errors for the effective stress intensity factors for cases involving 5, 8, 10, 12, 15, and 19 microcracks over time.

3.6.5 Effective stress intensity factor errors

We examine six cases with varying numbers of initial microcracks (i.e., $C = 5, 8, 10, 12$, and 19), in order to quantify the predicted stress intensity factors errors. The first step was to use the XFEM and Microcrack-GNN generated Mode-I and Mode-II stress intensity factors to compute the effective stress intensity factors as

$$K_{eff} = \sqrt{(K_I)^2 + (K_{II})^2}. \tag{3.9}$$

To compute the errors, we focus on crack tips where $K_{eff} \geq K_{crt}$, as this criteria determines the cracks which are most likely to propagate for any given time-step. For each time-step of each simulation in the test dataset, we then compute the effective stress intensity factors error as

$$K_{eff_{\%error}} = \max_{s \in N^t_{crt}} \left(\frac{\left| {K_{eff}}^t{}_{Pred} - {K_{eff}}^t{}_{True} \right|}{{K_{eff}}^t{}_{True}} \right)_s \times 100 \qquad \{t = 1, 2, \ldots, T_f\}, \tag{3.10}$$

Here, N_{crt}^{t} denotes the number of cracks-tips where $K_{eff} \geq K_{crt}$, ${K_{eff}}_{Pred}^{t}$ denotes the predicted K_{eff} at time t using Microcrack-GNN, and ${K_{eff}}_{True}^{t}$ denotes the true K_{eff} at time t generated by XFEM.

We show the resulting average of $K_{eff_{\%error}}$ with respect to time for cases with $C = 5, 8, 10, 12, 15,$ and 19 in Figure 3.12. From this figure, it is evident that the highest errors across all cases considered were observed for 15 microcracks. For the case involving 15 microcracks, we see the maximum percent error peaking at approximately 4.80%. In contrast, we observe the lowest error of approximately 0.20% for the case with 19 microcracks. Interestingly, the resulting errors did not show an increasing trend with the number of initial microcracks. For example, in configurations with 5, 8, 10, 12, and 15 initial microcracks, the percent errors were higher than the errors obtained for the 19 microcracks case. These outcomes imply that the initial positions and orientations of the neighboring microcracks in the system have a higher influence on the resulting errors compared to the number of initial microcracks.

3.6.6 Errors of effective stress intensity factor for entire test dataset

Continuing with the methodology outlined in Section 3.6.5, we calculated the average of the maximum percent errors on effective stress intensity factors across time, for each simulation in the test dataset. In Figure 3.13, we show the resulting average errors for each simulation (i.e., 10 error points for each C value) As shown in this figure, the highest error in effective stress intensity factors across all simulations of approximately 3.48% was obtained for the scenario involving 6 microcracks.

We examine Figures 3.14 and 3.15 in order to identify the source the error in stress intensity factor in the configuration involving of 6 microcracks. First, in Figure 3.14 we depict the evolution of the stress intensity factor error for Case No.1 in the dataset of 6 microcracks. Initially, the errors remain under 2% during the initial time-steps in the simulation. However, we see an increasing trend in error around time-step 15, peaking at approximately 6.75% at $t = 26$. In order to further understand this error increase we generate the von Mises stress distributions from the XFEM model and Microcrack-GNN framework for $t = 26$ of this simulation. The resulting XFEM versus Microcrack-GNN von Mises stresses are shown in Figure 3.15. We

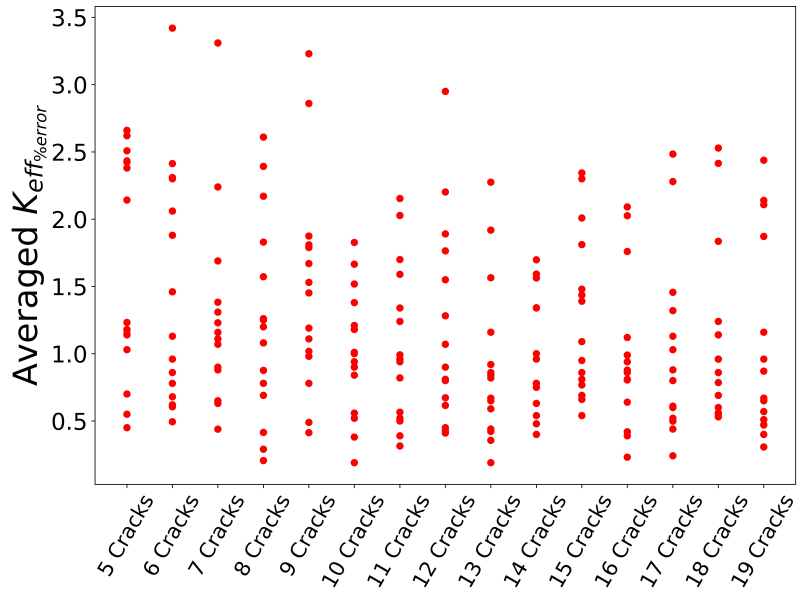


Figure 3.13: Averaged maximum percentage errors of effective stress intensity factors across all test cases (225 in total).

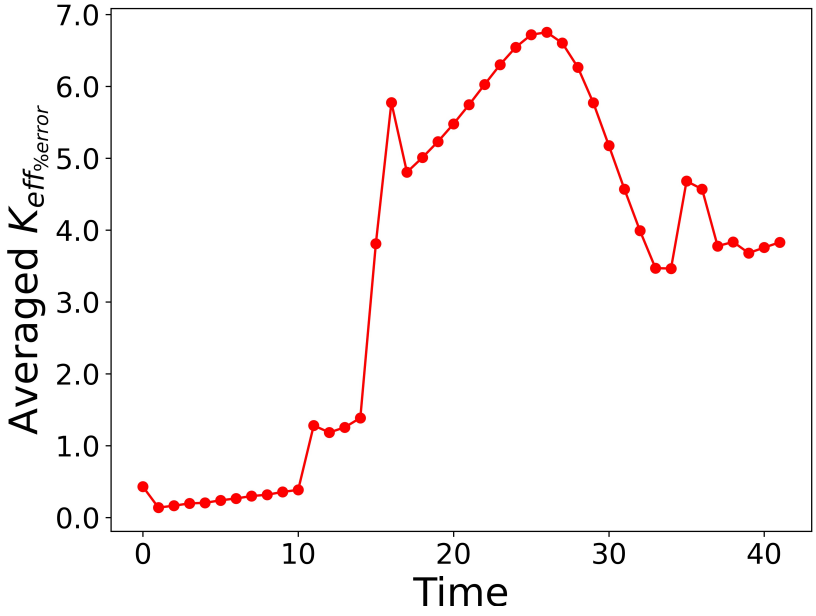


Figure 3.14: Maximum percentage error in effective stress intensity factor plotted against time for test case 1 involving 6 microcracks.

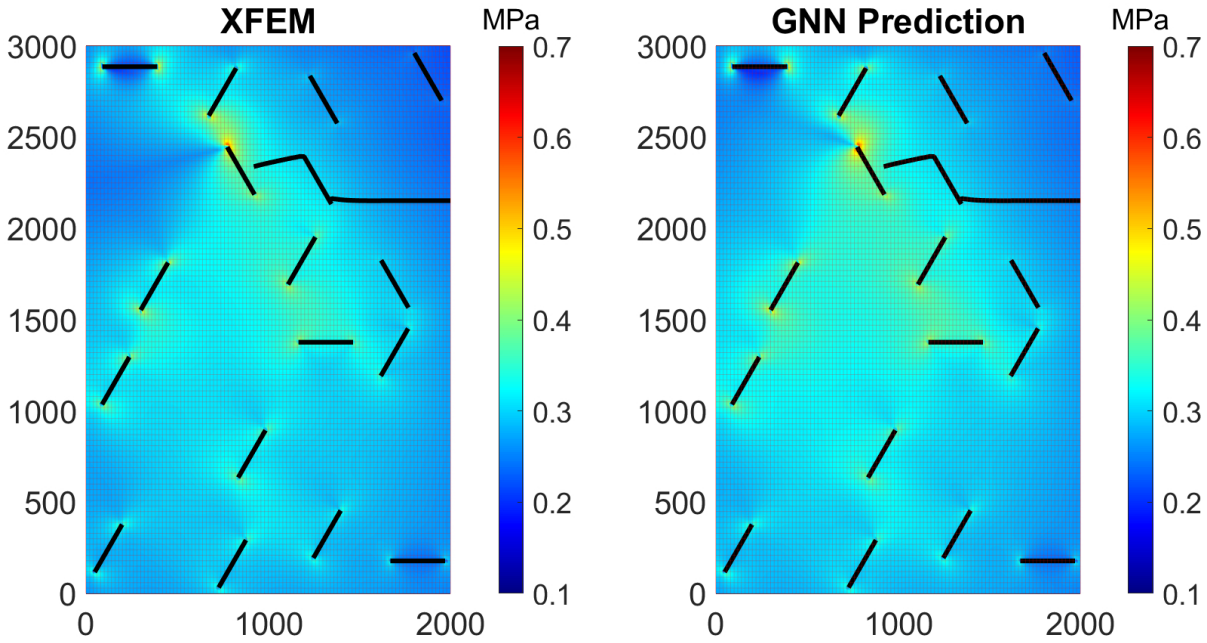


Figure 3.15: Comparison of von Mises stress distribution between XFEM (left) and Microcrack-GNN (right) for test case 1 involving 6 microcracks at time-step 26.

emphasize that although the maximum stress intensity factor error of 6.75% is seen at this time frame, the Microcrack-GNN displays a stress distribution highly resembling that of the XFEM model.

3.6.7 von Mises Stress distribution error

A key feature of Microcrack-GNN is its capability to simulate the stress evolution over time. The predicted Mode-I and Mode-II from K_I-GNN and K_{II}-GNN can be directly used to to compute the stress distribution within the domain. We employed the following approach to illustrate the resulting errors in the von Mises stress prediction for the test dataset. Firstly, we obtained the absolute errors (in MPa) at every point in the domain for each time-step as $|\sigma_{VM_{Pred}} - \sigma_{VM_{True}}|$. Subsequently, as depicted in equation (3.11), we utilized the maximum von Mises stress as a reference value for error percentage (i.e., $\sigma_{VM_{Max}}$).

$$\sigma_{VM\%error} = \max_{(i,j)\in\mathbb{R}^2} \left(\frac{|\sigma_{VMPred} - \sigma_{VMTrue}|}{\sigma_{VMMax}} \right) \times 100 \tag{3.11}$$

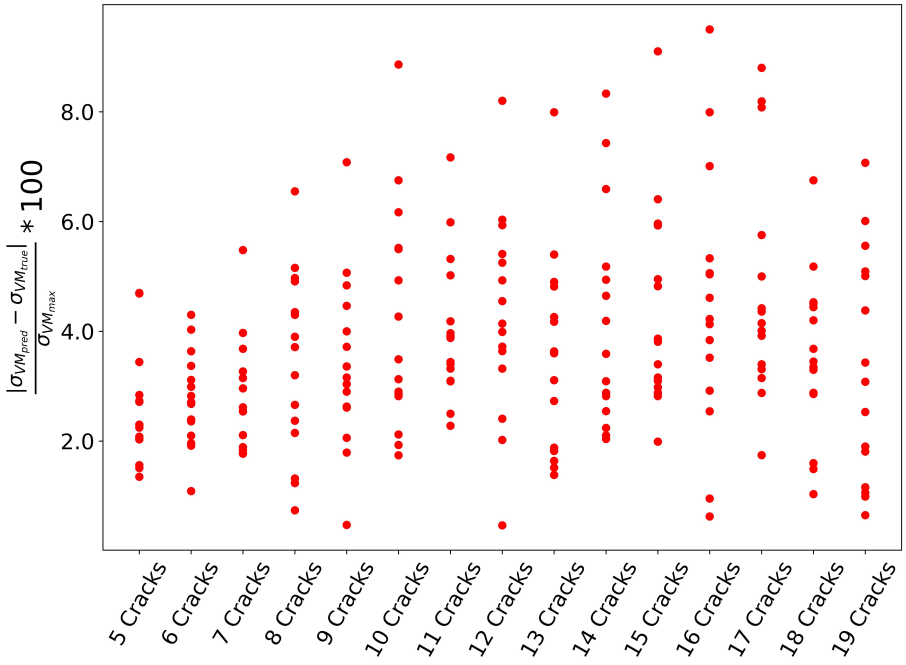


Figure 3.16: Maximum percentage errors of von Mises stresses for all 225 test cases.

We employ this method and derive the maximum von Mises stress error at every time-step of each simulation within the test dataset. For each test simulation, we extract the resulting maximum error across time as shown in Figure 3.16.

Figure 3.16 indicates that in cases involving 10, 12, 14, 15, 16, and 17 initial microcracks, the framework yielded errors exceeding 8%. It can be seen that Case No. 6 exhibited the highest percent error for von Mises stress (10.29%) across the entire dataset. In Figure 3.17, we examine showcase the maximum von Mises stress percentage error as a function of time for Case No. 6 of 16 microcracks, in order to understand the root causes of these high errors.

Initially, it is evident that the error remained below 3% throughout the simulation. However, the error sharply escalated around time-step 47, peaking at 10.29% during time-step 49. At this time-step, we compare the XFEM and Microcrack-GNN von Mises stress distribution in Figure 3.18, and present the resulting absolute error between the two. The origin of error can be attributed to the stress intensity factors of the coalescing cracks. Consequently, a limitation of Microcrack-GNN framework is the high error observed when multiple cracks coalesce. As the von Mises stresses are calculated based on the superimposition of the Mode-I and Mode-II stress intensity factors predictions, the error accumulates. A possible avenue for future work to mitigate this challenge is to directly predict the von Mises stress throughout the mesh.

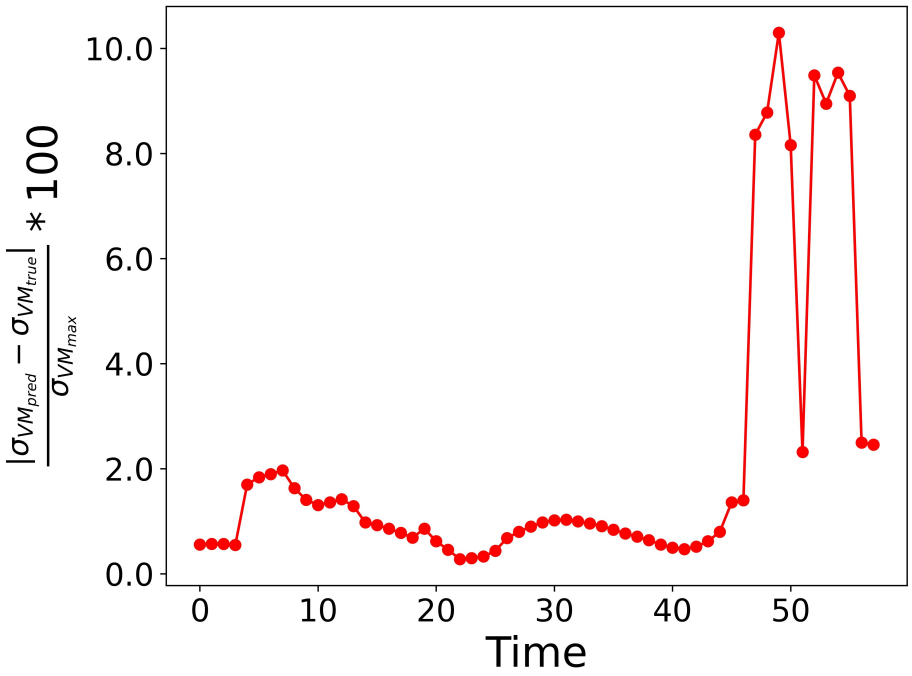


Figure 3.17: Maximum percentage error of von Mises stress versus time for test case 6 involving 16 microcracks.

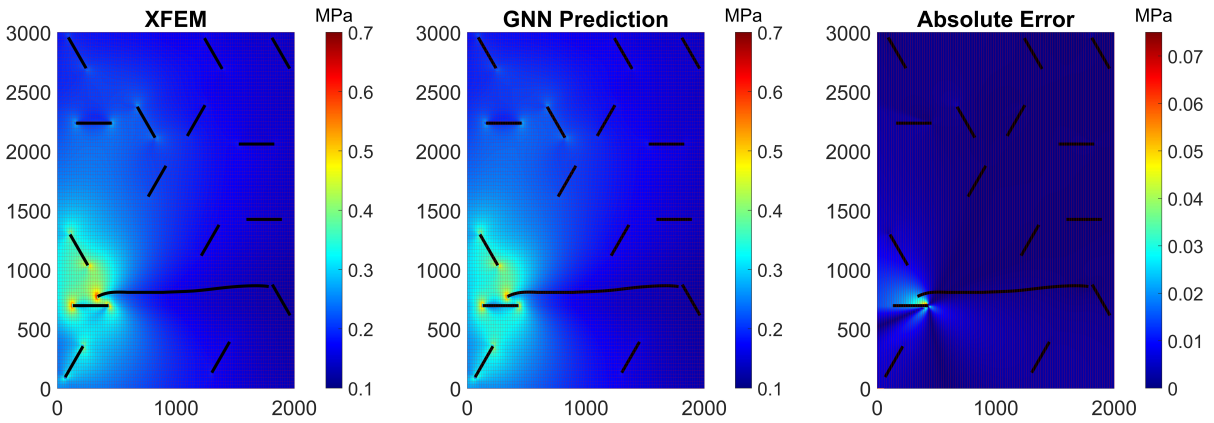


Figure 3.18: Comparison of von Mises stress distributions for test case 6 involving 16 microcracks: XFEM prediction (left), Microcrack-GNN prediction (center), and resulting absolute error (MPa) (right).

3.6.8 Additional Baselines

In this section, we devised two additional ML models in order to compare their performance against the developed Microcrack-GNN framework The training process involved 5 epochs. We utilized two distinct loss functions: Mean Squared Error (MSE) and Mean Absolute Error (L1 loss).

1. RCNN: The initial baseline model entailed a Recurrent Convolutional Neural Network (RCNN) comprising two identity convolution layers, two batch normalization layers, followed by the ReLU activation function, and output of Linear layer. The input used for RCNN involved a $(4 \times 38 \times 6)$ matrix, where (i) 4 denotes the number channels representing the time sequence $\hat{\mathbf{T}}$, (ii) 38 represents the number of nodes (i.e., crack-tips), and (iii) 6 signifies the number of features, including crack orientation, x and y positions, propagating vs non-propagating crack-tips, and K_I and K_{II} stress intensity factors, for the time sequence $\hat{\mathbf{T}}$.

2. REDNN: The second ML was a Recurrent Encoder-Decoder Neural Network (REDNN) featuring four convolution layers, one feed-forward layer, and four transpose convolutions, with the ReLU activation function applied to each layer. The input format for the REDNN remained consistent with that of the RCNN, utilizing a $(4 \times 38 \times 6)$ matrix.

We then evaluated the performance of each baseline model against the Microcrack-GNN. As illustrated in Figure 3.7, we computed the error in the predicted effective stress intensity factor, K_{eff}, and the predicted crack length for the scenario involving 12 microcracks. Table 1 shows the resulting errors. It is evident that the RCNN model surpassed the REDNN model in predicting Mode-I and Mode-II stress intensity factors. Notably, the RCNN with the L1 loss function yielded a lower error of 12.71% compared the RCNN with the MSE loss function, which reached error of 13.71%. Conversely, the REDNN models exhibited superior performance over the RCNN models in predicting crack length. The REDNN model with the MSE loss function achieved the lowest error of 16.03%, compared to 17.09% for the REDNN model with the L1 loss function. However, Microcrack-GNN significantly outperformed both RCNN and

REDNN across both prediction cases (i.e., K_{eff} and crack length), with the lowest percent errors of 1.85% and 0.32%, respectively. These findings highlight the efficacy of the developed GNN compared to other prevalent ML baselines.

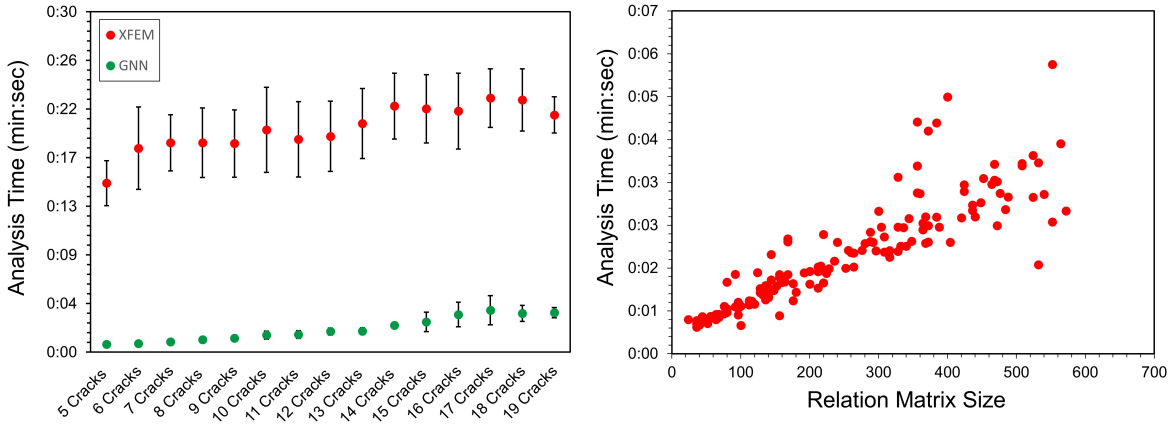
Table 1: Performance comparison of baseline models versus Microcrack-GNN

Models	$K_{eff}\%Error$	Length $\%$ Error
RCNN (L1)	12.71%	62.47%
RCNN (MSE)	13.71%	41.74%
REDNN (L1)	39.66%	17.09%
REDNN (MSE)	35.94%	16.03%
Microcrack-GNN	**1.85 %**	**0.32 %**

3.6.9 Simulation time versus number of initial microcracks

In the subsequent analysis, we evaluate the simulation time of the Microcrack-GNN framework in comparison to XFEM for varying numbers of initial microcracks. As shown in Figure 3.19a, we assess the average CPU time per time-step across different values of C, to gauge the framework's potential for accelerating simulation time against XFEM. We used an Intel(R) Core(TM) i3-10100 CPU @ 3.60GHz with 16GB RAM. For each varying number of microcracks from the test-set, a total of 10 simulations were performed. Upon examining Figure 3.19a, it becomes evident that the high-fidelity XFEM model demanded considerably longer simulation times than Microcrack-GNN. Specifically, the development of the GNN framework resulted in a speed-up ranging from 6x to 25x compared to XFEM.

Additionally, the simulation time of both XFEM and Microcrack-GNN increases for higher number of initial microcracks. This increase in computational time for higher number of microcracks may be due to the expanding size of the relation matrix. The relation matrix size, also known as adjacency matrix, is a binary tensor defined by the number of edges in the graph. Thus, for higher number of initial microcracks the number of edges increases, resulting in a larger relation matrix. This relation is illustrated for Microcrack-GNN in Figure 3.19b, which shows the required simulation time (in min:sec) versus size of the relation matrix.



(a) Required CPU-time

(b) Relation matrix size versus CPU-time

Figure 3.19: Average CPU time (min:sec) per simulation time frame compared across different scenarios: a) Varying number of initial microcracks (5 to 19). b) Different sizes of the relation matrix.

Lastly, Figure 3.19a highlights a significant simulation time improvement for the Microcrack-GNN framework (i.e., up to 25x faster) to simulate higher-complexity fracture problems. As a potential avenue for future work, the Microcrack-GNN framework could be extended to simulate a broader range of initial microcracks while maintaining similar time performance to the cases depicted in Figure 3.19a.

3.7 Conclusion

In conclusion, the integration of neural networks and graph theory for the development GNN simulators presents a promising avenue for enhancing the computational efficiency of existing high-fidelity fracture mechanics models. Specifically, the development of such models for simulating the evolution of stresses and crack propagation in brittle materials involving varying number of microcracks presents a novel contribution in the field. In this chapter, we devised four GNNs (shown in Figure 3.1) tailored to model the underlying physics governing these complex problems.

We established a dynamic framework capable of accurately predicting future crack-tip positions and microcrack coalescence events, as well as stress intensity factors and stress distributions in the domain as a function of time. The K_I-GNN and K_{II}-GNN models demonstrated high accuracy in predicting K_{eff}, with a maximum relative error of 4.80%. The framework

also exhibited good accuracy in predicting crack lengths with maximum errors of 1.01% and 4.29%. Notably, the Microcrack-GNN framework offers the flexibility to simulate fracture for varying numbers of microcracks ranging from 5 to 19 without additional modifications to the GNN models.

While this versatility underscores the adaptability and robustness of the proposed approach in addressing diverse fracture mechanics scenarios, it is important to acknowledge several limitations in the Microcrack-GNN framework. One significant limitation is the computation of von Mises stresses throughout the domain. The von Mises stresses are derived using the LEFM superposition principle on the predicted Mode-I and Mode-II stress intensity factors. This approach can lead to high errors exceeding 8% in certain cases. Additionally, the framework is not currently optimized, resulting in lengthy training times of approximately 5.18 hours using four NVidia T4 GPUs. This extended training duration can hinder productivity and scalability. We also implemented our own in-house GNN using PyTorch instead of leveraging existing libraries such as [181, 182]. Optimization of the GNN back-end libraries could include enhancements in performance through flexible GPU resource allocation, improved spatial message-passing techniques, and more efficient parameterizations. Furthermore, there is potential for optimizing the overall network architecture to reduce the required training time and preserve prediction accuracy [2, 36, 183, 56]. These optimizations could significantly enhance the efficiency of the framework. We also emphasize that the high-fidelity XFEM model utilized in this study was not parallelized to work using multiple CPUs. Exploring parallel computing strategies for XFEM could lead to performance improvements in future iterations. Addressing these limitations and implementing optimizations are crucial aspects of future work to enhance the overall efficiency and effectiveness of the Microcrack-GNN framework to become a fast and accurate simulator capable of handling large numbers of cracks. Ultimately, by leveraging dynamic graphs in future works there the framework's applicability can be extended towards ductile materials to capture crack nucleation phenomena, thereby enhancing its versatility and utility in various engineering applications.

Chapter 4

Transfer learning and graph neural networks towards generalized machine learning framework to simulate brittle crack problems

4.1 Introduction and motivation

Although ML methods like Microcrack-GNN hold promise to simulate complex dynamic problems, their accuracy may suffer when dealing with new problem-specific inputs (such as domain size or loading type) which are unknown to the training dataset. This limitation necessitates generating new, extensive training datasets and subsequent retraining for each new problem-specific input, which results in a computationally expensive approach. A potential solution to mitigate the challenges posed by large datasets is the implementation of Transfer Learning (TL) methods [184]. TL methods enable the transfer of learned information about the underlying physics and patterns of the problem across new problem-specific inputs. One common approach to transfer the learned information involves leveraging pre-trained weights from an initial ML model [185, 186]. TL has proven effective in various domains, with one notable example being object detection/recognition tasks using Convolutional Neural Networks (CNNs) [187, 188, 189, 190, 191, 192]. In such tasks, initial layers typically capture basic shapes like round edges, and vertical and horizontal lines, while deeper layers capture more problem-specific characteristics. For example, researchers have utilized TL strategies in the past to object detection models for identifying defects in various materials such as CFRP [193], composite structures [194], and graphene [195]. These approaches demonstrate the effectiveness of transfer learning in leveraging previously learned knowledge to enhance model performance in new problem domains.

TL has also demonstrated favorable results for predicting material properties from molecular dynamic simulations [196], predicting structure-property relationships from reconstructed microstructures [197], capturing hierarchical microstructure representations [198], and material design [199]. Additionally, TL techniques have been used for GNNs. For GNNs, the embeddings from the initial stages learn the global characteristics (or physics), while the embeddings in the final stages capture more problem-specific characteristics [200, 201]. For instance, Lee *et al.* [202] implemented TL using the pre-trained weights from the Crystal Graph Convolutional Neural Network (CGCNN) (ie., introduced in [203]) to predict properties in new crystal structures with limited training datasets. A similar approach was also employed in [204] using TL of the pre-trained CGCNN model to forecast methane adsorption in metal-organic frameworks.

In this study, we apply TL techniques to enhance the prediction capabilities of crack propagation, coalescence, and stress evolution in brittle materials. In Chapter 3, we introduced Microcrack-GNN, a framework capable of simulating these phenomena using GNNs under tensile loading conditions. Although Microcrack-GNN demonstrated high accuracy and significant speedup compared to an XFEM model, it also exhibited certain limitations [205]. Specifically, the framework was trained with 864 simulations tailored to cases involving only tensile loads, neglecting shear-loading scenarios. Moreover, it considered fixed domain sizes ($2000mm \times 3000mm$), constant crack length ($300mm$), and three crack orientations (0^o, 60^o, 120^o). The framework's accuracy was not validated for shear loading or different domain sizes, and the graph representation lacked information to accommodate varying loading types or domain sizes.

By leveraging TL, we introduce a versatile GNN framework termed ACCelerated Universal fRAcTure Emulator (ACCURATE), depicted in Figure 4.1. ACCURATE is designed to predict crack propagation and stress evolution across a range of fracture mechanics scenarios. Our approach involves a series of 5 TL update steps incorporating novel problem-specific inputs. During each update step, weights from the preceding TL steps are loaded and retrained. This iterative process aims to adapt the model to new problem parameters while retaining previously learned features.

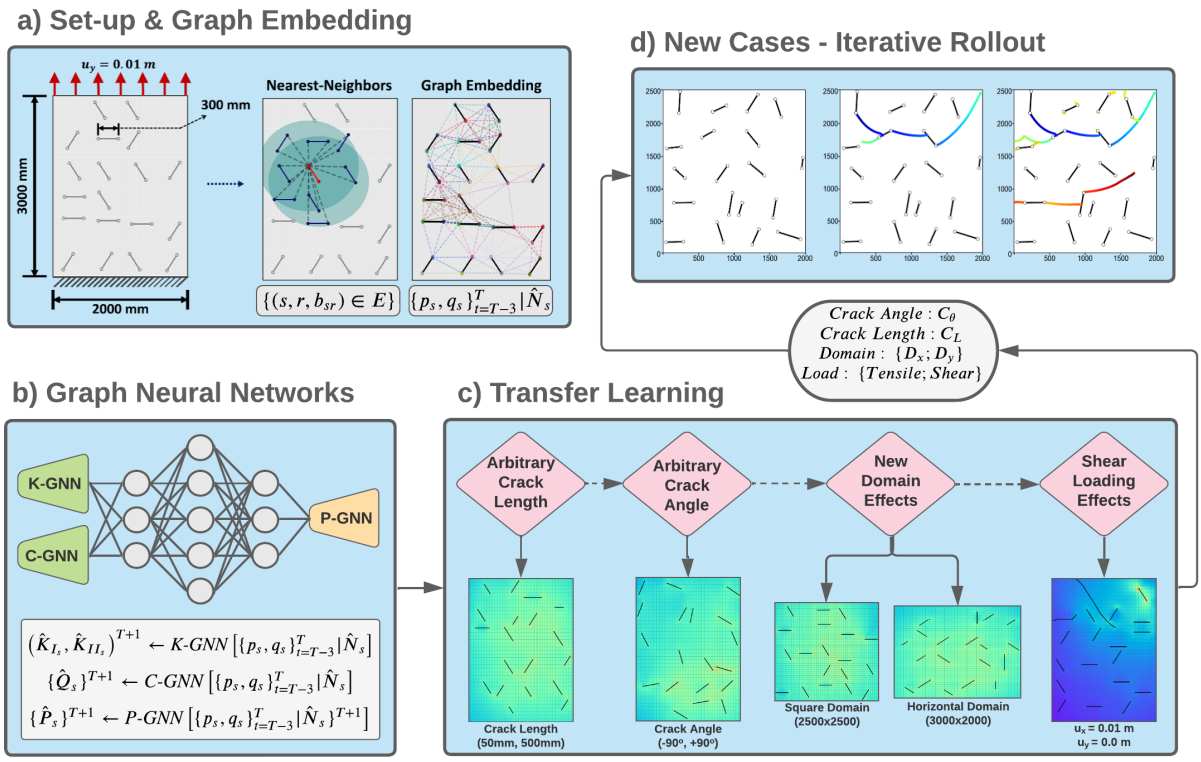


Figure 4.1: Flowchart depicting the structure of the ACCURATE framework: a) The initial setup of the problem involves formulating the nearest-neighbor approach and embedding the resulting topological graph (TL graph). b) Architecture of the GNN models, including the K-intensity-factor-GNN (*K-GNN*), Classifier-GNN (*C-GNN*), and Propagator-GNN (*P-GNN*). c) Sequence of TL applications, including: (i) arbitrary crack length, (ii) arbitrary crack angle, (iii) new domain dimensions (square and horizontal), and (iv) shear loading effects. d) Illustration of the iterative rollout process for unseen problem configurations.

As shown in Figure 4.1c, the TL update steps incorporate new problem-specific inputs including: (i) arbitrary crack length, (ii) arbitrary crack orientation, (iii) additional domain effects (square and horizontal), and (iv) shear loading effects. We show the effectiveness of TL methods of requiring significantly smaller training datasets to retain high prediction accuracy, thereby reducing training time. The final trained ACCURATE framework is proficient in predicting unseen cases with high accuracy involving new arbitrary crack angles, crack lengths, and boundary dimensions under shear or tensile loads, avoiding the need for additional TL. The new framework's generalized graph representation offers a flexible approach that can be easily adapted for addressing new problem-specific inputs in future endeavors. Lastly, ACCURATE demonstrates significantly accelerated simulation times, providing a 200x speedup compared to the high-fidelity XFEM surrogate model.

4.2 Methods

4.2.1 XFEM-based surrogate model and TL simulation set-up

We utilize the XFEM model outlined in Section 3.2.1 to create new datasets for training, validation, and testing, facilitating the incorporation of TL. This XFEM model adeptly simulates the two-dimensional fracture of brittle materials with multiple microcracks defined by arbitrary positions and orientations under tensile and shear loads. To explore the potential of TL, we craft six fresh case studies and progressively apply TL update steps as follows.

- Case 1: Vertical domain ($2000mm \times 3000mm$) with fixed crack lengths ($300mm$) and crack orientations (0^o, 60^o, and 120^o) in Mode-I loading.
- Case 2: Vertical domain ($2000mm \times 3000mm$) with arbitrary crack lengths (from $50mm$ to $500mm$), and fixed crack orientations (0^o, 60^o, and 120^o) in Mode-I loading.
- Case 3: Vertical domain ($2000mm \times 3000mm$) with fixed crack lengths ($300mm$), and arbitrary crack orientations (from -90^o to $+90^o$) in Mode-I loading.
- Case 4: Square domain of $2500mm \times 2500mm$ with fixed crack lengths ($300mm$) and crack orientations (0^o, 60^o, and 120^o) in Mode-I loading.
- Case 5: Horizontal domain of $3000mm \times 2000mm$ with fixed crack lengths ($300mm$) and crack orientations (0^o, 60^o, and 120^o) in Mode-I loading.
- Case 6: Shear loading case in a vertical domain ($2000mm \times 3000mm$) with fixed crack lengths ($300mm$) and crack orientations (0^o, 60^o, and 120^o), with the fixed bottom edge and constant displacement of $0.01\ m$ at the top edge towards the positive x-direction (shear load).

Using this methodology, we compiled a dataset consisting of 35 simulations, each extending up to 101 time-steps, for every case study from Case 1 to Case 6 above. We note that some simulations terminated before reaching the maximum of 101 time-steps due to the rapid propagation and coalescence of cracks within the domain. Despite this variability, we

recall that the Microcrack-GNN exhibited good prediction accuracy in predicting fracture events across different scenarios with varying numbers of cracks ranging from 5 to 19. However, it's important to acknowledge a noticeable trend: as the number of initial microcracks increased, so did the computational demands on the model, resulting in longer simulation times. To ensure a comprehensive coverage, we generated our training, validation, and test datasets for the most computationally intensive scenarios involving 19 microcracks. For each case study (i.e., Case 1 to Case 6), we allocated 20 simulations for training, 5 simulations for validation, and 10 simulations for testing purposes.

In line with the approach adopted for Microcrack-GNN, each training data point in our TL framework comprises a sequence spanning $N_{seq}+1$ time-steps, selected randomly from all training simulations. Here, the first $1,\ldots,N_{seq}$ time-steps serve as inputs, while the prediction at $(N_{seq}+1)$th time-step forms the output. Consistent with Microcrack-GNN, we employ a sequence length of $N_{seq}=4$, denoted as $\hat{\mathbf{T}} := \{T-3,\ T-2,\ T-1,\ T\}$. For each dataset, we calculate the total number of inputs as $N_{input} = N_{sim} \times (N_{steps} - N_{seq})$, yielding 1,940, 485, and 970 inputs for the training, validation, and test datasets, respectively. It's worth highlighting that the Microcrack-GNN framework utilized a significantly larger dataset for training, comprising a total of 36,288 inputs, which is approximately 20 times larger than the datasets employed for TL in ACCURATE.

4.2.2 Graph representation and spatial message-passing

The first step to develop the ACCURATE framework was to define a new novel graph representation incorporating information regarding the boundary dimensions and load types. We show the graph representation for ACCURATE in Figure 4.1a. The graph representation, $\langle \mathbf{V}, \mathbf{E} \rangle$, is similar to that of the Microcrack-GNN, with key modifications and additions for the node and edge features. We define the node as ξ_s, and include four new node features to describing: (i) load type $\hat{\mathcal{F}}_s = (u_{x_s}, u_{y_s})$ where tension is represented by $\{u_x = 0.0, u_y = 0.01m\}$ and shear by $\{u_x = 0.01m, u_y = 0.0\}$), and (ii) the effects on domain height and width, specifically the horizontal and vertical distances to the right and top edges of the domain denoted as $\hat{\mathcal{B}}_s = (d_{W_s}, d_{H_s})$. For the edges, represented by $\mathcal{E}_{sr}$, additional spatial features and physics-informed features have

been introduced. The spatial features comprise (i) the vertical distance $\Delta\hat{\mathcal{Y}}_{sr} = (y_r - y_s)$, (ii) the horizontal distance $\Delta\hat{\mathcal{X}}_{sr} = (x_r - x_s)$, (iii) the equivalent distance $\hat{\mathcal{L}}_{sr} = \sqrt{\Delta\hat{\mathcal{X}}_{sr}^2 + \Delta\hat{\mathcal{Y}}_{sr}^2}$, and (iv) the relative crack orientation $\Delta\hat{O}_{sr} = (\theta_r - \theta_s)$ between the crack-tips. Furthermore, the physics-informed features $\Delta\hat{\Pi}_{sr}$, encompass (i) the difference of Mode-I stress intensity factors $\Delta\hat{K}_{I_{sr}}$, (ii) the difference of Mode-II stress intensity factors $\Delta\hat{K}_{II_{sr}}$, and (iii) the difference of effective stress intensity factors $\Delta\hat{K}_{eff_{sr}} = \left(\sqrt{\Delta\hat{K}_{I_{sr}}^2 + \Delta\hat{K}_{II_{sr}}^2}\right)$ for each edge in the system. We formulate the graph representation for the node and edge features as

$$\begin{aligned} \xi_s^t &= \left(\hat{P}_s^t, \hat{N}_s^t, \hat{O}_s^t, \hat{K}_{I_s}^t, \hat{K}_{II_s}^t, \hat{\mathcal{F}}_s^{\,t}, \hat{\mathcal{B}}_s^t\right) \qquad \{t \in \hat{\mathbf{T}}\} \;; \{s \in \mathbf{V}\}, \\ \mathcal{E}_{sr}^t &= \left(\Delta\hat{\mathcal{X}}_{sr}^t, \Delta\hat{\mathcal{Y}}_{sr}^t, \Delta\hat{\mathcal{L}}_{sr}^t, \Delta\hat{O}_{sr}^t, \Delta\hat{\Pi}_{sr}^t\right) \qquad \{t \in \hat{\mathbf{T}}\} \;; \{(s, r, b_{sr}) \in \mathbf{E}\}. \end{aligned} \tag{4.1}$$

As depicted in Figure 4.1a, to realize the message-passing mechanism within the ACCURATE framework we employ the Graph Isomorphism Network with Edges (GINE) model featuring aggregated weights. The GINE model operates by taking inputs consisting of the node features, edge connectivity arrays, and edge features, subsequently producing a new one-hot encoded feature vector characterizing the latent space interactions [206]. We represent the message-passing model, GINE, as

$$\{q_s\}^{\hat{T}} \longleftarrow GINE\left(\{\xi_s\}^{\hat{T}}, \{e_{sr}\}^{\hat{T}}, \{\mathcal{E}_{sr}\}^{\hat{T}}\right) \qquad \{s \in \mathbf{V}\}. \tag{4.2}$$

To implement TL to ACCURATE, we leverage the pre-trained one-hot encoded feature embedding from Microcrack-GNN (previously defined as p_s in equation (3.4)). By transferring the pre-trained weights from μ_G in equation (3.4), we implement TL to the one-hot encoded feature from the ACCURATE framework (outlined in equation (4.2)). This strategy establishes a generalized GNN structure, enabling the incorporation of new node features and edge features across different scenarios.

4.2.3 K-GNN, C-GNN, and P-GNN

To develop the GNNs for predicting stress intensity factors and crack propagation, we used an initial MLP for predicting two outputs, followed by a second MLP with a sole prediction output. We illustrate both MLPs in Figure 4.1b. We define the initial MLP as K-intensity-factor-GNN (K-GNN) in Figure 4.1b. K-GNN served as a regression MLP to predict the crack-tips Mode-I and Mode-II stress intensity factors at the next time-step. From Figure 4.1b, we represent the second MLP as Classifier-GNN (C-GNN). C-GNN operated as a classifier MLP to predict the propagating and non-propagating crack-tips, $\hat{Q}_s$, at the next time-step. As described in Section 4.2.2, we implemented TL by transferring the pre-trained weights of μ_G from equation (3.4) to both K-GNN and C-GNN models. In other words, we used the one-hot encoded feature embedding from Microcrack-GNN (equation (3.4) in Section 3.2.4) as the first input, and the one-hot encoded feature embedding from ACCURATE (equation (4.2) in Section 4.2.2) as the second input to the K-GNN and C-GNN models. The resulting initial input graphs are described in equation (4.3) as

$$\begin{aligned}\left(\{\hat{K}_{I_s}\},\{\hat{K}_{II_s}\}\right)^{T+1} &\longleftarrow \textit{K-GNN}\left[\{p_s,q_s\}_{t=T-3}^{T}|\hat{N}_s\right],\\ \{\hat{Q}_s\}^{T+1} &\longleftarrow \textit{C-GNN}\left[\{p_s,q_s\}_{t=T-3}^{T}|\hat{N}_s\right] \qquad \{s\in\mathbf{V}\}.\end{aligned} \tag{4.3}$$

Additionally, the outputs from K-GNN and C-GNN serve as inputs to the final MLP network shown in Figure 4.1c. We define this MLP in Figure 4.1c as Propagate-GNN (P-GNN). P-GNN functions as a regression MLP with a single prediction output consisting of the future x- and y-coordinate crack-tip positions. As part of the input to P-GNN, we include the outputs from K-GNN (i.e., stress intensity factors, $\hat{K}I^{T+1}$ and $\hat{K}II^{T+1}$), and the output from C-GNN (i.e., propagating and non-propagating crack-tips, $\hat{Q}^{T+1}$). Given that crack propagation in quasi-static fracture is driven by crack-tips above the critical stress intensity factor, the outputs from K-GNN and C-GNN assist the final MLP to predict crack propagation in subsequent time-steps. Therefore, we define the P-GNN model as:

$$\{\hat{P}_s\}^{T+1} \longleftarrow \textit{P-GNN}\left[\{p_s,q_s\}_{t=T-3}^{T}|\hat{N}_s,\{\hat{K}_{I_s},\hat{K}_{II_s},\hat{Q}_s\}^{T+1}\right] \qquad \{s\in\mathbf{V}\}. \tag{4.4}$$

4.2.4 Order of transfer learning application

We illustrate the TL sequence in Figure 4.1c. Here, we first extracted the pre-trained embeddings (p_s and q_s) for the initial case study. The initial case studied involved scenarios with tensile loading in a fixed size vertical domain with cracks of constant length and orientation. We then transferred these pre-trained embeddings across the remaining case studies described in Section 4.2.1. We started by utilizing the graph embeddings for cases with arbitrary crack lengths. Subsequently, we transferred the new trained graph embeddings to address cases involving arbitrary crack orientations. We further followed this systematic approach and adapted ACCURATE towards new domain sizes, including square and horizontal domains, respectively. As depicted in Figure 4.1c, we considered new loading scenarios, such as shear loading for the final TL implementation. Ultimately, the final graph embeddings successfully simulated fracture in new domain configurations with arbitrary crack lengths and orientations under tension and shear loads In the following sections, we provide a comprehensive error analysis for each case study.

4.3 Results

In the subsequent sections, we conduct a thorough analysis of the errors encountered in each case study outlined in Section 4.2.1. We begin by assessing the framework's accuracy concerning the number of training samples utilized for TL. Following this, we delve into evaluating ACCURATE's performance in emulating crack propagation and stress evolution across each case study involved in the TL steps. We provide in-depth error analyses regarding the predictions of crack paths and effective stress intensity factors. Furthermore, we showcase ACCURATE's proficiency in simulating the evolution of crack propagation and stresses for unseen scenarios. For the unseen scenarios, we consider new domain sizes with arbitrary crack lengths and orientations, under both tension and shear loads. Lastly, we compare the computational times required for XFEM versus ACCURATE simulations.

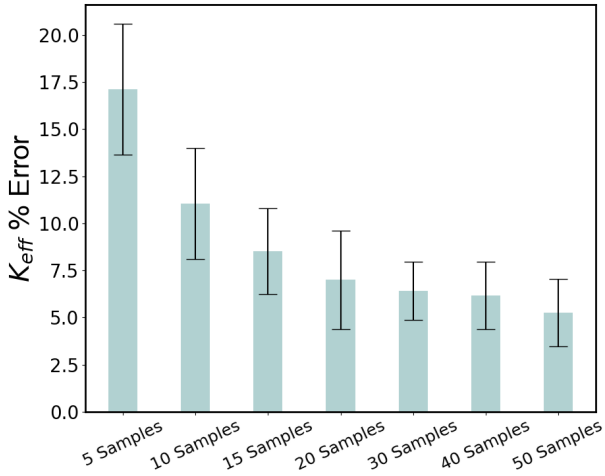


Figure 4.2: Graph showing the relationship between the number of TL training samples and the error in stress intensity factor for shear loading simulations involving 5, 10, 15, 20, 30, 40, and 50 training instances.

4.3.1 Framework's error versus number of TL training samples

In this study, we utilized 20 training samples for each TL update step. We note that Microcrack-GNN required significantly larger training dataset, approximately 50$\times$ larger, in order to achieve good accuracy. To investigate the impact of the number of TL training samples on the model's accuracy, we conducted a detailed analysis. Initially, we generated a total of 50 training samples for the shear load case study. Subsequently, we divided these samples into 7 random groups with 5, 10, 15, 20, 30, 40, and 50 TL training samples, respectively. Each group underwent TL over 10 training epochs. Following the training process, we assessed the performance of each model using 5 simulations from the test dataset, ensuring consistency in the error analysis. The choice of shear loading for this analysis stemmed from its higher transfer space complexity compared to the remaining case studies (e.g., tensile load, square domain, horizontal domain, arbitrary crack orientation, and arbitrary crack length).

Figure 4.2 illustrates the percent errors in the stress intensity factor obtained for each group. These errors were first computed with respect to time, followed by calculating the average and standard deviation across time. As anticipated, the highest error was observed when employing only 5 TL simulations at $20.10 \pm 3.33\%$, whereas the lowest error of $5.27 \pm 1.78\%$ occurred for 50 TL simulations. The graph clearly depicts a decreasing trend in error with the increasing number of TL simulations. Consequently, further augmenting the number of TL samples is

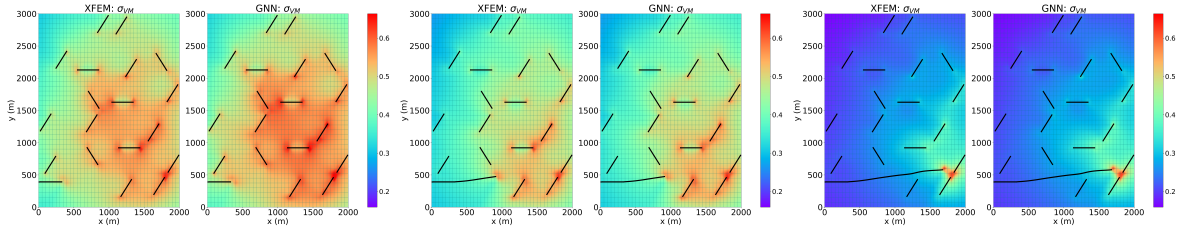
likely to yield a minor reduction in error. We also note that larger training datasets entail longer training times and increased computational costs. Therefore, we utilized 20 simulations for each TL update step to obtain satisfactory prediction accuracy while minimizing computational costs and training time. Lastly, as described in Section 4.2.1, we recall that a simulation can include up to 101 time steps, resulting in a TL dataset of up to discrete 1,940 inputs.

4.3.2 Mode-I and Mode-II stress intensity factors prediction

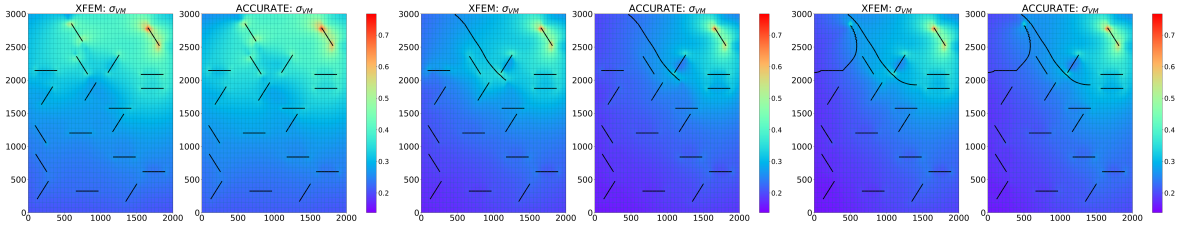
We conduct a qualitative analysis to assess the framework's capability in emulating stress evolution using TL. Stress computation is facilitated through the LEFM equations and the stress intensity factors (Mode-I and Mode-II). Figure 4.3 displays the evolution of the von Mises stress spanning from $t = 1\%$ to $t = 90\%$ for two loading scenarios: (i) vertical domain under tension as shown by Figures 4.3a-4.3c, and (ii) shear loading shown by Figures 4.3d-4.3f). In case (i) involving tension, the Mode-I stress intensity factors become pivotal to determine the propagating crack-tips and their propagation direction. Conversely, for case (ii) involving shear, the Mode-II stress intensity factors have a greater influence on the propagation of crack-tips and their crack path direction. This distinction is evident in Figure 4.3, where cracks under tension tend to propagate horizontally, while those under shear exhibit a diagonal propagation tendency.

Additionally, cracks initially oriented at 0^o exhibit a higher propensity for propagation for case (i) subjected to tension. In contrast, for case (ii) subjected to shear the crack-tips with diagonal orientations induce greater stress interactions among neighboring the cracks. A qualitative comparison between the XFEM model and ACCURATE reveals nearly identical time evolutions for both models. Consequently, ACCURATE demonstrates capability to predict stress evolution accurately for both cases (i) - (ii).

Applying a similar methodology as with the varied load cases, we evaluated the performance of ACCURATE across different domain sizes. We recall from Section 4.2.1 that two distinct domain configurations were considered involving (i) square domains measuring $2500mm \times 2500mm$, and (ii) horizontal domains measuring $3000mm \times 2000mm$. For these configurations, we maintained a consistent crack length of $300mm$ and as well as fixed crack orientations including 0^o, 60^o, and 120^o. By incorporating TL to accommodate diverse domain

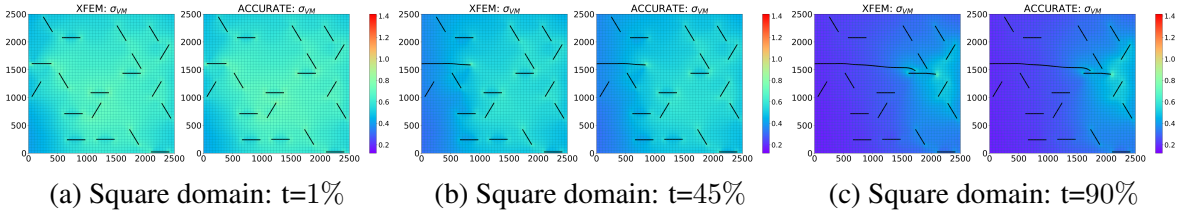


(a) Tensile load: t=1% (b) Tensile load: t=45% (c) Tensile load: t=90%

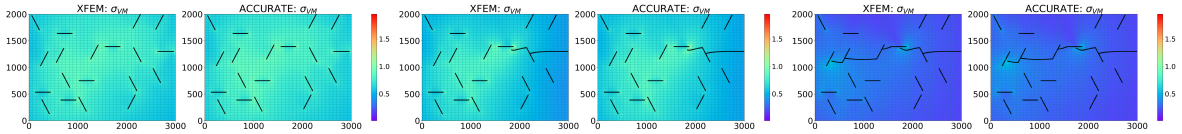


(d) Shear load: t=1% (e) Shear load: t=45% (f) Shear load: t=90%

Figure 4.3: Evolution of von Mises stress (MPa) from $t = 1\%$ to $t = 90\%$ for (a-c) test cases under tensile loading, and (d-f) test cases under shear loading.



(a) Square domain: t=1% (b) Square domain: t=45% (c) Square domain: t=90%

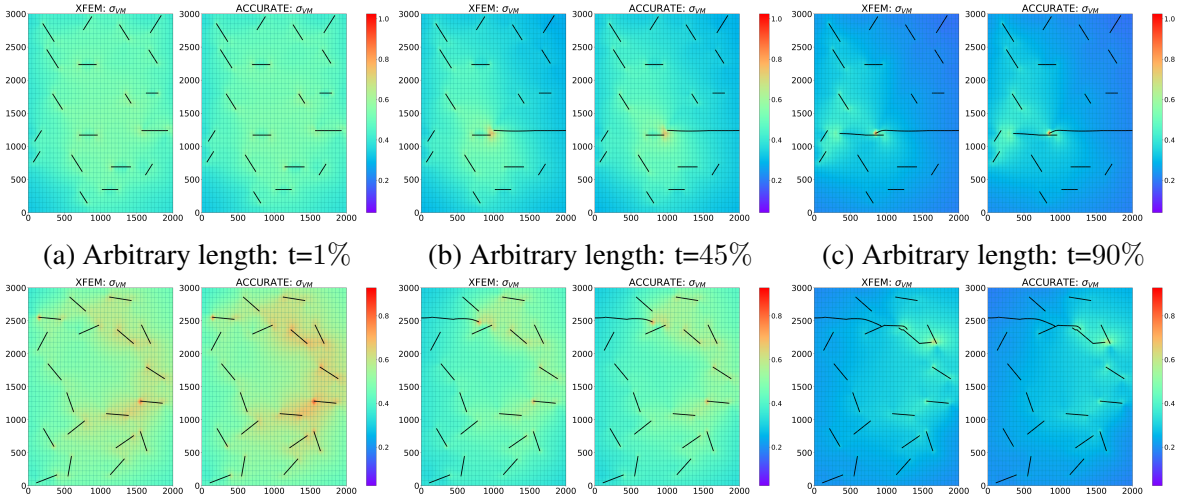


(d) Horizontal domain: t=1% (e) Horizontal domain: t=45% (f) Horizontal domain: t=90%

Figure 4.4: Evolution of von Mises stress (MPa) from $t = 1\%$ to $t = 90\%$ for (a-c) test cases with square domain, and (d-f) test cases with horizontal domain.

geometries—vertical, square, or horizontal—ACCURATE demonstrates versatility in simulating fracture occurrences across new domains.

Figure 4.4 illustrates the temporal evolution of von Mises stress from $t = 1\%$ to $t = 90\%$, where $\%$ represents a percentage of the total number of time-steps (101) until failure, for both the square domain and the horizontal domain cases. In the square domain scenario, ACCURATE consistently produces stress distributions virtually identical to those generated by XFEM from $t = 1\%$ to $t = 90\%$. Likewise, in the case of the horizontal domain, ACCURATE's stress evolution closely mirrors that of the XFEM model. We present comprehensive quantitative analyses for these scenarios in the next section.



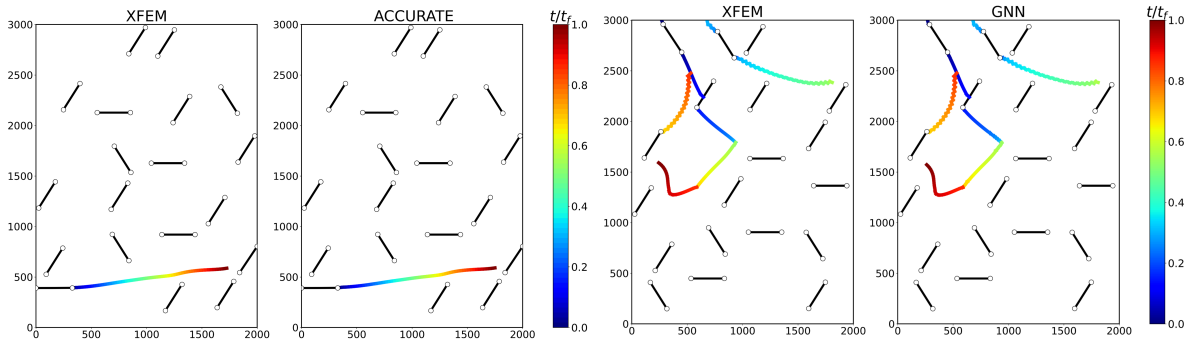
(a) Arbitrary length: t=1% (b) Arbitrary length: t=45% (c) Arbitrary length: t=90%

(d) Arbitrary angle: t=1% (e) Arbitrary angle: t=45% (f) Arbitrary angle: t=90%

Figure 4.5: Evolution of von Mises stress (MPa) from $t = 1\%$ to $t = 90\%$ for (a-c) test cases with arbitrary crack length, and (d-f) test cases with arbitrary crack orientation.

Finally, we assessed the framework's performance in predicting stress evolution for scenarios involving cracks of arbitrary lengths and orientations. For these case studies, we maintained a fixed domain size of $2000mm \times 3000mm$. We present the stress evolution results for arbitrary crack lengths in Figures 4.5a-4.5c, and arbitrary crack orientations in Figures 4.5d-4.5f.

For scenarios with arbitrary crack lengths, ACCURATE demonstrates excellent agreement with XFEM stress predictions across all depicted time steps. These outcomes suggest that the framework, initially trained for vertical domains under tension, effectively transfers knowledge to scenarios with arbitrary crack lengths. However, while Figures 4.5d-4.5f display generally accurate stress distributions, slight discrepancies appear in regions of high stress interactions in cases involving arbitrary angles. For example, at $t = 1\%$ the highest stress interaction occurs at approximately $x = 1600mm, y = 1200mm$ near the right-most crack. Here, the predicted stress distribution by ACCURATE appears slightly higher compared to the XFEM simulation, indicating a minor discrepancy. This discrepancy may arise from the propagation of the right-most crack (at approximately $x = 1700mm, y = 2300mm$) with orientation close to 90^o.. Despite this error, ACCURATE effectively identifies crack-tips with the highest stress distributions for all depicted time steps.



(a) Tensile load - Crack path

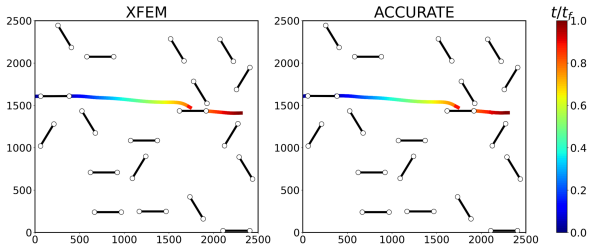
(b) Shear load - Crack path

Figure 4.6: Evolution of crack path for: a) Test case subjected to Tensile load. b) Test case subjected to Shear loading.

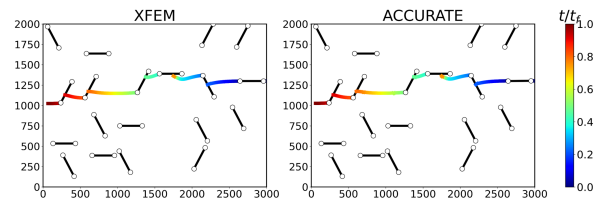
4.3.3 Prediction of microcrack propagation and coalescence

Next, we perform a similar qualitative analysis for the predicted crack paths (using *P*-GNN) of each simulation shown in Section 4.3.2. Figures 4.6a and 4.6b show the evolution of crack growth for the case studies involving a vertical domain subjected to tension and shear loads, respectively. From Figure 4.6a, the predicted evolution of crack growth for the vertical domain subjected to tension is qualitatively indistinguishable from that generated by the XFEM model. This is consistent with the Microcrack-GNN which performed predictions of crack growth for a vertical domain with high accuracy. Similarly, for the vertical domain subjected to shear load shown in Figure 4.6b the predicted crack path evolution by ACCURATE is also qualitatively identical to the XFEM-based simulator. In Section 4.3.2, we showed the GNN framework's ability to predict stress evolution for cases involving shear loads. Because ACCURATE is able to predict the stress intensity factors at future time steps prior to predicting the future crack-tip positions, as shown in equation (4.4), we take advantage of these prior predictions in the *P*-GNN model by including the predicted $\hat{K}_I^{T+1}$ and $\hat{K}_{II}^{T+1}$ as part of *P*-GNN's input. Therefore, the high accuracy achieved when predicting the stress evolution aids *P*-GNN to achieve high accuracy for the case study involving shear loading.

Figures 4.7a - 4.7b show the evolution of crack growth of the square domain and horizontal domain cases, respectively, for the XFEM simulator versus ACCURATE. Similar to the predicted stress evolution of the square domain case, the predicted crack growth evolution is qualitatively identical to the XFEM crack path evolution. For the horizontal domain case study, we obtained



(a) Square domain - Crack path



(b) Horizontal domain - Crack path

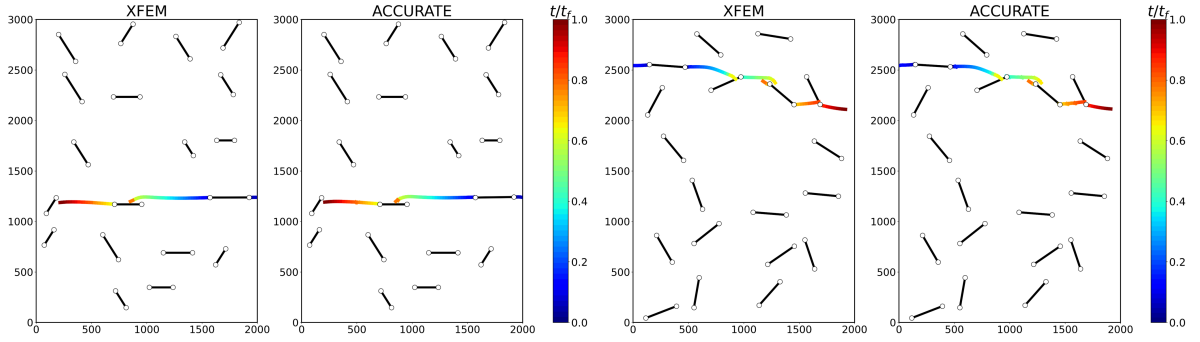
Figure 4.7: Evolution of crack path for: a) Test case with Square domain. b) Test case with Horizontal domain.

good crack path predictions compared to XFEM. A detailed error analysis is described in Section 4.3.4.

Lastly, Figures 4.8a - 4.8b show the XFEM versus predicted crack growth evolution for the case studies involving arbitrary length and arbitrary angles, respectively. Similarly to the accuracy of the predicted stresses in the arbitrary length case, the predicted crack growth for this case shows good agreement with the XFEM simulator. This result shows the ACCURATE framework is capable of emulating both the stress and crack growth of cases with arbitrary crack lengths. Additionally, the simulation with arbitrary crack angles qualitatively shows good prediction accuracy. We emphasize that while the simulation showing the most observable difference in predicted stress versus XFEM was for the case of arbitrary angle during the initial time-step ($t = 1\%$), the *K*-GNN model still captured the regions with the highest stress between interacting crack-tips, as well as the crack-tips with the highest stress intensity factors across the remaining time-steps. These overall good predictions by ACCURATE may facilitate *P*-GNN to predict future crack-tip positions with good accuracy compared to the XFEM model as shown in Figure 4.8b. Therefore, Figures 4.6 - 4.8 show a qualitative result for the capability of the ACCURATE framework to predict crack growth evolution of cases with variable configurations using TL on very small datasets.

4.3.4 Errors on effective stress intensity factor and crack path

we conduct an in-depth analysis of errors in crack paths and effective stress intensity factors over time for each simulation detailed in Sections 4.3.2 and 4.3.3. Following the approach described for MicroCrack-GNN, we assess the error in predicted K_I and K_{II} factors, by computing the



(a) Arbitrary crack length - Crack path (b) Arbitrary crack angle - Crack path

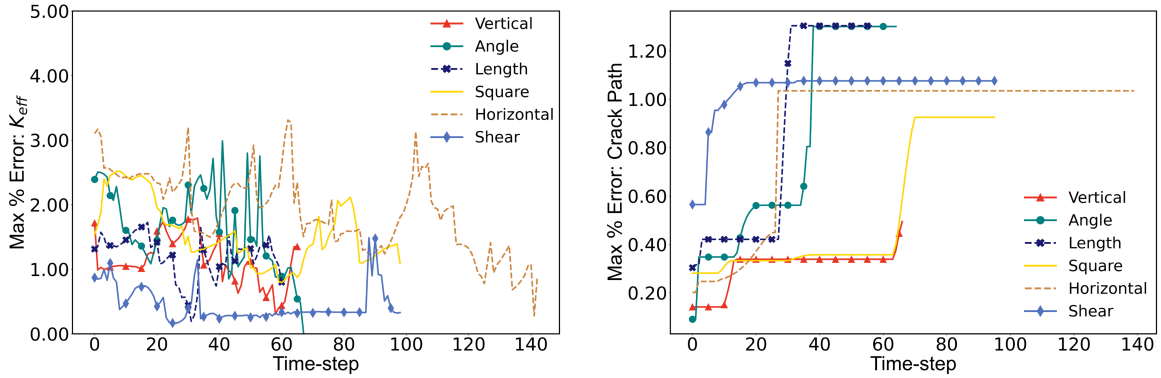
Figure 4.8: Evolution of crack path for: a) Test case with Arbitrary crack length. b) Test case with Arbitrary crack orientation.

error in effective stress intensity factors, K_{eff}. Figures 4.3 - 4.5 highlight that the greatest errors in stress occur at propagating crack-tips and their nearest neighboring crack-tips. In quasi-static fracture, the propagation of crack-tips is dictated by those where the effective stress intensity factor equals or exceeds the critical stress intensity factor, denoted as $K_{eff} \geq K_c$. Consequently, Figure 4.9a illustrates the progression of maximum percent error in K_{eff} derived from predicted K_I and K_{II} for the simulations outlined in Section 4.3.2.

We compute the error in K_{eff} using equation (4.5).

$$K_{eff_{\%error}} = \max_{s \in N_{crt}^t} \left(\frac{\left| {K_{eff}}^t{}_{Pred} - {K_{eff}}^t{}_{True} \right|}{{K_{eff}}^t{}_{True}} \right)_s \times 100 \qquad \{t = 1, 2, \ldots, T_f\}, \tag{4.5}$$

Here, N_{crt}^t represents the number of crack-tips where the effective stress intensity factor K_{eff} is greater than or equal to the critical stress intensity factor K_{crt} at a specific time t. ${K_{eff}}_{Pred}^t$ denotes the predicted K_{eff} at time t generated by ACCURATE, while ${K_{eff}}_{True}^t$ represents the actual K_{eff} at time t computed by the XFEM fracture model. The fluctuations observed in Figure 4.9a indicate variations in the errors of K_{eff} over time. On average, across all time steps, the errors for the vertical domain, shear load, square domain, horizontal domain, arbitrary length, and arbitrary angle cases are $1.13 \pm 0.36\%$, $0.46 \pm 0.27\%$, $1.55 \pm 0.49\%$, $1.89 \pm 0.61\%$, $1.20 \pm 0.34\%$, and $1.64 \pm 0.62\%$, respectively. The highest error jumps of approximately 3.2% and 3.0% occur for the horizontal domain and arbitrary crack angle simulations, respectively. Conversely, the lowest error in K_{eff} is observed in the simulation involving shear loading.



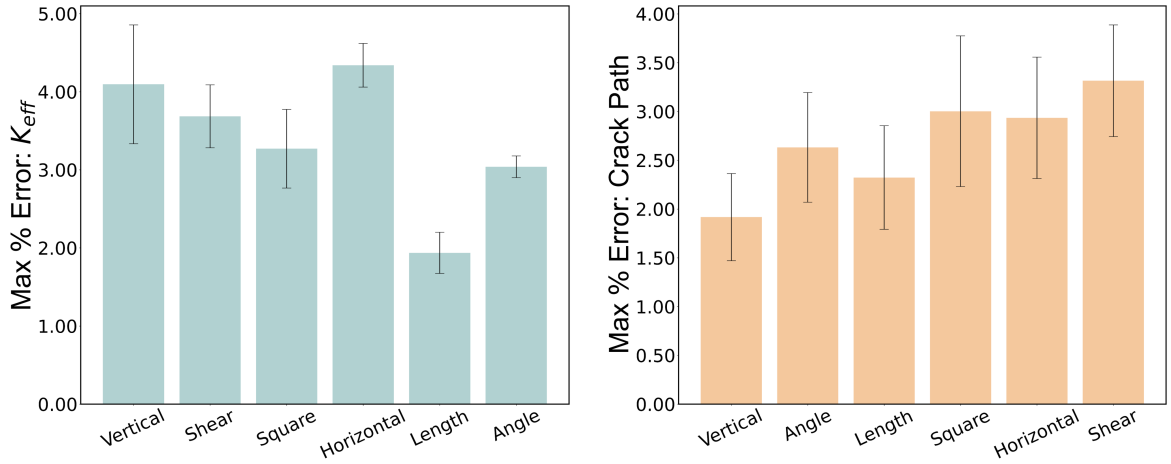
(a) Maximum error in K_{eff} vs. time for each case study

(b) Maximum error in crack path vs. time for each case study

Figure 4.9: The maximum percentage errors in effective stress intensity factor and crack path over time for the simulations outlined in Sections 4.3.2 through 4.3.3.

Furthermore, In Figure 4.9b, we show the maximum percentage errors for the predicted crack paths. Unlike the errors of K_{eff} which showed a fluctuating trend, the crack path maximum error remains relatively constant over time until it reaches a new maximum error. We compute the maximum crack path error using the maximum of the difference between the crack growth predicted by the XFEM model and the crack growth predicted by ACCURATE. The highest errors in crack growth over time are observed for the cases involving arbitrary crack length and arbitrary crack orientation, both reaching approximately 1.35%. In contrast, the lowest error of approximately 0.5% is observed for the standard vertical domain case of constant crack lengths and orientations. In summary, from Figure 4.9b, the average errors across time are (i) $0.31 \pm 0.08\%$ for the vertical domain, (ii) $1.04 \pm 0.12\%$ for shear load, (iii) $0.51 \pm 0.26\%$ for the square domain, (iv) $0.89 \pm 0.29\%$ for the horizontal domain, (v) $0.85 \pm 0.44\%$ for arbitrary crack lengths, and (vi) $0.82 \pm 0.42\%$ for arbitrary crack orientations.

We then show the highest maximum errors in crack paths and K_{eff} for each case study in Figures 4.10b and 4.10a, respectively. To generate Figures 4.10b and 4.10a, we start by computing the timewise maximum error for each test simulations. Then, we identify the simulation with the highest error across all time steps for each case study. The case resulting in the highest error in K_{eff} of $4.34 \pm 0.28\%$ was the horizontal domain, and the lowest error in K_{eff} of $1.94 \pm 0.26\%$ was obtained for the arbitrary length case.



(a) Maximum error in predicted K_{eff} for each case study

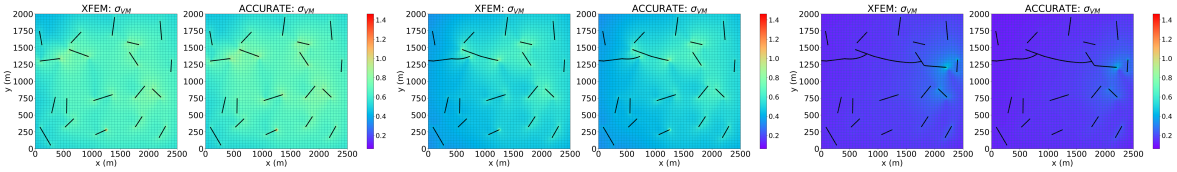
(b) Maximum error in predicted crack path for each case study

Figure 4.10: The maximum percentage error in the predicted stress intensity factor and crack path across all test simulations for each case study.

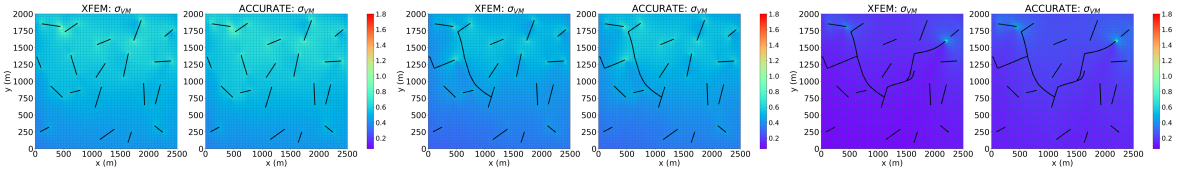
For crack paths shown in Figure 4.10b, the maximum errors demonstrate high prediction accuracy across each case study, with all test cases showing maximum errors below than 4%. The highest error in crack paths of $3.32 \pm 0.57\%$ was obtained for shear load , while the lowest error of $1.92 \pm 0.45\%$ was obtained for the vertical domain case. We observe good prediction accuracy by the ACCURATE framework across all simulations in the test dataset of (i) 95.5% for crack growth, and (ii) 96.5% for stress intensity factors predictions. By integrating TL, the ACCURATE framework only required 20 simulations consisting of new arbitrary initial crack configurations, boundary dimensions, and shear loading scenarios to achieve high prediction accuracy.

4.3.5 Unseen Cases

In this section, we highlight a significant aspect of the proposed ACCURATE framework: its capability to predict stresses and crack growth for entirely new, unseen scenarios without requiring additional TL implementations. We emphasize from Figure 4.1c, that ACCURATE was only exposed to cases involving arbitrary crack lengths and orientations in vertical domains under tension during the initial TL update steps. However, during TL for square domains, horizontal domains, and shear loading, arbitrary crack lengths and orientations were not considered. Additionally, TL of square and horizontal domains did not consider shear loads. Therefore,



(a) Tension - 2500×2000: t=1% (b) Tension - 2500×2000: t=45% (c) Tension - 2500×2000: t=90%



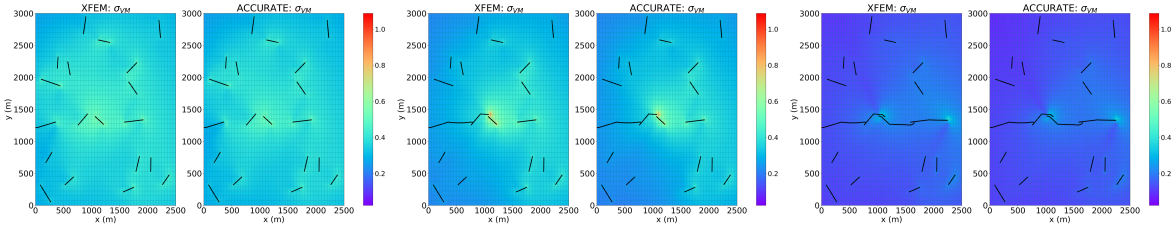
(d) Shear - 2500×2000: t=1% (e) Shear - 2500×2000: t=45% (f) Shear - 2500×2000: t=90%

Figure 4.11: The von Mises stress evolution (MPa) from $t = 1\%$ to $t = 90\%$ for: (a-c) An unseen case of a 2500mm $\times$ 2000mm domain with arbitrary crack lengths and crack orientations subjected to tensile load. (d-f) An unseen case of a 2500mm $\times$ 2000mm domain with arbitrary crack lengths and crack orientations subjected to shear load.

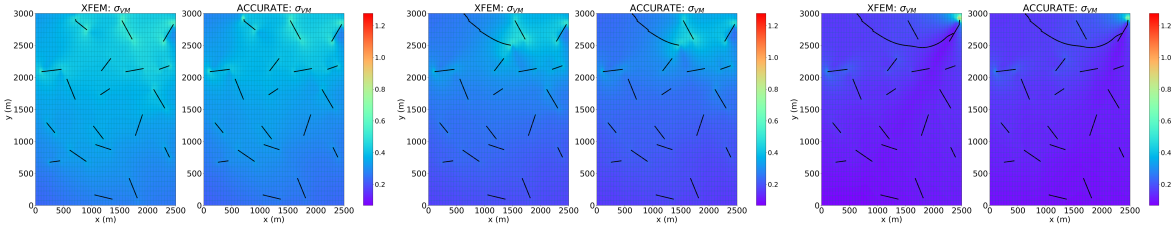
in this section we demonstrate the framework's capability to to predict new unseen cases without implementing additional TL steps. For this analysis, we introduce four new unseen case studies involving (i)-(ii) a domain of size $2500mm \times 2000mm$ with arbitrary crack lengths and orientations under tension and shear loads, and (iii)-(iv) a domain of size $2500mm \times 3000mm$ with arbitrary crack lengths and orientations under to tension and shear. This approach showcases the framework's ability to handle scenarios involving arbitrary crack orientations and lengths without necessitating retraining or the generation of new time-consuming and computationally demanding simulations.

In Figures 4.11a-4.11f, and Figures 4.12a-4.11f, we illustrate the evolution of von Mises stresses for both XFEM and ACCURATE from $t = 1\%$ to $t = 90\%$ for the new, unseen scenarios (i)-(ii), respectively. Despite these configurations—featuring boundary dimensions of $2500mm \times 2000mm$ and $2500mm \times 3000mm$, along with arbitrary crack lengths and orientations—being absent during the ACCURATE framework's training, it adeptly predicts stress distributions comparable to XFEM results. This high prediction accuracy for unseen cases is evident in both tension and shear loading scenarios.

We also perform a similar qualitative assessment for the predictions of crack growth by the *P*-GNN model for the new unseen cases. The evolution of crack growth for cases (i) and (ii) is illustrated in Figures 4.13a and 4.13b, respectively. Likewise, in Figures 4.14a and 4.14b we



(a) Tension - 2500×3000: t=1% (b) Tension - 2500×3000: t=45% (c) Tension - 2500×3000: t=90%



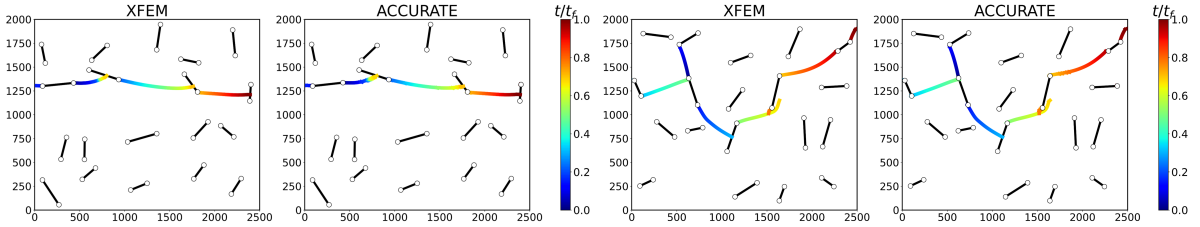
(d) Shear - 2500×3000: t=1% (e) Shear - 2500×3000: t=45% (f) Shear - 2500×3000: t=90%

Figure 4.12: The von Mises stress evolution (MPa) from $t = 1\%$ to $t = 90\%$ for: (a-c) An unseen case of a 2500mm $\times$ 3000mm domain with arbitrary crack lengths and crack orientations subjected to tensile load. (d-f) An unseen case of a 2500mm $\times$ 3000mm domain with arbitrary crack lengths and crack orientations subjected to shear load.

depict the crack growth evolution for cases (iii) and (iv), respectively. Across all these unseen scenarios, the predicted crack paths exhibit visually similar crack patterns to those generated by the XFEM model.

Next, in Figures 4.15a and 4.15b we present the maximum errors in both K_{eff} and crack path over time for the unseen cases. Among the errors in K_{eff}, the highest error peak of 4.15% occurred in the unseen case featuring a $2500mm \times 2000mm$ domain under to shear loading. For unseen cases (i) through (iv), the average of the maximum errors across time were 2.31 ± 0.75, 1.2 ± 0.74, 1.92 ± 0.55, and 0.89 ± 0.38, respectively, for K_{eff}. These findings underscore ACCURATE's capability to replicate stress evolution accurately in novel domain dimensions with cracks of arbitrary lengths and orientations under tension or shear. The framework demonstrates high prediction accuracy for K_{eff} (approximately 96%) compared to XFEM.

Applying the same approach to the errors in crack path, the highest error peak of approximately 2.9% was observed for the $2500mm \times 2000mm$ domain under tension. The average errors in crack path across time for unseen cases (i) through (iv) were 0.58 ± 0.27, 1.59 ± 0.67, 1.77 ± 0.58, and 0.97 ± 0.08, respectively. The observed errors in crack path, coupled with



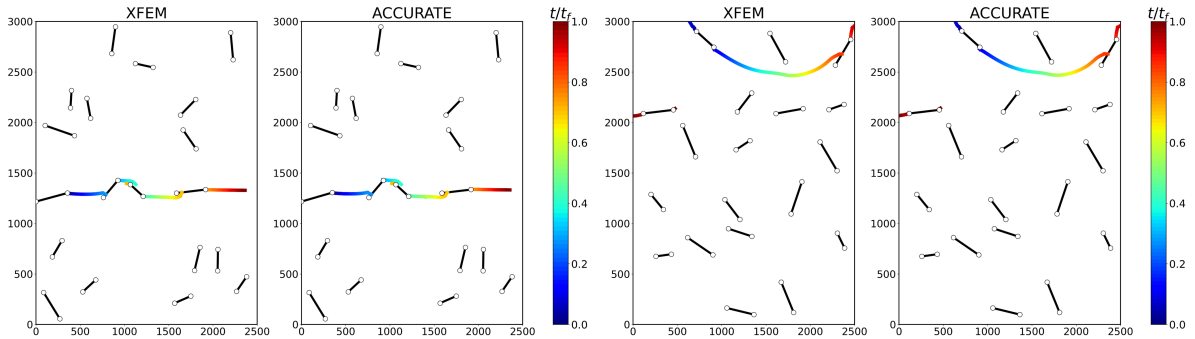
(a) Tension: 2500×2000 (b) Shear: 2500×2000

Figure 4.13: Evolution of crack path for: a) An unseen case of a 2500mm × 2000mm domain with arbitrary crack lengths and crack orientations subjected to tensile load. b) An unseen case of a 2500mm × 2000mm domain with arbitrary crack lengths and crack orientations subjected to shear load.

Figure 4.15b, demonstrate ACCURATE's capability to predict crack growth with remarkable accuracy for the unseen cases (approximately 97%). One potential explanation for ACCURATE's adeptness in handling new arbitrary crack lengths and angles without necessitating TL for achieving high accuracy could be attributed to both the normalization and randomization approaches implemented. For instance, for a given time-step in a simulation involving a vertical domain under tension, two or more microcracks may have already merged, thus, forming a single larger crack with arbitrary length and orientation. This characteristic is effectively captured by the normalization and randomization processes inherent in the dataset, aiding ACCURATE in its accurate predictions. The normalization criteria for the crack-tip positions used was obtained from [207, 208]. Specifically, the x-coordinate positions were normalized by the domain's width (e.g., $2000mm$ for vertical and $3000mm$ for horizontal), while the y-coordinate positions were normalized by the domain's height (i.e., $3000mm$ for vertical and $2000mm$ for horizontal). Hence, these findings underscore that while TL was applied for case study independently, the sequential implementation of TL enabled the framework to generalize across various combinations of problem-specific inputs without necessitating additional TL.

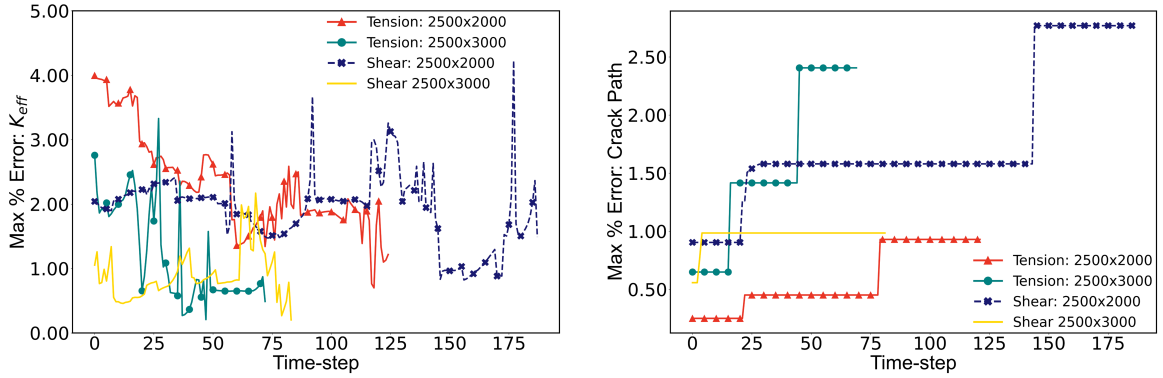
4.3.6 Analysis time VS. number of microcracks

To assess the computational efficiency of the ACCURATE framework, we measured the total simulation time required by both the XFEM model and ACCURATE framework to generate 10 test simulations in the vertical domain study. In Figure 4.16a, we compare the average simulation time of the high-fidelity XFEM model (depicted in green) versus the ACCURATE



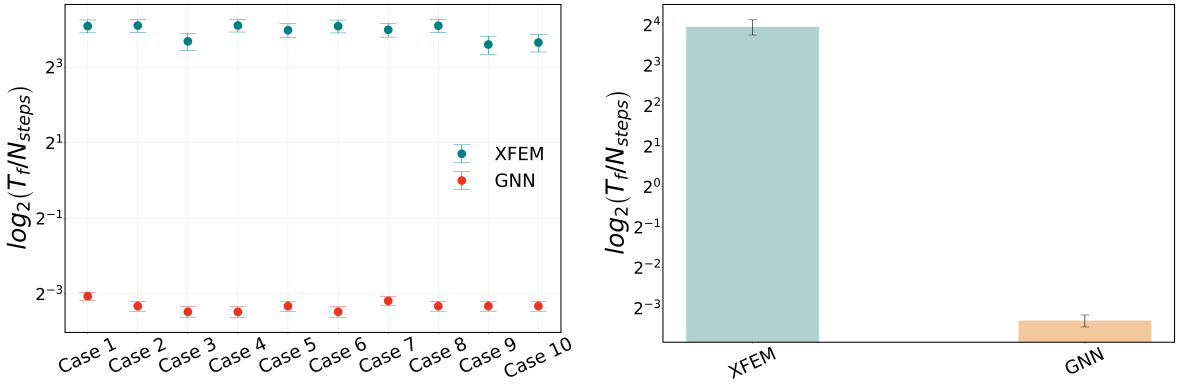
(a) Tension: 2500×3000 (b) Shear: 2500×3000

Figure 4.14: Evolution of crack path evolution for: a) An unseen case of a 2500mm × 3000mm domain with arbitrary crack lengths and crack orientations subjected to tensile load. b) An unseen case of a 2500mm × 3000mm domain with arbitrary crack lengths and crack orientations subjected to shear load.



(a) Maximum error in K_{eff} vs. time for unseen cases (b) Maximum error in crack path vs. time for unseen cases

Figure 4.15: The maximum percentage errors with respect to time in effective stress intensity factor and crack path for each unseen case study.



(a) Simulation time of all test cases

(b) Average simulation time across test cases

Figure 4.16: a) Comparison of simulation time (seconds per time-step) between XFEM surrogate model and ACCURATE for all simulations in the test dataset. b) Average simulation time (seconds per time-step) comparison between XFEM surrogate model and ACCURATE across all test simulations.

framework (depicted in red) for each simulation. Additionally, in Figure 4.16b we show the average across all 10 simulations shown in Figure 4.16a, in order to compare the XFEM model (shown in light blue) versus the ACCURATE framework (shown in light orange). For the XFEM model, we obtained each simulation using a personal laptop equipped with an Intel Core i9-12900H CPU operating at 2.50GHz and 16.0GB of RAM. For ACCURATE, each simulation was gathered using the same personal computer involving a laptop-grade GPU NVIDIA GeForce RTX 3070 Ti. ACCURATE's simulation time was computed from the generation of the initial graph representation at t_0 to the prior GNNs predictions, *K*-GNN and *C*-GNN, followed by the final GNN prediction, *P*-GNN.

To develop ACCURATE, we leveraged the open-source and optimized PyTorch Geometric (PyG) library for the development of ACCURATE. The optimized PyG library resulted in a significant time improvement of up to 2 orders of magnitude faster (200x) compared to the XFEM model. We show the simulation speedup in Figure 4.16. While the XFEM fracture model demands over 10 seconds for each time-step in the simulation, ACCURATE completes each time-step in approximately 0.1 seconds. Consequently, the XFEM model would necessitate roughly 28 hours to generate a total of 100 simulations, whereas the ACCURATE framework achieves this task in approximately 17 minutes. It's important to note that the XFEM model was not parallelized, which constrains our performance comparison.

4.4 Conclusion

In conclusion, this work presents ACCURATE, an accelerated and versatile fracture mechanics framework capable of accurately simulating fracture phenomena induced by multiple cracks' interactions, propagation, and coalescence in brittle materials across diverse problem configurations. ACCURATE achieves this by predicting Mode-I and Mode-II stress intensity factors for each crack-tip and simulates crack growth by predicting the future positions of all crack-tips. Through the application of TL to the pretrained MicroCrack-GNN model, we conducted studies on new problem-specific configurations while significantly reducing the required training simulations to just 20 (compared to 960 for MicroCrack-GNN). By implementing 5 sequential TL update steps covering cases with arbitrary crack lengths, orientations, square and horizontal domains, and shear loading, ACCURATE acquired generalized knowledge of fracture mechanics. This was demonstrated by ACCURATE's ability to accurately simulate stress evolution and crack propagation for new unseen cases involving different domain dimensions along with arbitrary crack lengths and orientations under both tension and shear loads. These capabilities highlight ACCURATE's potential for exploring a wide range of problem configurations beyond those examined in this study.

Another noteworthy aspect of the ACCURATE framework is its remarkable improvement in simulation speed. ACCURATE exhibited a speedup of 2 orders of magnitude (i.e., approximately 200 times faster) compared to the high-fidelity XFEM model. This significant acceleration suggests that an ideally parallelized XFEM model would require around 200 CPU cores in order to achieve comparable performance. Moreover, the development of ACCURATE highlights the advantages of utilizing ML techniques such as TL and GNNs to develop reduced-order and accelerated computational models to simulate fracture mechanics. These models can be trained effectively even with very small datasets, leading to reduced computational costs. In future research, ACCURATE could be extended to incorporate dynamic effects such as crack bifurcation, and ductile material properties. By further enhancing its capabilities, the framework has the potential to address a wider range of fracture mechanics problems, making it even more versatile and applicable in various engineering contexts.

Chapter 5

Dynamic and adaptive mesh-based graph neural network framework for multiphysics problems

5.1 Introduction

As described in previous chapters, the development of computationally efficient and dependable modeling techniques remains a continuous endeavor in the fields of solid mechanics and material science. In various engineering disciplines, high-fidelity mesh-based computational models serve as indispensable tools for simulating complex physical phenomena, thereby circumventing the need for costly experiments. Typically, these models involve solving coupled systems of partial differential equations (PDEs) that govern the underlying physics of the problem at hand. Traditionally, a widely adopted strategy to tackle such PDEs and advance the physics over time has been through discretization of the problem domain into a mesh. Subsequently, numerical approximations of the PDE solutions are obtained within this mesh framework. Although this approach has demonstrated its efficacy by yielding accurate and dependable results in numerous applications, its computational cost escalates rapidly as the complexity of the problem increases.

Fracture mechanics is a fundamental area of study in materials engineering, as it pertains to the understanding and prediction of material failure. Traditionally, physics-based models have been employed extensively in this field, offering insights into the behavior of cracks and their propagation. Computational models for material failure can generally be categorized into two methodologies: sharp interface methods and diffused interface methods. In sharp interface methods, cracks are treated as discontinuities in the displacement field, while in diffused interface methods, the discontinuity is smoothed out over a length scale using a continuous surrogate field. In [209], and references therein, a thorough review and comparative study of these methods can

be found. One of the most prominent diffused interface methods is the multiphysics phase field (PF) technique [81, 81, 82, 210, 211, 212, 213, 214]. This approach involves formulating an energy functional, denoted as Π, which regularizes the crack over a characteristic length scale ϵ using a smooth scalar damage field, ϕ, typically taking values between 0 and 1. The evolution of the scalar damage field, ϕ, is formulated by minimizing the energy functional, leading to the propagation and evolution of cracks. PF fracture models have gained widespread adoption due to their scalability and ease of implementation [80, 86, 215]. They have been successfully applied in various scenarios, involving crack propagation, nucleation, and branching across diverse material ranging from brittle and ductile materials to composite materials with anisotropy, as well as biological systems [216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226].

While diffused interface methods like the PF technique offer a promising approach to simulating crack propagation, they come with computational challenges. For instance, the characteristic length scale of material damage is often much smaller than the overall domain size, necessitating high mesh resolution in the vicinity of cracks. To mitigate the computational cost associated with this high mesh resolution, PF methods are commonly combined with Adaptive Mesh Refinement (AMR) techniques. AMR allows for the use of different mesh resolutions, employing coarser meshes in regions where minimal changes occur in the problem's physics and finer meshes where significant changes are present. This adaptive approach helps optimize computational resources while accurately capturing crack propagation dynamics. Several studies have highlighted the pivotal role of AMR in enhancing the efficiency of phase field models [227, 228, 229, 230, 231]. Efforts to improve the computational efficiency and robustness of PF fracture methods have also led to the development of various solver techniques, including staggered, monolithic, and fast Fourier transform-based solvers. Each of these approaches has its advantages and drawbacks, reflecting the ongoing pursuit of optimization in PF modeling. Despite these advancements, PF models still entail solving complex systems of coupled PDEs, resulting in computational costs that escalate with problem complexity. For example, PF methods necessitate computing an additional (pseudo) time-dependent PDE in order to propagate the scalar damage field, further adding to computational expenses. These computational demands limit the widespread application of PF fracture techniques in large-scale scenarios such as

simulating fracture in glaciers, bridges, and subsurface fracture network evolution. Addressing these challenges remains an active area of research, with ongoing efforts focused on enhancing the efficiency and scalability of PF models for broader practical use.

Reduced-order modeling techniques, particularly Machine Learning (ML), offer a promising solution to address the challenges associated with mesh-based models. Several studies have demonstrated the efficacy of ML in various fields of solid mechanics and material science. These include applications in structural health monitoring [232], smart materials [233], fracture mechanics [234, 235, 236], structural analysis [237], and applied mechanics [238, 239]. Other recent works have explored deep learning architectures for predicting stress-strain responses [240, 241], fracture behavior [242], hierarchical material structures [243], and material property estimation [244, 245]. ML techniques have also demonstrated significant advancements in PF models, offering accelerated and accurate predictions in material science and solid mechanics applications [246, 247, 248, 249, 130, 250]. In a notable study by Montes et al. [251], Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs) were integrated with PF models to simulate the evolution of two-phase mixtures, particularly spinodal decomposition. This ML-PF hybrid approach provided remarkable speedups compared to high-fidelity PF models, achieving orders of magnitude reduction in computational time while maintaining accuracy.

Furthermore, as shown in Chapters 3 and 4, GNN techniques have showcased remarkable success in dynamic simulation problems due to their accelerated performance [156, 155, 160, 3]. The graph representation approach utilized by GNNs aligns well with mesh-based problems, where the simulation mesh can directly serve as the model's graph representation. In a recent study [174], researchers developed *MeshGraphNet*, a mesh-based graph neural network tailored for simulating finite element simulations. This model demonstrated high accuracy when compared against FEM simulations, showcasing its efficacy in simulating various engineering phenomena such as flag dynamics, plate bending, and flow over rigid bodies. Notably, MeshGraphNet provided simulation speeds ranging from one to two orders of magnitude faster than traditional FEM approaches. Mesh-based GNNs have found applications in diverse engineering problems, including simulating crack propagation in PF models with AMR [5], FE

displacements and stresses [252, 253, 254], and flow over airfoils and cylinders [255, 256]. Although mesh-based GNNs have shown promise, adapting this approach for PF simulations with AMR has remained unexplored. An mesh-based GNN framework with AMR would need to dynamically incorporate new nodes and edges, representing the refined mesh at each time-step, presenting a unique and challenging problem.

In this chapter, we introduce a novel dynamic and adaptive mesh-based Graph Neural Network (*ADAPT*-GNN) designed to replicate PF models of single-edge crack propagation while achieving a remarkable simulation speed-up of up to 36x. Leveraging the second-order PF model with AMR as outlined in [211], we construct the training, validation, and test datasets by varying the initial crack position, lengths, and orientations. The *ADAPT*-GNN architecture is adept at generating new graph structures at each time-step by dynamically incorporating or removing nodes and edges through the AMR strategy. This unique methodology enables us to harness the benefits of operating with smaller mesh sizes, characterized by fewer cells, nodes, and edges, while also leveraging the computational efficiency offered by dynamic GNNs. Utilizing this framework, we predict displacements ($\mathbf{u}$, ν) and the scalar damage field (ϕ) at each time-step for every point within the adaptive mesh. Subsequently, we utilize the predicted displacements and scalar damage field to compute the stress evolution across the domain. We anticipate that this novel approach will open avenues for significantly accelerated mesh-based simulations across other mechanics and material science problems. Although our current investigation focuses on PF fracture models, we believe that this methodology can be readily extended to address other materials and mechanics problems formulated similarly to PF models.

5.2 Methods

5.2.1 Phase field fracture model with adaptive mesh refinement

We employ the open-source PF fracture model with AMR outlined in [211] to simulate a range of fracture mechanics scenarios. As illustrated in Figure 5.2a, we consider problems involving brittle materials featuring single-edge notched cracks subjected to tension.

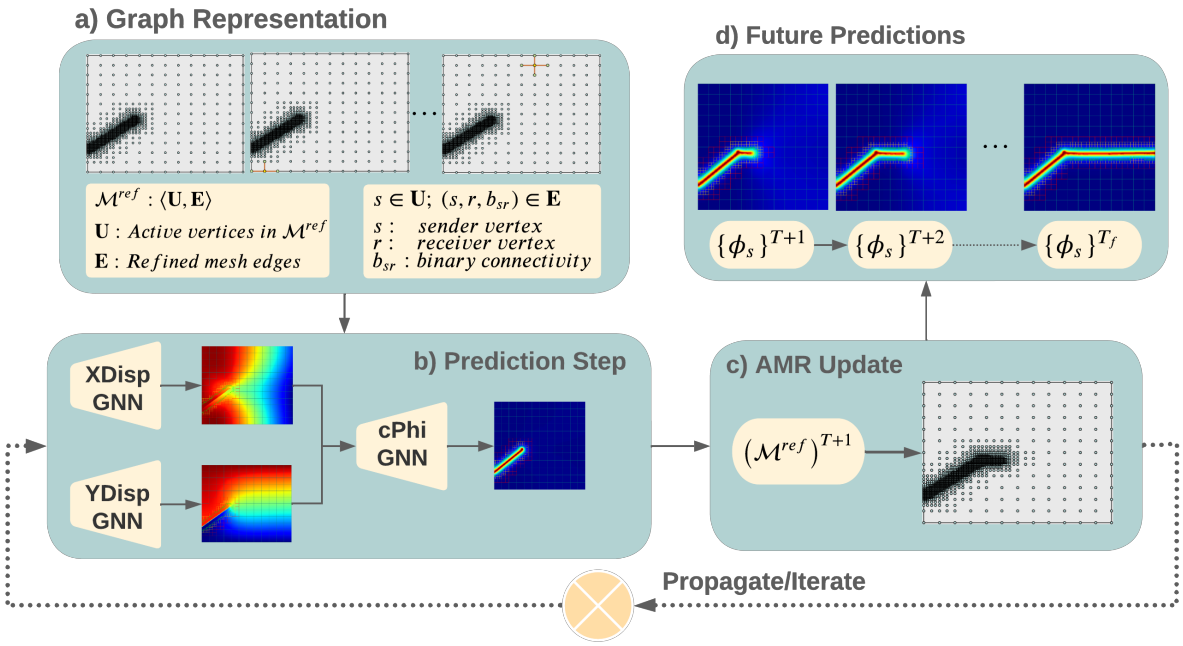
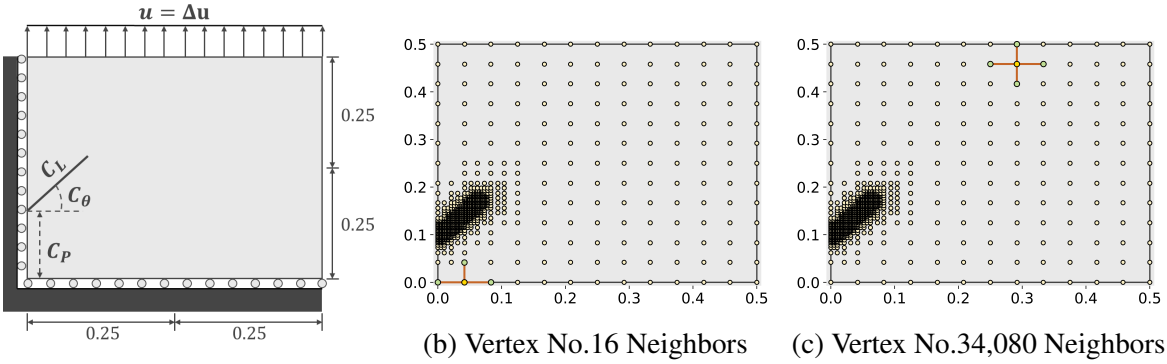


Figure 5.1: Structure of the phase-field AMR-based GNN framework: a) The graph representation including nodes and edges for the refined mesh. b) The prediction step involving XDisp-GNN and YDisp-GNN for predicting displacements at $t+1$, and cPhi-GNN for predicting the scalar damage field at $t+1$ using the predicted displacements as input. c) The AMR update step illustrates the process of adaptive mesh refinement for the time step $t+1$. d) Illustration of future predictions for $T+1, T+2, ..., T_f$.

The second-order PF model tailored for fracture problems formulates the energy functional, $\mathcal{F}$, as:

$$\mathcal{F} = \int_{\Omega} \left[\mathcal{W}\left(\varepsilon(\mathbf{u}), \phi\right) + \frac{\mathcal{G}_c}{2\epsilon} \left(\phi^2 + \epsilon^2 |\nabla \phi|^2 \right) \right] d\Omega. \tag{5.1}$$

Here, $\mathcal{G}_c$ denotes the fracture energy, ϕ denotes the scalar damage field, $\mathbf{u}$ denotes the displacement field, $\mathcal{W}$ denotes the strain energy density, the strain energy density is a function of the strain ε, and Ω denotes the material domain. The PF model from [211] is implemented in MATLAB. The model utilizes the isogeometric analysis (IGA) numerical tool. For local refinement, this framework leverages polynomial splines over hierarchical T-meshes (PHT-splines). To dynamically adjust mesh resolution in areas of high gradients and singularities, an adaptive h-refinement scheme is integrated with PHT-splines. Furthermore, the hybrid-staggered algorithm is employed to refine mesh resolution iteratively until a convergence threshold is reached. This iterative refinement process ensures the accurate capture of changes in local zones at each



(a) Geometrical configuration

(b) Vertex No.16 Neighbors

(c) Vertex No.34,080 Neighbors

Figure 5.2: a) Illustration of the problem geometry and the configuration of input parameters C_L, C_P, and C_θ. b-c) Visualization of active nodes and active edges connected to their neighboring mesh vertices.

time-step while effectively reducing computational costs. Utilizing the second-order phase field fracture model, we generate a substantial dataset of two-dimensional fracture simulations featuring single-edge notched cracks.

5.2.2 Graph representation

A pivotal aspect of the developed GNN is its capability to dynamically adapt the architecture of the graph to correspond with the refined mesh at every time-step. This feature ensures that the GNN can effectively harness the computational efficiency afforded by the AMR technique. We recall from [211] that the *h*-refinement defined the two-dimensional representation of the mesh, $\mathcal{M}$, using $\mathcal{U}^i = \{\xi_1^i, \xi_2^i, \ldots, \xi_{n_i+1}^i\}$. Here, $\mathcal{U}^i$ includes the set of active vertices, ξ denotes the number of active elements, and n_i the parametric direction where $i \in \{1, 2\}$ in a two-dimensional case. For the graph representation of the mesh-based GNNs, we adapt this notation and define the instantaneous refined graphs as $\mathcal{M}^{ref} : \langle \mathbf{U}, \mathbf{E} \rangle$. Here, $\mathbf{U}$ and $\mathbf{E}$ denote the active vertices and edges in $\mathcal{M}^{ref}$, respectively. As shown in Figures 5.2b - 5.2c, we note that $\mathbf{E}$ includes edges connecting each active node, $\xi_s \in \mathbf{U}$, to the adjacent active nodes. We also note that $s : \{1, 2, \ldots, N\}$ for any positive integer, where N denotes the total number active vertices in $\mathcal{M}^{ref}$.

At each time-step, the vertices are characterized by their spatial positions $\hat{\mathcal{P}}_s$, their neighboring active mesh nodes (adjacent nodes) $\hat{\mathcal{A}}_s$, their displacement values $\hat{\mathcal{D}}_s$ (comprising x-

and y-displacement fields, u_s and ν_s), and their energy- and physics-informed parameters $\hat{\Pi}_s$. Essentially, $\hat{\Pi}_s$ comprises scalar damage field variable values, ϕ_s, von Mises stress values, σ_s, binary indicators of active/inactive nodes, $\mathcal{I}_s \in \{0, 1\}$ (where 0 indicates inactive mesh nodes and 1 denotes active mesh nodes), the Laplacian of the scalar damage field, $\Delta\phi_s$, and the applied displacement loading u_{0_s}.

$$\begin{aligned}
\hat{\mathcal{P}}_s &= \{(x_s, y_s)\} \qquad \{s \in \mathbf{U}\};\ \{\mathbf{U} \in \mathcal{M}^{ref}\},\\
\hat{\mathcal{A}}_s &= \{\mathcal{A}_s\} \qquad \{s \in \mathbf{U}\};\ \{\mathbf{U} \in \mathcal{M}^{ref}\},\\
\hat{\mathcal{D}}_s &= \{(u_s, \nu_s)\} \qquad \{s \in \mathbf{U}\};\ \{\mathbf{U} \in \mathcal{M}^{ref}\},\\
\hat{\Pi}_s &= \{(\phi_s, \sigma_s, \mathcal{I}_s, \Delta\phi_s, u_{0_s})\} \qquad \{s \in \mathbf{U}\};\ \{\mathbf{U} \in \mathcal{M}^{ref}\},\\
\{\xi_s\} &= \{(\hat{\mathcal{P}}_s, \hat{\mathcal{A}}_s, \hat{\mathcal{D}}_s, \hat{\Pi}_s)\} \qquad \{s \in \mathbf{U}\};\ \{\mathbf{U} \in \mathcal{M}^{ref}\}.
\end{aligned} \tag{5.2}$$

Moreover, each edge in $\mathcal{M}^{ref}$ is associated with a binary value indicating if the current "sender" node ξ_s, and all other active "receiver" nodes ξ_r in $\mathbf{E}$, belong to the same neighboring array $\hat{\mathcal{A}}_s$. This binary value is represented using $(s, r, b_{sr}) \in \mathbf{E}$, where s and r define the "sender" and "receiver" nodes, respectively (i.e., for any positive integer $s : \{1, 2, \ldots, \mathbf{U}\}$ and $r : \{1, 2, \ldots, \mathbf{U}\}$), and $b \in \{0, 1\}$. When $s = r$, we set $b_{sr} = 1$. Figure 5.2 illustrates an example of the graph representation, depicting active nodes (shown as green nodes) and their corresponding edges (shown as orange lines) in Figures 5.2b - 5.2c for node 16 and node 34,080, respectively. As shown in equation (5.3), this graph representation allows us to denote the indices of neighbors associated with each active mesh node.

$$\hat{\beta}_{sr} = \{(\xi_s, \xi_r, b_{sr})\} \qquad \{(s, r, b_{sr}) \in \mathbf{E}\};\ \{\mathbf{E} \in \mathcal{M}^{ref}\}. \tag{5.3}$$

When $b_{sr} = 1$, we then define the first five edge features as shown in equation (5.4). These include (i) the distances in the x- and y-directions between the sender and receiver nodes, $\delta\mathcal{X}_{sr} = (x_r - x_s)$ and $\delta\mathcal{Y}_{sr} = (y_r - y_s)$, (ii) the magnitude of the distance between the sender and receiver nodes, $\mathcal{L}_{sr} = \left(\sqrt{\delta\mathcal{X}_{sr}^2 + \delta\mathcal{Y}_{sr}^2}\right)$, (iii) the difference in the scalar damage field between the sender and receiver nodes, $\delta\phi_{sr} = (\phi_r - \phi_s)$, and (iv) the difference in the stress

field between the sender and receiver nodes, $\delta\sigma_{sr} = (\sigma_r - \sigma_s)$.

$$\begin{aligned} \delta\hat{\mathcal{P}}_{sr} &= \left\{ (\delta\mathcal{X}_{sr}, \delta\mathcal{Y}_{sr}, \mathcal{L}_{sr}) \right\} \qquad \{(s,r,b_{sr}) \in \mathbf{E}\};\ \{\mathbf{E} \in \mathcal{M}^{ref}\}, \\ \delta\hat{\Pi}_{sr} &= \left\{ (\delta\phi_{sr}, \delta\sigma_{sr}) \right\} \qquad \{(s,r,b_{sr}) \in \mathbf{E}\};\ \{\mathbf{E} \in \mathcal{M}^{ref}\}. \end{aligned} \tag{5.4}$$

We then leverage the energy functional from equation (5.1) in order to incorporate physics-based information into the ADAPT-GNN framework. The physics-based features include gradients of ϕ, and a term accounting for displacement and stress effects on the strain energy density. The three gradient edge features include (i) the gradient of the damage field, $\nabla\hat{\phi}sr$, (ii) the gradient of x- and y-displacements, $\nabla\hat{\mathcal{D}}sr$, and (iii) the gradient of stress, $\nabla\hat{\sigma}_{sr}$. The resulting edge features become

$$\begin{aligned} \nabla\hat{\phi}_{sr} &= \left\{ \left(\frac{\delta\phi_{sr}}{\delta\mathcal{X}} + \frac{\delta\phi_{sr}}{\delta\mathcal{Y}} \right) \right\} \qquad \{(s,r,b_{sr}) \in \mathbf{E}\}; \{\mathbf{E} \in \mathcal{M}^{ref}\} \\ \nabla\hat{\mathcal{D}}_{sr} &= \left\{ \left(\frac{\delta u_{sr}}{\delta\mathcal{X}} + \frac{\delta\nu_{sr}}{\delta\mathcal{X}} \right), \left(\frac{\delta u_{sr}}{\delta\mathcal{Y}} + \frac{\delta\nu_{sr}}{\delta\mathcal{Y}} \right) \right\} \qquad \{(s,r,b_{sr}) \in \mathbf{E}\}; \{\mathbf{E} \in \mathcal{M}^{ref}\} \\ \nabla\hat{\sigma}_{sr} &= \left\{ \left(\frac{\delta\sigma_{sr}}{\delta\mathcal{X}} + \frac{\delta\sigma_{sr}}{\delta\mathcal{Y}} \right) \right\} \qquad \{(s,r,b_{sr}) \in \mathbf{E}\}; \{\mathbf{E} \in \mathcal{M}^{ref}\} \\ \{e_{sr}\} &= \left\{ \left(\hat{\beta_{sr}}, \delta\hat{\mathcal{P}}_{sr}, \delta\hat{\Pi}_{sr}, \nabla\hat{\phi}_{sr}, \nabla\hat{\mathcal{D}}_{sr}, \nabla\hat{\sigma}_{sr} \right) \right\} \qquad \{(s,r,b_{sr}) \in \mathbf{E}\}; \{\mathbf{E} \in \mathcal{M}^{ref}\}. \end{aligned} \tag{5.5}$$

5.2.3 Spatial Message-Passing Process

For each GNN, The ADAPT-GNN framework utilizes the Graph Isomorphism Network with Edge Features (GINE) from [206] as the message-passing model. The GINE message-passing network comprises operations aim to map the input nodes and edge features to the latent space. The resulting latent space embedding is then fed into an MLP with Rectified Linear Unit (ReLU) activation function. As defined in equations (5.2) - (5.5), the input to the message-passing network consists of the node and edge features, for the current time-step.

The message-passing outputs are represented as $\xi_s^{'}, e_{sr}^{'}$, capturing the transformed vertex and edge attributes in the latent space for the current time-step t, respectively. It's essential to

note that for each GNN module within the framework (e.g., *XDisp*-GNN, *YDisp*-GNN, and *cPhi*-GNN), meticulous adjustments to the message-passing network are imperative to maintain high accuracy and preserve the relational information within the graphs [176, 257, 258]. Consequently, we fine-tuned the message-passing network of each GNN by varying parameters such as the number of message-passing steps (i.e., iterations for vertices and edges through the encoder networks) and the hidden layer dimensions. The detailed procedures for optimizing the GINE encoders and the resulting enhancements are presented in Section 5.4.

5.2.4 Training-set and Validation-set

In accordance with Section 5.2.1, the dataset utilized for training, validation, and testing was generated through the implementation of the second-order phase field fracture model as detailed in [211]. It is notable that the fourth-order phase field model, also provided in [211], could potentially yield more precise outcomes. However, we opted for the second-order approach to balance computational efficiency and proof of concept. The configuration consisted of a domain measuring 0.5 m by 0.5 m, featuring a maximum of 193 by 193 mesh nodes. The domain encompassed a single edge crack subjected to tensile displacement loading. Material properties were simulated under isotropic and homogeneous conditions, with specified values: Young's Modulus, $E = 210$ N/mm^2, Poisson's ratio, $\nu = 0.3$, critical energy release rate, $G_{1c} = 2.7$, and length scale parameter, $l_0 = 0.0125$ m. We note that dynamic effects, such as crack-tip bifurcation, were not considered in our analysis. Analogous to the scenario delineated in [211], we fixed the lower boundary of the domain and applied incremental displacement loads, perpendicular to the top edge in the positive y-direction (representing tensile loading). Specifically, the load configuration was $\Delta u = 1 \times 10^{-4}$ mm for the initial 45 displacement steps, transitioning to $\Delta u = 1 \times 10^{-6}$ mm for subsequent steps to mitigate dynamic effects.

We generated a comprehensive dataset comprising 1245 distinct simulations, achieved by varying the initial crack length, edge position, and crack angle. Specifically, the crack lengths (C_L), edge positions (C_P), and crack angles (C_θ) were systematically altered across ranges: $C_L : 0.05, 0.10, \ldots, 0.45$ m, $C_P : 0.1, 0.15, \ldots, 0.4$ m, and $C_\theta : -65^o, -60^o, \ldots, 65^o$, respectively. For visualization purposes, refer to Figure 5.2a, illustrating the problem setup

along with the varied configurations of C_L, C_P, and C_θ. Additionally, to ensure consistency, cases resulting in crack-tip locations extending beyond the domain's boundaries were excluded. Notably, each simulation encompasses a range of 100 to 450 time-steps, with each input to the GNN framework representing a single time frame. Consequently, the dataset's size spans from $124,500$ to $560,250$ instances. Additionally, we removed cases resulting in crack-tip locations beyond the domain's bounds. We note that each simulation contains 100 to 450 time-steps, and each input to the GNN framework involved a single time frame, thus, resulting in a dataset size of $124,500$ to $560,250$.

To conduct a rigorous error analysis on the test set, we employed a systematic approach. We first randomly curated a subset of 30 simulations for the test set, ensuring an equitable distribution of cases with positive ($C_\theta \geq 0^o$) versus negative ($C_\theta < 0^o$) crack angles, top ($C_P \geq 0.5$ m) versus bottom ($C_P < 0.5$ m) initial edge positions, and large ($C_L \geq 0.25$ m) versus small ($C_L < 0.25$ m) initial crack lengths. The remaining simulations were then allocated with 1100 for the training set and 115 for the validation set. We organized the training set into shuffled batches sized 32, while maintaining the validation set in a sequential order with a batch size of 1. Lastly, each model (*XDisp*-GNN, *YDisp*-GNN, and *cPhi*-GNN) underwent training for a total of 20 epochs to ensure robust learning and convergence.

5.3 Adaptive Mesh-based Graph Neural Network

As illustrated in Figure 5.1, the *ADAPT*-GNN framework encompasses three primary GNNs: (i) *XDisp*-GNN, (ii) *YDisp*-GNN, and (iii) *cPhi*-GNN. ADAPT-GNN starts by utilizing the node and edge features (from equations (5.2) - (5.5)) at the current time-step t, in order to predict the displacement fields at future time-step $t+1$. Subsequently, it uses the node and edge features at time t (from equations (5.2) - (5.5)) alongside the predicted displacements fields at $t+1$ to predict the scalar damage field ϕ at time $t+1$. A distinctive aspect of *ADAPT*-GNN is its integration of AMR. This feature enables the framework to dynamically expand the graph's size at each time-step by utilizing the instantaneous refined mesh as the graph representation itself. Additionally, it harnesses both the GPU-usage and order reduction offered by ML techniques, as well as the computational efficiency provided by the AMR approach. Thus, reducing computational

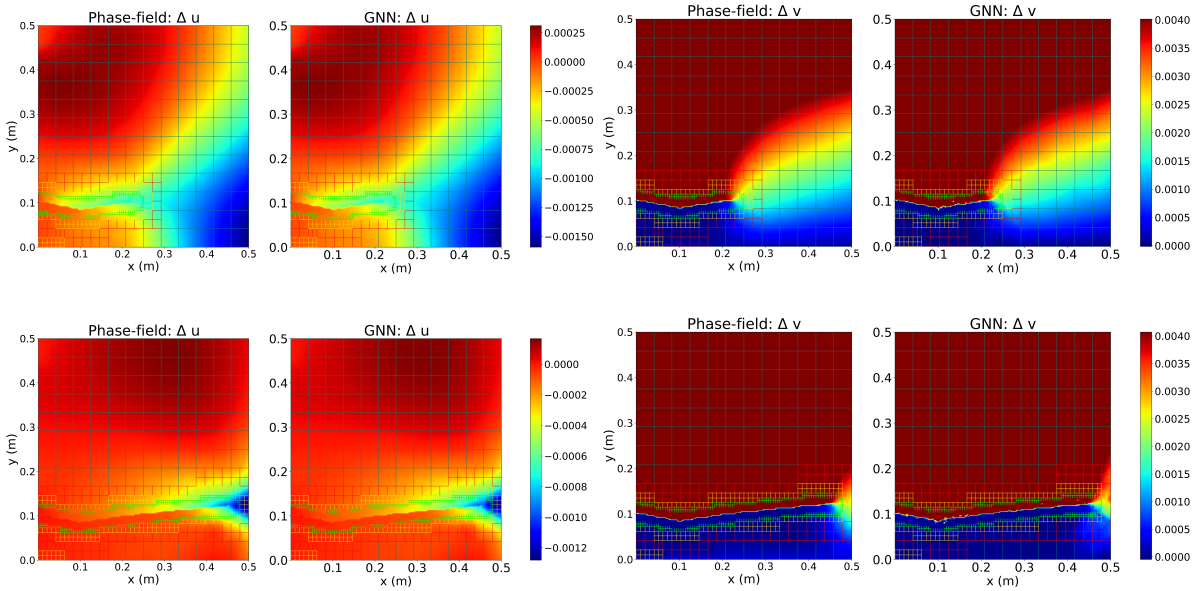
demands while enhancing simulation speed. The next sections will present the implementation of the "Prediction Step" (*XDisp*-GNN, *YDisp*-GNN, and *cPhi*-GNN), and the "AMR Update" depicted in Figure 5.1 in greater detail.

5.3.1 *XDisp*-GNN and *YDisp*-GNN

We developed the *XDisp*-GNN and *YDisp*-GNN modules to predict the displacements in the x and y directions, respectively, for each node within the refined mesh. Initially, we constructed the input graph representation based on the methodology outlined in Section 5.2.2. Following the detailed process in Section 5.2.3, both *XDisp*-GNN and *YDisp*-GNN employ a GINE message-passing model to encode the node and edge information into latent space embeddings. To predict displacements at future time-steps, we utilized the latent space representations generated by the message-passing networks of *XDisp*-GNN and *YDisp*-GNN as inputs to two distinct Attention Temporal Graph Convolutional Networks (ATGCN) [259]. The ATGCN architecture is tailored to learn both local and global spatiotemporal patterns in state evolution. It begins with a Temporal Graph Convolutional Network (T-GCN) [260], which integrates Graph Convolutional Networks (GCNs) and Gated Recurrent Units (GRUs) in a sequential manner to capture local trends. Furthermore, the ATGCN introduces a tanh activation function (attention mechanism) to dynamically adjust the influence of historical states, enabling the capture of global variation trends.

We opted for the ATGCN model because of its unique combination of gated recurrent units within graph convolutional networks, facilitating the learning of temporal changes while preserving spatial relations within the graphs. Additionally, its integration of an attention mechanism enables the capture of both local and global spatiotemporal variations. The input graph at a specific time-step, t, and the corresponding predicted displacements at the subsequent time-step, $t+1$, generated by the ATGCNs can be described as follows:

$$(\hat{u}_s, \hat{\nu}_s)^{t+1} \longleftarrow ATGCN\left[\{\xi_s^{'}, e_{sr}^{'}\}^t\right] \qquad \{s \in \mathbf{V}\}; \{(s, r, b_{sr}) \in \mathbf{E}\}. \tag{5.6}$$



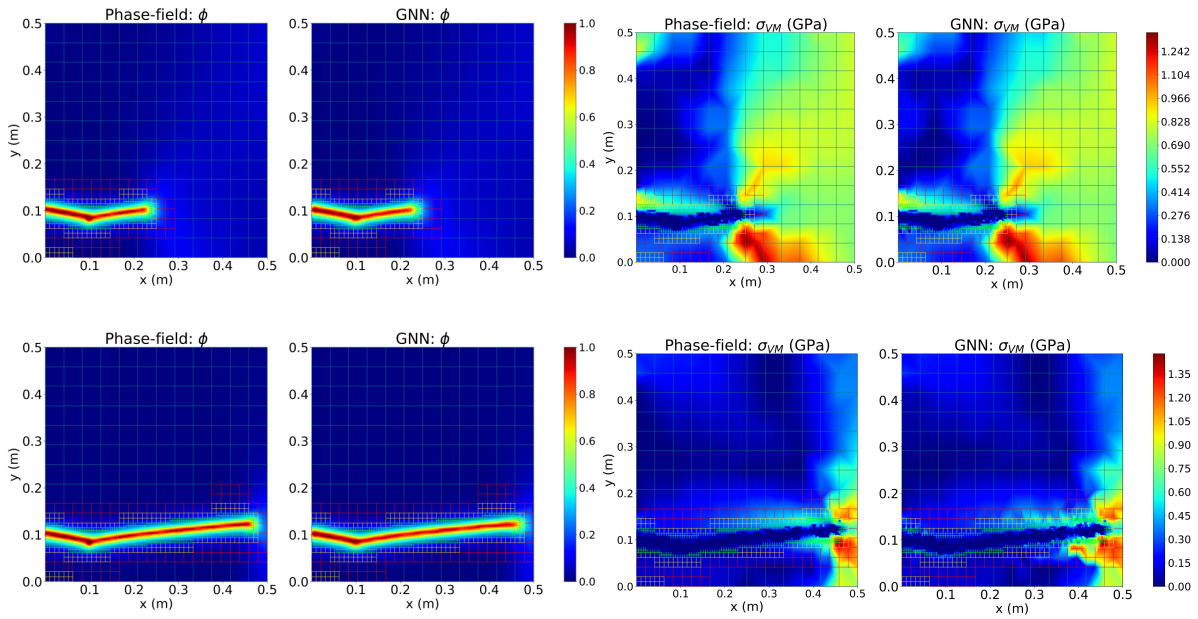
(a) PF versus *XDisp*-GNN (b) PF vs. *YDisp*-GNN

Figure 5.3: a) Comparison between the PF fracture model and the *XDisp*-GNN prediction for a simulation from the test set featuring a small crack ($C_L = 0.1$ m) with a negative angle and positioned at $C_P = 0.1$ m. b) Comparison between the PF fracture model and the *YDisp*-GNN prediction for the same simulation scenario described above.

Equation (5.6) shows a single ATGCN utilized for both *XDisp*-GNN and *YDisp*-GNN for the sake of simplicity. We note that each GNN incorporates a GINE message-passing network followed by an ATGCN model. Furthermore, to train *XDisp*-GNN, we employed the SmoothL1Loss function since Δu values exhibit variations from negative to positive, while for *YDisp*-GNN, we utilized the MSELoss function as Δv values are non-negative. For both GNNs, we employed the Adam optimizer from [261]. Essentially, the ATGCNs use the current time-step (t) features of vertices and edges in the latent space, for predicting the x- and y-displacements at the future time-step, $t + 1$. Figure 5.3 presents a comparison between the PF model and the predicted x- and y-displacements.

5.3.2 *cPhi*-GNN

Following the prediction of x- and y-displacements, we concatenated them and utilized the combined output as input to *cPhi*-GNN. As previously discussed, the primary objective of *cPhi*-GNN is to forecast the crack field, ϕ_s, at future time-steps. Similar to the *XDisp*-GNN and *YDisp*-GNN models, *cPhi*-GNN utilizes the features of vertices and edges in the latent space as



(a) PF vs. *cPhi*-GNN (b) PF vs. σ-GNN

Figure 5.4: a) Comparison between the PF fracture model and the *cPhi*-GNN prediction for the same test case scenario depicted in Figure 5.3, which involves a small crack ($C_L = 0.1$ m) with a negative angle located at $C_P = 0.1$ m. b) Comparison between the PF fracture model and the σ_{VM} prediction for the identical test case scenario described above.

the first portion of its input. We further augment this input by concatenating the predicted x- and y-displacements at future time-steps. Consequently, we modified the ATGCN used for predicting ϕ and incorporated two additional vertex and edge features representing the previously predicted displacements. Like *XDisp*-GNN and *YDisp*-GNN, *cPhi*-GNN underwent training using the Adam optimizer [261]. We define the architecture of *cPhi*-GNN as:

$$\left(\hat{\phi}_s\right)^{t+1} \longleftarrow ATGCN\left[\{\xi_s^{'}, e_{sr}^{'}\}^t, (\hat{u}_s, \hat{\nu}_s)^{t+1}\right] \qquad \{s \in \mathbf{V}\}; \{(s, r, b_{sr}) \in \mathbf{E}\}. \tag{5.7}$$

One notable feature of the *ADAPT*-GNN framework, is its ability to forecast the evolution of stress, utilizing the predicted x- and y-displacements, along with ϕ. Derived from the second-order PF fracture model in [211], the stress tensor is defined as $\underline{\sigma} = (1-\phi)^2\left[\lambda \text{tr}(\underline{\varepsilon})\mathbf{I} + 2\mu\underline{\varepsilon}\right]$. Where $\underline{\varepsilon}$ denotes the strain tensor, and λ and μ represent the Lam'e constants. Utilizing this formulation alongside the predictions from the GNN framework, we calculate the stress evolution.

We compare the PF fracture model and the predicted ϕ, alongside the predicted von Mises stress σ_{VM} in Figure 5.4.

5.3.3 AMR Update

The "AMR Update" step stands as the cornerstone within the *ADAPT*-GNN framework, pivotal for its efficacy. It's imperative to underscore the indispensable role of AMR in augmenting the performance of ADAPT-GNN. The conventional approach of employing a static fine mesh as the graph representation invariably leads to a large number of edge connections. This large number of edges stems from the dependence of a point's solution on distant points. Such an arrangement inevitably results in large graphs leading to increased computational costs. However, a potential avenue for mitigation of large graphs lies in increasing the number of message-passing steps to align with the required hop distance. A study by Hamilton et al. [262] demonstrates that to facilitate information transfer between two nodes separated by "x" hops, the GNN necessitates "x" message-passing blocks. Yet, the exigencies of achieving finer mesh resolution in our context demand an exceedingly large number of hops, rendering the conventional approach untenable due to its impracticality in handling an overwhelming number of message-passing steps.

To harness the combined capabilities of AMR and GNNs, post the completion of the Prediction Step depicted in Figure 5.1 involving *XDisp*-GNN, *YDisp*-GNN, and *cPhi*-GNN, we implement mesh refinement. This entails the addition of new nodes in regions exhibiting a scalar damage field (ϕ) surpassing a predefined threshold value (set at 0.5, following [211]). Moreover, we then develop a new graph representation for the next time-step by introducing new vertices and edges for the resulting new refined mesh.

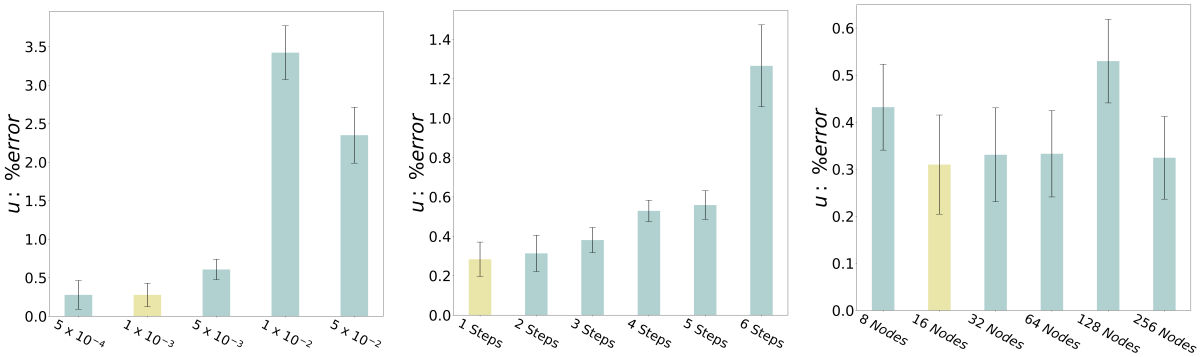
Throughout the training process, we employ a Boolean mask array to explicitly train the active nodes while disregarding the inactive ones. As illustrated in Figure 5.2, this methodology ensures the dynamism of the graph, where edges are exclusively established between adjacent active nodes. Consequently, training computations are confined solely to the active nodes. Finally, upon the generation of the new refined graph representation for the subsequent time-step, we iteratively execute this procedure until failure propagation permeates the entire domain.

5.4 Cross-validation

To optimize the framework further, we conducted cross-validation on *XDisp*-GNN, *YDisp*-GNN, and *cPhi*-GNN using the 10-fold (k-fold) cross-validation methodology [263]. This approach involved examining various training parameters such as learning rates, the number of message-passing steps, and the number of nodes in the hidden layers of the MLP network. Initially, we partitioned the original training dataset (comprising 1100 simulations) into 10 distinct groups and shuffled them. Next, one group was designated as the 'new validation dataset,' while training was conducted on the remaining 9 groups, serving as the 'new training dataset'. This training process was executed for 5 epochs before selecting another combination of validation and training groups. This procedure was repeated for each GNN model and for each investigated training parameter. Evaluation of performance was based on the averaged maximum percentage errors in the predicted x-displacements, y-displacements, and ϕ field. It's essential to note that cross-validation was exclusively applied to the message-passing GINE models associated with each GNN (*XDisp*-GNN, *YDisp*-GNN, and *cPhi*-GNN). As for the ATGCN models of each GNN, the only trainable parameter was the filter size (or number of nodes). Therefore, cross-validation was not implemented for the ATGCN models in this study. The filter size for each ATGCN was selected to align with the optimal number of hidden layer nodes derived from the cross-validation of the GINE message-passing models.

5.4.1 *XDisp*-GNN cross-validation

The averaged percent errors for x-displacement predictions are depicted in Figure 5.5. In Figure 5.5a, learning rates of 5×10^{-4}, 5×10^{-3}, 1×10^{-2}, and 5×10^{-2} (light blue) resulted in higher errors for *XDisp*-GNN compared to the learning rate of 1×10^{-3} (yellow). Notably, the highest error of $3.42 \pm 0.35\%$ occurred for the learning rate of 1×10^{-2}, contrasting with the smaller learning rate of 1×10^{-3} showing an error of $0.28 \pm 0.15\%$. Consequently, for the *XDisp*-GNN model we selected the optimal learning rate of 1×10^{-3}.



(a) Learning rates: u (b) Message-passing steps: u (c) Hidden layer nodes: u

Figure 5.5: a) Cross-validation results for the *XDisp*-GNN model: Different learning rates were tested, including 5 $\times 10^{-4}$, 5 $\times 10^{-3}$, 1 $\times 10^{-2}$, and 5 $\times 10^{-2}$, represented in light blue, while our model's learning rate of 1 $\times 10^{-3}$ is highlighted in yellow. b) Cross-validation results for the *XDisp*-GNN model: Various message-passing steps were evaluated, ranging from 2 to 6, indicated in light blue, with our model's message-passing steps set to 1, highlighted in yellow. c) Cross-validation results for the *XDisp*-GNN model: Different numbers of hidden layer nodes were tested, including 8, 32, 64, 128, and 256, represented in light blue, while our model's hidden layer nodes were set to 16 and highlighted in yellow.

Next, Figure 5.5b displays the averaged percent errors for message-passing steps ranging from 1 to 6. Notably, the model achieved the lowest percent error of $0.28 \pm 0.09\%$ with message-passing steps of $M = 1$, compared to an error of $1.27 \pm 0.21\%$ for the highest number of message-passing steps of $M = 6$. Thus, we opted for the optimal parameter of $M = 1$ step to enhance the *XDisp*-GNN model. We note that for less message-passing steps the required training time and simulation is decreased.

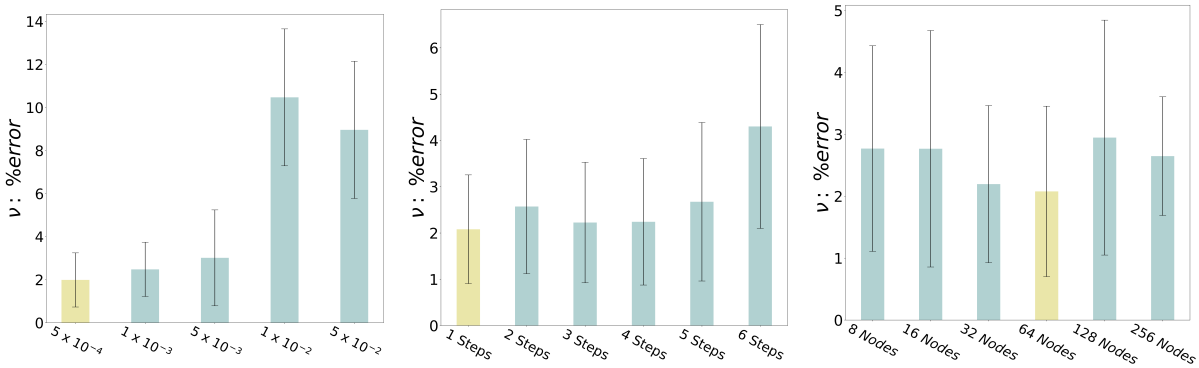
Lastly, as illustrated in Figure 5.5c, we evaluated the number of nodes in the hidden layers of the MLP network. Remarkably, employing 16 nodes yielded the least error at $0.31 \pm 0.11\%$, compared to the highest error of $0.53 \pm 0.09\%$ with 128 nodes for *XDisp*-GNN. It's also worth noting that higher node counts entail increased computational requirements. Consequently, we selected a filter size of 16 for the ATGCN model in *XDisp*-GNN, aligning with the optimal hidden layer nodes count obtained for *XDisp*-GNN.

5.4.2 Cross-validation for *YDisp*-GNN

As illustrated in Figure 5.6, The process of cross-validation for *YDisp*-GNN involved evaluating the averaged percent errors for y-displacement predictions across various learning rates, message-passing steps, and numbers of nodes. In Figure 5.6a, the learning rate of 5×10^{-4} (yellow) yielded lower error compared to the learning rates of 1×10^{-3}, 5×10^{-3}, 1×10^{-2}, and 5×10^{-2} (light blue). For the learning rate of 5×10^{-4} the error was $1.99 \pm 1.26\%$, while the highest error of $10.47 \pm 3.18\%$ was obtained for learning rate of 1×10^{-2}. Additionally, Figure 5.6b illustrates that the model achieved the lowest percent error with a single message-passing step ($M = 1$) at $2.08 \pm 1.17\%$, while the highest number of message-passing steps ($M = 6$) resulted in the highest error at $4.30 \pm 2.20\%$. Lastly, the cross-validation for the number of nodes is shown in Figure 5.6c. Using 64 nodes, *YDisp*-GNN resulted in the lowest error of $2.08 \pm 1.38\%$, while the highest error of $2.95 \pm 1.90\%$ was obtained for the case of 128 nodes. We emphasize that selecting the appropriate learning rate proved critical for achieving higher accuracy in both *XDisp*-GNN and *YDisp*-GNN compared to the number of message-passing steps and number of hidden layer nodes. Therefore, we opted for a filter size of 64 for the ATGCN model in *YDisp*-GNN, aligning with the optimal number of hidden layer nodes for *YDisp*-GNN's message-passing GINE model.

5.4.3 Cross-validation for *cPhi*-GNN

As depicted in Figure 5.7, the final stage of the cross-validation process involved the *cPhi*-GNN model. Similar to *YDisp*-GNN, the lowest error of $0.33 \pm 0.12\%$ was found for the optimal learning rate of 5×10^{-4} (depicted in yellow). The highest error of $6.86 \pm 0.81\%$ occurred with a learning rate of 5×10^{-2}. Regarding the number of message-passing steps, Figure 5.7b indicates that $M = 6$ resulted in the highest error at $0.54 \pm 0.10\%$, whereas the lowest error was observed for $M = 4$ at $0.33 \pm 0.04\%$. Moreover, considering the number of hidden layer nodes, the configuration with 32 nodes exhibited the lowest percent error of $0.36 \pm 0.06\%$, while 16 nodes yielded the highest percent error of $0.53 \pm 0.08\%$. Consequently, similar to *XDisp*- and



(a) Learning rates: ν (b) Message-passing steps: ν (c) Hidden layer nodes: ν

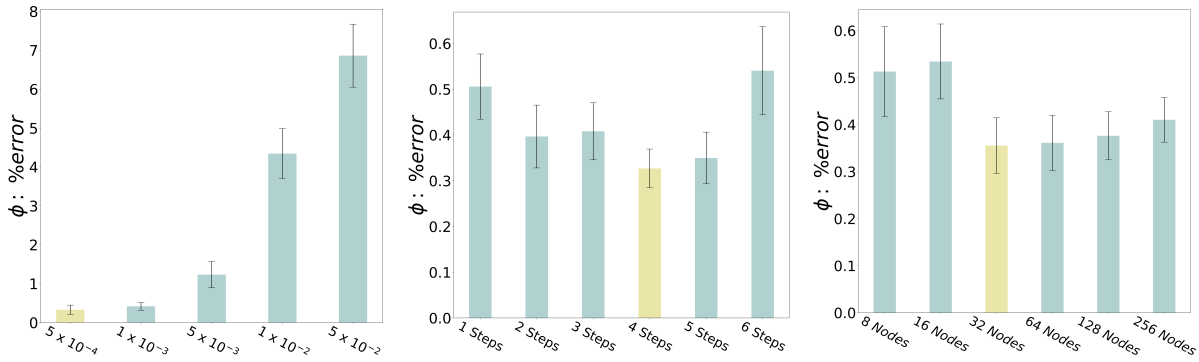
Figure 5.6: a) Cross-validation results for the *YDisp*-GNN model: Various learning rates were tested, including 1 $\times 10^{-4}$, 5 $\times 10^{-3}$, 1 $\times 10^{-2}$, and 5 $\times 10^{-2}$, represented in light blue, while our model's learning rate of 5 $\times 10^{-4}$ is highlighted in yellow. b) Cross-validation results for the *YDisp*-GNN model: Different numbers of message-passing steps were evaluated, ranging from 2 to 6, indicated in light blue, with our model's message-passing steps set to 1 and highlighted in yellow. c) Cross-validation results for the *YDisp*-GNN model: Various numbers of hidden layer nodes were tested, including 8, 16, 32, 128, and 256, represented in light blue, while our model's hidden layer nodes were set to 64 and highlighted in yellow.

YDisp-GNN, we opted for a filter size of 32 for the ATGCN model in *cPhi*-GNN, aligning with the optimal number of hidden layer nodes for *cPhi*-GNN's message-passing GINE model.

5.5 Results

5.5.1 ADAPT-GNN prediction of displacements, crack field and stresses

First, we showcase the framework's proficiency in forecasting the evolution of the scalar damage field ϕ, x-displacements Δu, y-displacements $\Delta \nu$, and von Mises stress σ_{VM} for a crack configuration from the test dataset characterized by a positive crack angle with a large crack length and positioned at the bottom edge. Figure 5.8 provides a comparative analysis between PF and *ADAPT*-GNN regarding the evolution of ϕ. We underscore that the results depicted in Figures 5.8 and 5.12 involve both the time evolution along with the resulting errors for the scalar damage field and displacement fields. These were acquired by cascading ADAPT-GNN from t_0 to T_f, implying that predictions from prior time-steps serve as inputs for subsequent ones. We also highlight that during t_1 to t_{33}, the kinking of the predicted crack path was not as sharp as in the PF model. Furthermore, we acknowledge the oscillations within the crack field of the PF model, attributable to nuances in the second-order model and implementation errors.



(a) Learning rates: ϕ (b) Message-passing steps: ϕ (c) Hidden layer nodes: ϕ

Figure 5.7: a) For the *cPhi*-GNN model, cross-validation was performed with various learning rates: 1×10^{-4}, 5×10^{-3}, 1×10^{-2}, and 5×10^{-2}, represented in light blue. Our model's learning rate was determined to be 5×10^{-4} and is highlighted in yellow. b) The cross-validation results also involved testing different numbers of message-passing steps, ranging from 1 to 6, shown in light blue. Our model's message-passing steps were set to 4 and are highlighted in yellow. c) Additionally, the cross-validation process considered various numbers of hidden layer nodes: 8, 16, 64, 128, and 256, depicted in light blue. Our model's hidden layer nodes were determined to be 32 and are highlighted in yellow.

These discrepancies might transfer to *ADAPT*-GNN's predictions. Despite these intricacies, our results demonstrate a strikingly similar overall prediction of the crack path compared to the PF fracture model throughout the simulation. In summary, these qualitative findings underscore the efficacy of the developed GNN in accurately predicting the scalar damage field's evolution, thereby attesting to its robustness and reliability in fracture modeling scenarios.

Based on the same test case scenario illustrated in Figure 5.8, Figure 5.9 presents a comparative analysis between PF and *ADAPT*-GNN for predictions of x- and y-displacements, as well as von Mises stresses during t_{50}. Upon inspection of the x-displacements, it's evident that the predicted field closely mirrors that of the PF fracture model, showcasing a remarkable similarity. However, a discernible deviation emerges when scrutinizing the y-displacements. Here, a notable prediction error becomes apparent within the sharp interface of positive to negative y-displacements within the crack region. It's important to note that in PF fracture models, the y-displacement exhibits a sharp transition within the crack, from negative to positive values. Therefore, errors within the crack region are less consequential in PF fracture models. Furthermore, the plots in Figure 5.9 were generated utilizing the "*tricontourf*" function, which interpolates between active mesh points using the y-displacement values. Consequently, the

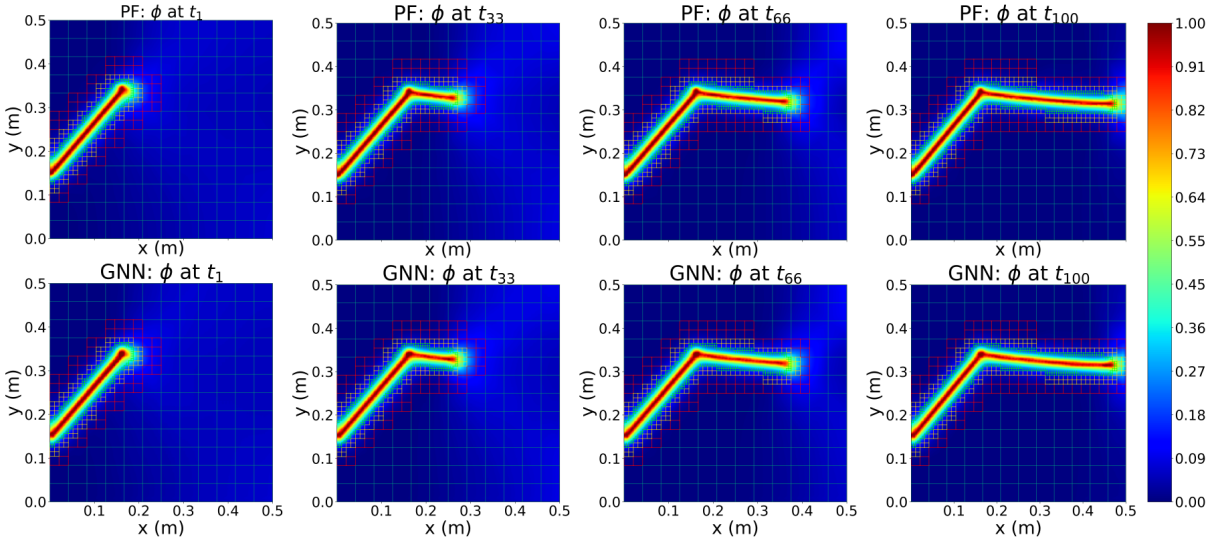


Figure 5.8: PF fracture model compared with *ADAPT*-GNN regarding the evolution of the scalar damage field, ϕ, for a crack configuration from the test dataset. This configuration features a positive crack angle with a large crack size ($C_L = 0.25$ m) and bottom edge position ($C_P = 0.15$ m).

highest y-displacement errors originating within the crack are interpolated to regions outside the crack. Hence, Figure 5.9 underscores the framework's adeptness in predicting y-displacements with good accuracy in regions outside the crack.

Finally, Figure 5.5.2 highlights a pivotal aspect of the developed ADAPT-GNN framework: its capacity to also simulate the stress evolution. The predicted displacement fields, and scalar damage field can be leveraged to compute the resulting von Mises stresses in the material. Notably, there exists a strong qualitative agreement between the von Mises stresses derived from the PF model, and those obtained using *ADAPT*-GNN predictions. As a result, Figures 5.8 and 5.9 highlight the framework's efficacy in simulating displacements, scalar damage field, and von Mises stress with good accuracy. Additionally, as supplementary material, animations of seven test cases have been included in [5], offering further insight into the framework's predictive capabilities.

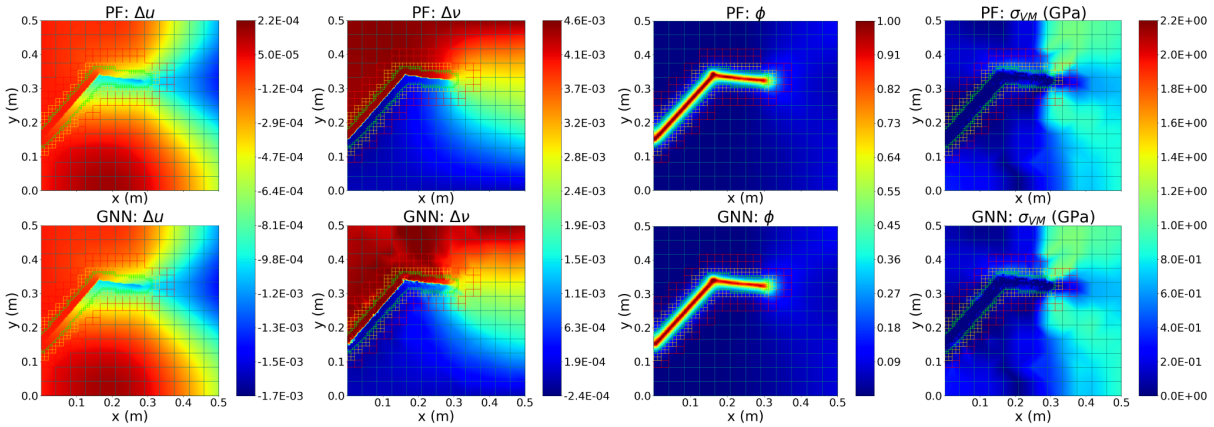


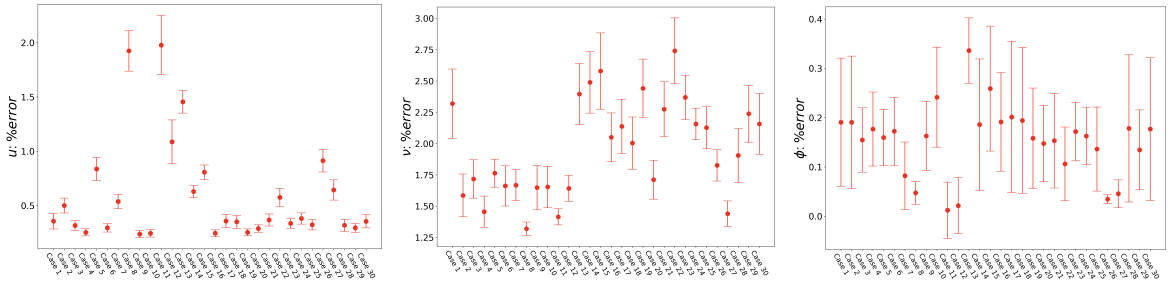
Figure 5.9: Comparison between the PF fracture model and *ADAPT*-GNN for predicting x-displacements, Δu, y-displacements, $\Delta \nu$, scalar damage field, ϕ, and computed von Mises stress, σ_{VM} for the same test case scenario depicted in Figure 5.8. This scenario involves a positive crack angle with a large crack size ($C_L = 0.25$ m) and bottom edge position ($C_P = 0.15$ m).

5.5.2 Prediction errors

Here, we comnputed the maximum $\%$ errors in order to gather the errors generated by the *XDisp*-GNN, *YDisp*-GNN, and *cPhi*-GNN models using

$$\%error = max\left[\Sigma_{i=1}^{\mathcal{M}} \frac{1}{\mathcal{M}} \left(\frac{|\phi_{pred}(t,i) - \phi_{true}(t,i)|}{\phi_{true}(t,i)}\right) \times 100\right] \qquad \ldots \{t \in T_f\}. \tag{5.8}$$

ϕ_{true} and ϕ_{pred} denote the true, and the predicted scalar damage fields, respectively, and T_f denotes the final time for complete fracture. We note that while equation (5.8) is utilized to quantify errors in ϕ, it is also applicable for computing errors in the displacement fields. The maximum percent errors of *XDisp*-GNN, *YDisp*-GNN, and *cPhi*-GNN are visually represented in Figures 5.10a, 5.10b, and 5.10c, respectively. The outcomes presented in Figure 5.10 illustrate the maximum percent error derived such that the predictions from the previous time-steps are employed as input to the subsequent time-step. As described in Section 5.5 and shown in Figure 5.8, the PF simulator adopted in this study is susceptible to instability errors due to oscillations in the scalar damage field within the crack region. Since these errors are located in the active nodes of the refined mesh within the crack, the computation of errors at these nodes may exhibit inconsistencies compared to the remaining nodes outside the crack. Moreover, because *ADAPT*-GNN provides predictions for all mesh points in $\mathcal{M}$, for each time-step in the



(a) Percent Errors in u (b) Percent Errors in ν (c) Percent Errors in ϕ

Figure 5.10: Maximum percentage errors across time for each simulation in the test set (Case 1 - Case 30) for: a) Predicted u. b) Predicted ν. c) Predicted ϕ.

simulations, we initially obtain the average percent error for all mesh points. As demonstrated in equation (5.8), we then select the maximum percent error across time for each simulation. This meticulous error analysis procedure ensures that the generated errors are effectively captured across all nodes in $\mathcal{M}$. As depicted in Figure 5.10a, we identify the test case exhibiting the highest $\%$ error in the predicted x-displacement field, u, as $1.98 \pm 0.27\%$ error for Case 11. In contrast, the lowest $\%$ error is attributed to Case 9, with a value of $0.24 \pm 0.33\%$. Examining Figure 5.10b, for y-displacements, Case 22 exhibits the highest $\%$ error at $2.74 \pm 0.26\%$, while the most accurate prediction is observed in Case 8, achieving $1.32 \pm 0.05\%$ error. Similarly, in terms of the scalar damage field, ϕ, the maximum prediction error is encountered in Case 13, standing at $0.19 \pm 0.13\%$, while lowest error is seen in Case 3 of $0.01 \pm 0.06\%$. These findings underscore the efficacy of *ADAPT*-GNN in accurately forecasting displacements and crack propagation. Despite *YDisp*-GNN exhibiting the highest error among the three implemented GNNs, a maximum $\%$ error of $2.74 \pm 0.26\%$ translates to micrometers in a $0.5m \times 0.5m$ domain, underscoring its significant accuracy.

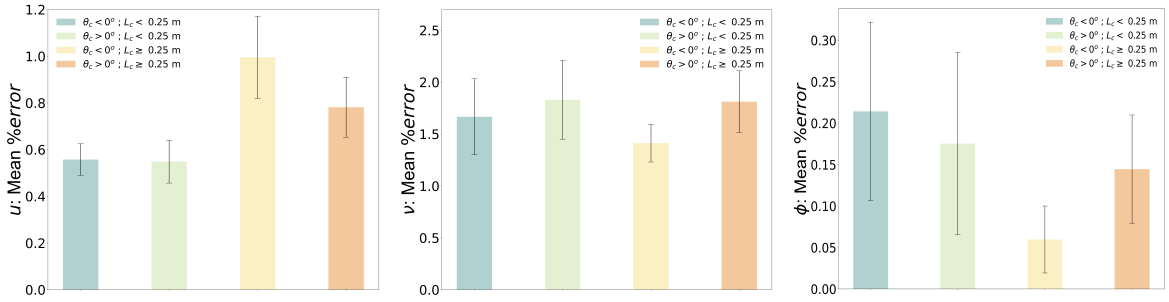
5.5.3 Parametric error analysis of initial crack length and crack orientation

To investigate the influence of initial crack lengths and orientations on the prediction errors, we partitioned the test dataset into four categories: (i) $\theta_c < 0^o$; $L_c < 0.25$ m for small crack length + negative crack orientation, (ii) $\theta_c > 0^o$; $L_c < 0.25$ m for small crack length + positive crack orientation, (iii) $\theta_c < 0^o$; $L_c \geq 0.25$ m for large crack length + negative crack orientation, and (iv) $\theta_c > 0^o$; $L_c \geq 0.25$ m for large crack length + positive crack orientation. Subsequently,

we calculated the mean and standard deviation of the maximum percent errors of each group. For each of these parametric groups, the resultant x-displacement, y-displacement, and scalar damage field errors are shown in Figure 5.11a, Figure 5.11b, and Figure 5.11c, respectively.

First, from Figure 5.11a we show that *XDisp*-GNN exhibits a noticeable distinction in the errors concerning small versus large crack lengths. For cracks of smaller length, the initial crack orientation does not exert a significant influence on the percent error. For example, the errors for small initial crack lengths involving positive and negative initial orientations are $0.55 \pm 0.09\%$ and $0.56 \pm 0.07\%$, respectively, differing only by approximately 0.01%. However, as the crack length increases beyond 0.24 m, the errors escalate. Moreover, for cases involving large crack lengths, the initial crack orientation does impact *XDisp*-GNN's prediction accuracy. The highest error was obtained for the group with negative orientations and large crack lengths at $1.00 \pm 0.18\%$ occurs. In contrast, for large crack lengths with positive crack orientations, the error diminished to $0.78 \pm 0.13\%$. Section 5.5.2 describes a plausible justification for why the groups involving smaller crack lengths yield lower errors. The highest mesh-wise errors for x-displacements were observed during the first time-steps. At the initial time-steps, the applied displacement load steadily increases until the crack smoothly propagates. As the crack begins to propagate smoothly, the errors then diminish throughout the remaining time-steps. Consequently, a smaller crack necessitates more time-steps for complete propagation across the domain. Essentially, smaller cracks entail more time-steps where crack propagation is smooth, resulting in lower errors compared to larger cracks. Therefore, regardless of the initial crack orientation, *XDisp*-GNN excels in configurations with smaller crack lengths, whereas for larger cracks, it achieves greater accuracy for configurations involving positive orientations.

Next, in Figure 5.11b we present the resulting parametric analysis regarding the effects of initial crack length and crack orientation for *YDisp*-GNN. Unlike *XDisp*-GNN, the highest error for *YDisp*-GNN was obtained for cases involving smaller crack lengths and positive crack orientations. These cases resulted in errors of $1.83 \pm 0.38\%$. In contrast, the lowest error for *YDisp*-GNN of $1.41 \pm 0.18\%$, was obtained for cases with larger crack lengths and negative crack orientation. Notably, both groups with positive crack orientations exhibited similar errors. For instance, the error difference between cases involving larger cracks + positive orientations,



(a) Percent Errors in u (b) Percent Errors in ν (c) Percent Errors in ϕ

Figure 5.11: Parametric error analysis was conducted to assess the contribution of initial crack angles and crack lengths on: a) Predicted u. b) Predicted ν. c) Predicted ϕ.

and smaller cracks + positive orientations was approximately 0.02%. However, for the groups with negative crack orientations, the optimal performance was observed when for larger cracks.

Lastly, as depicted in Figure 5.11c for *cPhi*-GNN, we note that the group with the highest error corresponded to smaller crack lengths with negative crack orientations, yielding $0.21 \pm 0.11\%$. The lowest error for *cPhi*-GNN was observed as $0.06 \pm 0.04\%$ for cases featuring larger crack lengths with negative crack orientations.

5.5.4 Parametric error analysis of initial edge position and crack orientation

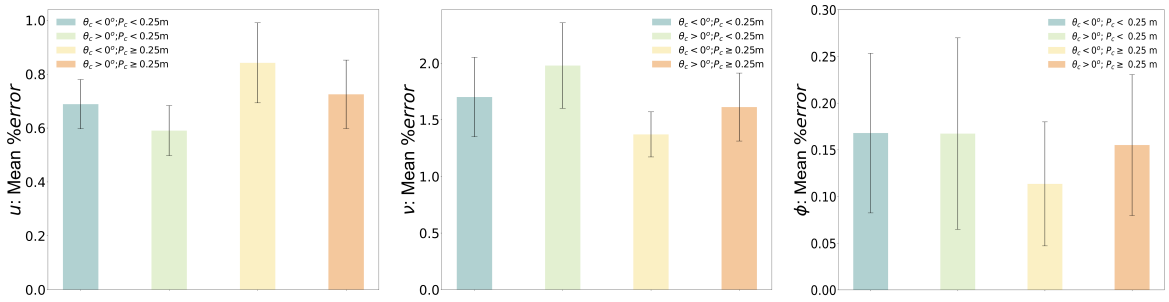
To examine the impact of varying initial edge positions and crack orientations on prediction errors, we employ a similar approach as detailed in Section 5.5.3. We divide the test dataset into four distinct groups: (i) $\theta_c < 0^o$; $P_c < 0.25$ m for bottom edge position + negative crack orientation, (ii) $\theta_c > 0^o$; $P_c < 0.25$ m for bottom edge position + positive crack orientation, (iii) $\theta_c < 0^o$; $P_c \geq 0.25$ m for top edge position + negative crack orientation, and (iv) $\theta_c > 0^o$; $P_c \geq 0.25$ m for top edge position + positive crack orientation. The prediction errors corresponding to each parametric group for x-displacement, y-displacement, and scalar damage field are shown in Figure 5.12a, Figure 5.12b, and Figure 5.12c, respectively.

The results for *XDisp*-GNN presented in Figure 5.11a reveal the lowest errors for cases featuring cracks with $P_c < 0.25$ m (i.e., a bottom edge position). For example, cases with a bottom edge position + negative crack orientation exhibited errors of $0.69 \pm 0.09\%$, and cases with a bottom edge position + positive crack orientation yielded errors of $0.59 \pm 0.09\%$. Moreover, for *XDisp*-GNN the highest error of $0.84 \pm 0.15\%$ occurred for cases involving top edge positions

+ negative crack orientations. We recall from Section 5.5.3, that for cases involving large cracks the highest error ($1.00 \pm 0.18\%$) was obtained for negative crack orientations. In contrast, the error then decreased for positive crack orientations ($0.78 \pm 0.13\%$). Combining the insights from Figure 5.12a and Figure 5.11a discussed in Section 5.5.3, we note that the *XDisp*-GNN model achieved better accuracy for initial conditions with small crack length, positive crack orientation, and bottom edge position.

Unlike *XDisp*-GNN, the lowest errors for *YDisp*-GNN were obtained for cases top edge position. For instance, the prediction errors were obtained as $1.37 \pm 0.2\%$ and $1.61 \pm 0.3\%$ for $\theta_c < 0^o$; $P_c \geq 0.25$ m and $\theta_c > 0^o$; $P_c \geq 0.25$ m, respectively. Additionally, the highest error was obtained as $1.98 \pm 0.38\%$ for the group with bottom edge position + positive crack orientations. To discern why *YDisp*-GNN yielded lower prediction errors for cases with cracks situated at the upper edge position, it's essential to consider the load distribution. In this study, a uniform tensile displacement load was applied in the positive y-direction along the upper edge of the boundary. Therefore, cracks initially positioned near the top edge of the domain (i.e., where the tensile load is concentrated) may offer more information towards enhancing *YDisp*-GNN's predictive accuracy. Drawing insights from the analysis presented in Figure 5.11b within Section 5.5.3, along with Figure 5.12b, it becomes evident that *YDisp*-GNN excelled particularly in scenarios featuring large crack length, negative crack orientation, and top edge position.

Finally, as depicted in Figure 5.12c, the error analysis for *cPhi*-GNN reveals a pattern consistent with the observations made in Sections 5.5.3 through 5.5.4. Previously, we noted that predictions of ϕ were minimally influenced by crack orientations, while crack lengths played a more significant role in model accuracy. In line with this, the lowest error of $0.11 \pm 0.07\%$ was observed for the configuration involving a top edge position and negative crack orientation. Conversely, the highest error of $0.17 \pm 0.10\%$ was recorded for the scenario with a bottom edge position and positive crack orientation. Ultimately, the data in Figure 5.12c and the discussion in Section 5.5.3, we can conclude that *cPhi*-GNN performed most effectively in scenarios characterized by larger crack length, negative crack orientation, and top edge position.



(a) Percent Errors in u (b) Percent Errors in ν (c) Percent Errors in ϕ

Figure 5.12: Parametric error analysis was conducted to assess the contribution of initial crack angles and initial edge positions on: a) Predicted u. b) Predicted ν. c) Predicted ϕ.

5.5.5 Simulation time analysis

We conducted a comparative analysis of simulation times between *ADAPT*-GNN and the PF fracture model, to evaluate the computational efficiency of the *ADAPT*-GNN framework. Using an Nvidia GeForce RTX 3070 ti GPU on a personal computer system, we measured the simulation time for *ADAPT*-GNN across the entire test dataset. The simulation time for *ADAPT*-GNN was initialized prior to loading each model, and finalized at the last time-step. A similar procedure was followed for the PF fracture model. Figure 5.13 presents the mean and standard deviation of the simulation time per time-step for the PF fracture model compared to *ADAPT*-GNN. Notably, *ADAPT*-GNN showcased superior performance, achieving simulation times 15x to 36x faster than the PF fracture model. Moreover, significant enhancements in *ADAPT*-GNN's performance can be achieved by leveraging more advanced GPU units. It's important to acknowledge that while *ADAPT*-GNN exhibited superior performance in this instance, the PF model utilized in this study lacked CPU parallelization. A PF model with optimal parallel scaling may outperform *ADAPT*-GNN when employing more than 16 or 32 processors. Additionally, the extensive training times required for each model within the *ADAPT*-GNN framework should be considered. For instance, training each model for a total of 20 required 9 hours and 22 minutes for each *XDisp*-GNN and *YDisp*-GNN model, and 10 hours and 57 minutes for *cPhi*-GNN. This translates to a cumulative training time of 29 hours and 41 minutes for the framework. In scenarios where training the models from scratch is necessary, *ADAPT*-GNN would begin to outperform the PF model for 34+ simulations of 100 time-steps

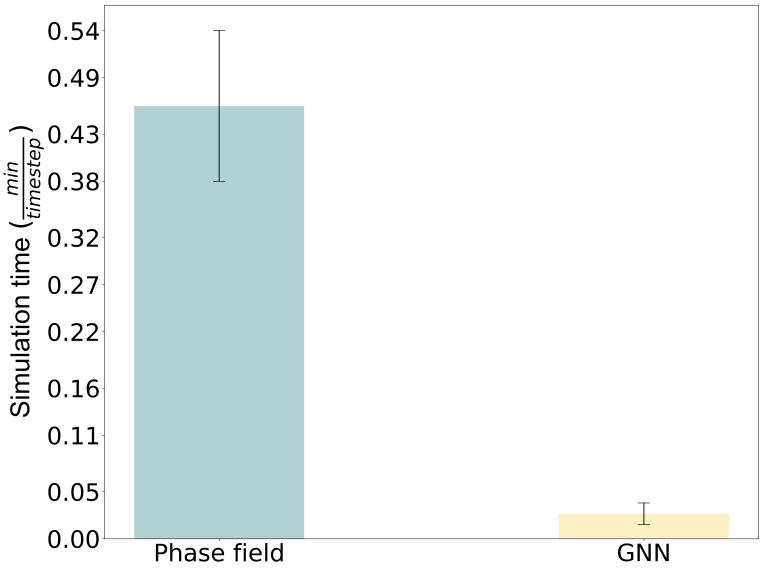


Figure 5.13: Simulation time analysis comparing the PF fracture model to the GNN framework revealed a 36x speed-up achieved by the *ADAPT*-GNN model.

each. Additionally, a significant drawback of data-driven ML methods is the substantial time required for dataset collection. For example, approximately 30 days were needed to generate 1245 simulations by simultaneously running 3-4 PF models. This underscores the importance of conventional fracture models, such as PF models, in facilitating the development of new ML frameworks aimed at accelerating computational times in the future. It's crucial to emphasize that this study does not aim to replace conventional PF fracture models but rather to demonstrate the potential of ML in expediting computational times.

5.6 Conclusion

In conclusion, the emergence of mesh-based GNN models represents a promising avenue for simulating complex fracture phenomena with notable computational efficiency gains compared to traditional high-fidelity computational models. However, the integration of this approach into PF simulations with AMR remains largely unexplored in prior studies. Addressing this gap, our work introduces an adaptive mesh-based GNN framework (*ADAPT*-GNN) designed to emulate PF fracture models for scenarios involving single-edge notched cracks under tensile

loading conditions. From Figure 5.1, the *ADAPT*-GNN framework predicts the x- and y-displacement fields first, followed by the prediction of the scalar damage field at subsequent time-steps. These predicted displacement fields as well as the scalar damage field values are then leveraged to compute the stress distribution within the material. A distinguishing feature of the developed framework is its ability to harness the computational efficiencies offered by both ML techniques and AMR. This is accomplished by formulating each instantaneous graph using the instantaneous refined mesh itself. When utilizing an NVIDIA GeForce RTX 3070 Ti GPU on a personal computer, this dynamic graph representation results in significant simulation speed-ups, achieving up to 36 times faster computation times compared to the PF fracture model. The framework also demonstrates commendable prediction accuracies across the test dataset, with maximum errors of $1.98 \pm 0.27\%$, $2.74 \pm 0.26\%$, and $0.19 \pm 0.13\%$ for x-displacements, y-displacements, and scalar damage field values, respectively.

However, it is essential to acknowledge several limitations inherent to the *ADAPT*-GNN framework. Notably, the PF model utilized in this study lacked parallel CPU capabilities, potentially impacting comparative performance metrics in scenarios with extensive processor utilization. For instance, using 16 or 32 processors in a PF model with parallel CPU capabilities may result in faster simulation time than the *ADAPT*-GNN framework. Additionally, the necessity for re-training each model in the framework may limit the framework's superior simulation time over the PF model for generating less than 34 simulations (each comprising 100 time-steps). Furthermore, *ADAPT*-GNN currently lacks the capability to predict cases involving center cracks, shear loading, and cracks located at the right edge of the domain, underscoring the importance of conventional fracture models in advancing ML algorithms.

Looking ahead, future endeavors may explore transfer learning (TL) approaches, such as those proposed by Perera et al. [4], to enhance the framework's predictive capabilities for unseen scenarios. Extending the *ADAPT*-GNN framework to encompass a broader range of PF models beyond fracture models is a promising direction for research. As novel GNN techniques continue to evolve, innovative methodologies like subgraphs may be investigated to further enhance computational efficiency and accuracy in fracture simulations.

5.7 Data availability

The trained models with examples can be found in the following GitHub repository `https://github.com/rperera12/Phase-Field-ADAPT-GNN`. Supplementary data containing animations have been included along with the manuscript.

Chapter 6

Multiscale graph neural networks with adaptive mesh refinement for accelerating mesh-based simulations

6.1 Introduction and motivation

Recently, mesh-based GNNs have garnered attention across various engineering domains [252, 253, 254, 5, 255, 256]. However, mesh-based GNNs face challenges when dealing with problems characterized by very fine meshes. The inherent message-passing (MP) blocks [176] utilized in these frameworks for information transfer and relation learning between nodes and edges [169, 258] introduce a significant challenge. For fine meshes, GNN-based approaches often necessitate numerous MP blocks to effectively capture relationships between nodes located far apart, a requirement that is unfavorable for many mesh-based problems where information exchange between distant nodes is crucial for achieving high accuracies. Moreover, increasing the number of MP blocks can lead to over-smoothing [264, 265, 266].

To address these challenges, Multiscale GNNs have emerged [267], mimicking traditional iterative multigrid schemes. These models undergo multiple graph coarsening operations, generating smaller meshes at each level, which are subsequently processed through MP blocks [268, 269, 270]. The downscaled mesh levels establish new connections between distant nodes, facilitating information transfer beyond previous local neighborhoods and mitigating the need for an excessive number of MP steps. This technique has demonstrated superior accuracy compared to conventional mesh-based GNNs across dynamic problems such as PF crack propagation [6], flow field predictions [271], fluid dynamics over varying-shaped rigid bodies [272, 273], and time-independent PDEs on unstructured meshes and sparse linear systems [274, 275, 276]. Despite their efficacy to avoid over-smoothing, these studies were formulated

using fixed initial graphs and employed different graph-pooling techniques for downsampling steps [277, 272, 278, 279], which can potentially increase computational costs and have yet to be tailored for problems involving AMR.

In this study, we propose a novel multiscale GNN coupled with block-structured AMR (BSAMR) to simulate time-evolving mesh-based problems featuring near-singular operators. An example of a highly complex mesh-based time-evolving problem featuring near-singular is PF fracture. Our framework introduces a straightforward and computationally efficient downsampling/upscaling strategy for mesh-based AMR scenarios by leveraging multiple mesh resolution levels. Specifically, as depicted in Figure 6.1 our downsampling/upscaling approach iteratively reduces the mesh size by eliminating the highest mesh refinement level during each coarsening operation. We evaluate the framework's performance on PF fracture problems employing five AMR levels and compare the accuracy and computational time of three coarsening approaches: (i) four coarsening/downscale operations, (ii) two coarsening/downscale operations, and (iii) one coarsening/downscale operation, progressing from the finest to the coarsest mesh resolution. Initially focusing on Mode-I fracture problems featuring a single crack on the left domain edge, we then implement TL [280, 4] to extend the framework's applicability across other configurations of initial conditions. For instance, cases involving (i) center cracks, (ii) right-edge cracks, and (iii) shear loading. Subsequently, we conduct a comprehensive accuracy analysis for each problem configuration and compare the computational efficiency of our developed multiscale and adaptive GNN framework against high-fidelity PF models.

6.2 Methods

6.2.1 Physics based PF fracture model

In this study, we present a multiscale mesh-based GNN framework coupled with BSAMR tailored for PF fracture problems. PF fracture simulations pose significant computational challenges due to two main factors: (i) the presence of near-singular operators arising from the near-zero modulus within the crack field, and (ii) the necessity of high mesh resolution in the vicinity of the crack tip to accurately capture the crack propagation behavior. Towards the

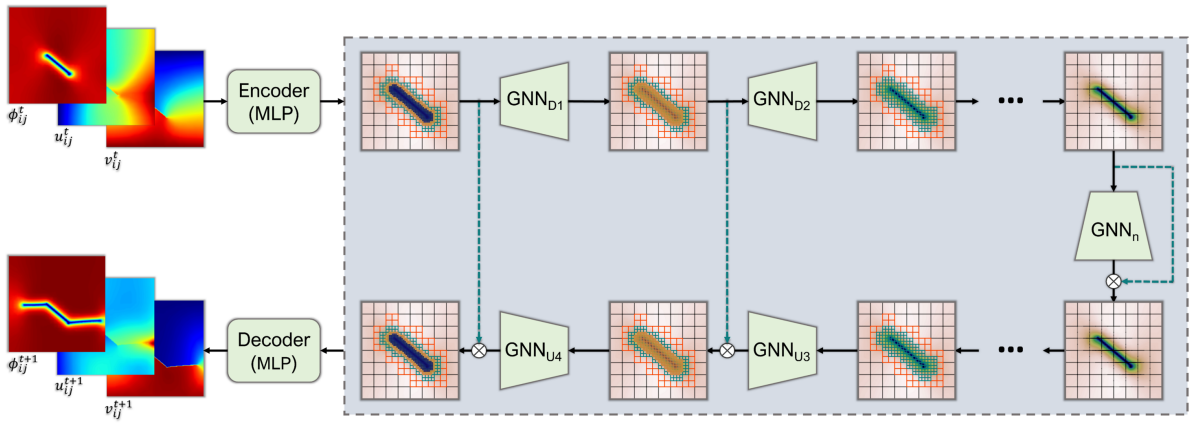
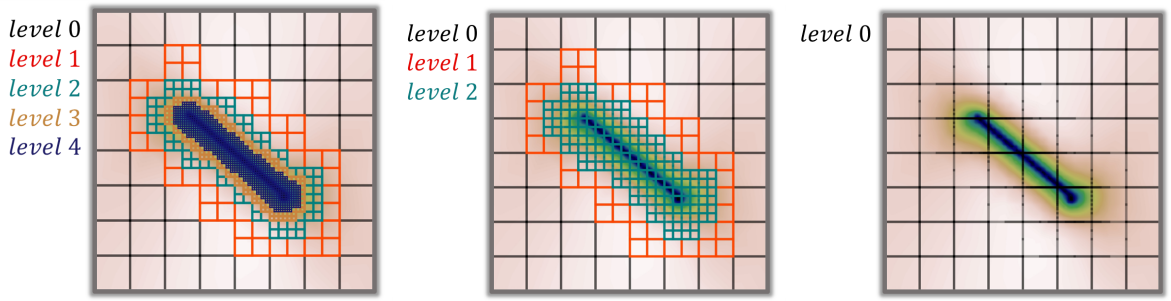


Figure 6.1: The architecture of the multiscale GNN framework for the Four-Stage Refinement architecture is as follows: The model initially utilizes MLP networks to encode the input graph representation. The feature embedding then undergoes processing through MP GNN blocks followed by mesh coarsening operations denoted by GNN_{D1} and GNN_{D2}. An additional MP GNN block, GNN_n, operates on the coarsest mesh. Subsequently, the resulting coarsened embedding is reconstructed through MP GNN blocks followed by mesh upscaling operations denoted by GNN_{U3} and GNN_{U4}. The dashed green lines defines skip connectors. Finally, the reconstructed embedding is passed through a decoder MLP network to predict the crack field and displacement fields at future times.

PF formulation of fracture mechanics, the sharp crack is regularized using the smooth scalar field $\phi(\boldsymbol{x})$, which varies continuously between 0 and 1. This scalar damage field is utilized to construct an energy functional Π, which is subsequently minimized to derive a set of coupled PDEs governing elastic equilibrium and crack propagation. For a detailed review of the PF fracture formulation, readers are directed to the extensive literature on the subject [281].

In our work, we adopt a second-order energy functional as described in Section 5.2.1 of Chapter 5. Following the methodology outlined in [5], we consider a linear elastic isotropic brittle material with dimensions $0.5m \times 0.5m$, characterized by a Young's modulus of $E = 210$ GPa, a Poisson's ratio of $\nu = 0.3$, and a fracture energy density of $G_c = 2.7$ N/m^2, with a characteristic length scale $d = 0.0125$m. In our simulations, we fix the bottom edge of the material and apply displacement along the top edge, with loading direction corresponding to the desired loading condition (tensile or shear). We systematically vary the initial crack length C_L, edge position C_P, and crack angle C_θ to generate datasets encompassing diverse initial conditions, enabling comprehensive analysis and evaluation of our proposed framework's performance.



(a) Initial refined graph: $\mathcal{M}_4^{ref}$ (b) Second downscale graph: $\mathcal{M}_2^{ref}$ (c) Fourth downscale graph: $\mathcal{M}_0^{ref}$

Figure 6.2: The representation of the instantaneous refined mesh graphs includes: a) Refinement levels 0-4. b) Refinement levels 0-2. c) Refinement level 0 (i.e., coarsest mesh).

6.2.2 Graph Neural Network

To define the graph representation for the multiscale and adaptive GNN framework with BSAMR capabilities, we modified the approach used in the *ADAPT*-GNN, as described in Section 5.2.2 of Chapter 5. As depicted in Figure 6.2, similar to the *ADAPT*-GNN model, we constructed the graph representation utilizing the instantaneous refined mesh $\mathcal{M}^{ref} : \langle \mathbf{U}, \mathbf{E} \rangle$, where $\mathbf{U}$ encompasses the mesh vertices and $\mathbf{E}$ comprises the resulting mesh edges. Node features ξ_s incorporate the mesh vertex positions $\hat{P}s$, their corresponding x- and y-displacement values $\hat{D}s$, and their scalar damage field values ϕs. In addition to these existing features, we introduced two new node features to represent the applied displacement loading along the x- and y-directions, denoted as $u0_s, v0s$, respectively.

$$\begin{aligned} \hat{P}_s &= \{(x_s, y_s)\} \qquad \{s \in \mathbf{U}\}, \{\mathbf{U} \in \mathcal{M}^{ref}\}, \\ \hat{D}_s &= \{(u_s, v_s)\} \qquad \{s \in \mathbf{U}\}, \{\mathbf{U} \in \mathcal{M}^{ref}\}, \\ \{\xi_s\} &= \{\hat{P}_s, \hat{D}_s, \phi_s, u_{0_s}, v_{0_s}\} \qquad \{s \in \mathbf{U}\}, \{\mathbf{U} \in \mathcal{M}^{ref}\}. \end{aligned} \tag{6.1}$$

These applied displacement loading features enable the framework to distinguish between cases subjected to tensile loading and those experiencing shear loading. Edge features e_{sr} are defined based on the binary connectivity array with values of $b_{sr} \in 0, 1$, where s and r denote the "sender" and "receiver" nodes, respectively. Specifically, $b_{sr} = 1$ for nodes sharing an edge, as

well as for cases where $s = r$.

$$\{e_{sr}\} = \{\xi_s, \xi_r, b_{sr}\} \qquad \{(s, r, b_{sr}) \in \mathbf{E}\}, \{\mathbf{E} \in \mathcal{M}^{ref}\}. \tag{6.2}$$

Moreover, as shown in Figure 6.2a, we construct the initial graph for each time-step in a given simulation using the instantaneous refined mesh comprising all five mesh resolution levels, denoted as $\mathcal{M}^{ref}$. The framework then iteratively removes each refinement level until reaching the coarsest mesh at level 0 resolution, as illustrated in Figure 6.1. We define the resulting graphs as follows: (i) $\mathcal{M}^{ref}4$ representing the initial instantaneous refined mesh with mesh resolution levels 0-4, (ii) $\mathcal{M}^{ref}3$ denoting the first downscaled mesh with mesh resolution levels 0-3, (iii) $\mathcal{M}^{ref}2$ denoting the second downscaled mesh with mesh resolution levels 0-2, (iv) $\mathcal{M}^{ref}1$ representing the third downscaled mesh with mesh resolution levels 0-1, and (v) $\mathcal{M}_0^{ref}$ representing the coarsest downscaled mesh with mesh resolution level 0 only.

6.2.3 Multiscale GNN framework

The presented multiscale and adaptive GNN framework seamlessly integrates AMR into a multigrid approach. The initial mesh refinement graph undergoes coarsening by removing the finest mesh level at each coarsening operation step. This coarsening process enhances connectivity between distant nodes, enabling the transfer of information beyond previous local neighborhoods. The approach reduces the number of MP steps required, thereby mitigating over-smoothing and substantially decreasing computational costs while preserving high prediction accuracy.

In Figure 6.1, we show the architecture of the multiscale GNN framework. As depicted in Figure 6.2a and defined in Section 6.2.2, the initial step of the framework involves feeding the graph representation for a given time "t", into an encoder network. Here, we employ an MLP model denoted as MLP_{in} to serve as the encoder network.

$$\{\xi_s'\} \leftarrow MLP_{\text{in}}\{\xi_s^t, e_{sr}^t\} \qquad \{s \in \mathbf{U}\}, \{(s, r, b_{sr}) \in \mathbf{E}\}, \{(\mathbf{U}, \mathbf{E}) \in \mathcal{M}^{ref}\}. \tag{6.3}$$

To capture relationships within local neighborhoods, the resulting latent-space embedding, denoted as $\xi_s^{'}$, is then fed into an MP block, labeled as GNN_1. In this framework, we utilize Graph Transformers [282] for the MP networks. Originally introduced for sequence modeling and language translation tasks [283], Graph Transformers have been extended for graph-based applications by incorporating multi-head and self-attention mechanisms to process vertex and edge features of neighboring vertices, thereby generating additional attention embeddings [282]. The architecture of all Transformer MP networks our framework consist of four attention heads and 128 hidden nodes. The block labeled as "GNN_{D1}" in Figure 6.1 encompasses the first MP network, GNN_1, along with the first downscale operator, $Down_1$. Each downscale operator removes the current finest mesh refinement level. For example, $Down_1$ from Figure 6.1 eliminates refinement level 4 from the graph $\mathcal{M}_4^{ref}$ shown in Figure 6.2a. The resulting new graph from the $Down_1$ operator is graph $\mathcal{M}_3^{ref}$.

$$\begin{aligned}
\{\xi_s^4\} \leftarrow GNN_1\{\xi_s^{'}, e_{sr}^t\} \qquad & \{s \in \mathbf{U}\}, \{(s, r, b_{sr}) \in \mathbf{E}\}, \{(\mathbf{U}, \mathbf{E}) \in \mathcal{M}^{ref}\}, \\
\{\xi_s^3\} \leftarrow Down_1\{\xi_s^4, e_{sr}^t, e_{sr}^3\} \qquad & \{s \in \mathbf{U}^3\}, \{(s, r, b_{sr}) \in \mathbf{E}^3\}, \{(\mathbf{U}^3, \mathbf{E}^3) \in \mathcal{M}_3^{ref}\}. \quad (6.4)
\end{aligned}$$

Where ξ_s^4 represents the resulting node embedding obtained from the MP block GNN_1 in the refined mesh $\mathcal{M}^{ref}$. Additionally, e_{sr}^3 denotes the edge feature vector for the downscaled mesh $\mathcal{M}_3^{ref}$, and ξ_s^3 denotes the resulting downscaled node embedding transitioning from $\mathcal{M}^{ref}$ to $\mathcal{M}_3^{ref}$. Similarly, the block labeled as "GNN_{D2}" in Figure 6.1 encompasses another MP network, GNN_2, along with the second downscale operator, $Down_2$. As depicted in Figure 6.2b, $Down_2$ results in graph $\mathcal{M}_2^{ref}$ by eliminating refinement level 3 from the graph $\mathcal{M}_3^{ref}$.

$$\begin{aligned}
\{\xi_s^3\} \leftarrow GNN_2\{\xi_s^3, e_{sr}^3\} \qquad & \{s \in \mathbf{U}^{\ni}\}, \{(s, r, b_{sr}) \in \mathbf{E}^{\ni}\}, \{(\mathbf{U}^3, \mathbf{E}^3) \in \mathcal{M}_3^{ref}\}, \\
\{\xi_s^2\} \leftarrow Down_2\{\xi_s^3, e_{sr}^3, e_{sr}^2\} \qquad & \{s \in \mathbf{U}^2\}, \{(s, r, b_{sr}) \in \mathbf{E}^2\}, \{(\mathbf{U}^2, \mathbf{E}^2) \in \mathcal{M}_2^{ref}\}, \quad (6.5)
\end{aligned}$$

where e_{sr}^2 represents the edge feature vector for the downscaled mesh $\mathcal{M}_2^{ref}$, and ξ_s^2 denotes the resulting downscaled node embedding transitioning from $\mathcal{M}_3^{ref}$ to $\mathcal{M}_2^{ref}$. This process is repeated using two additional MP blocks, each followed by their respective coarsening operations.

As illustrated in Figure 6.2c, these operations result in the downscaled node embedding for the coarsest mesh ξ_s^0, from $\mathcal{M}_1^{ref}$ to $\mathcal{M}_0^{ref}$. The framework then incorporates an additional MP network, labeled as "GNN_n" in Figure 6.1, for operating on ξ_s^0.

$$\{\xi_s^{0'}\} \leftarrow Agg\left[\xi_s^0, GNN_n\{\xi_s^0, e_{sr}^0\}\right] \qquad \{s \in \mathbf{U}^0\}, \{(s, r, b_{sr}) \in \mathbf{E}^0\}, \{(\mathbf{U}^0, \mathbf{E}^0) \in \mathcal{M}_0^{ref}\} \tag{6.6}$$

Where, e_{sr}^0 denotes the edge feature vector for the downscaled mesh $\mathcal{M}_0^{ref}$ as depicted in Figure 6.2c. The operation Agg refers to an aggregation operation for the skip connectors, which are illustrated as dashed green lines in Figure 6.1. Finally, $\xi_s^{0'}$ denotes the resulting node embedding for mesh $\mathcal{M}_0^{ref}$ obtained by aggregating the outputs from MP block GNN$_n$ and the skip connector.

In this step, "GNN$_{U1}$" incorporates the first Upscale operation, denoted as $Up1$, to reconstruct mesh level 1 ($\mathcal{M}_1^{ref}$). This includes the skip connector from the node embedding ξ_s^1 to the MP GNN, represented as GNN_4.

$$\begin{aligned} \{\xi_s^{1'}\} &\leftarrow Up_1\{\xi_s^{0'}, e_{sr}^0, e_{sr}^1\} \qquad \{s \in \mathbf{U}^1\}, \{(s, r, b_{sr}) \in \mathbf{E}^1\}, \{(\mathbf{U}^1, \mathbf{E}^1) \in \mathcal{M}_1^{ref}\}, \\ \{\xi_s^{1''}\} &\leftarrow Agg\left[\xi_s^1, GNN_4\{\xi_s^{1'}, e_{sr}^1\}\right] \qquad \{s \in \mathbf{U}^1\}, \{(s, r, b_{sr}) \in \mathbf{E}^1\}, \{(\mathbf{U}^1, \mathbf{E}^1) \in \mathcal{M}_1^{ref}\}, \end{aligned} \tag{6.7}$$

$\xi_s^{1'}$ represents the reconstructed node embedding from Up_1, which involves transitioning from $\mathcal{M}_0^{ref}$ to $\mathcal{M}_1^{ref}$. $\xi_s^{1''}$ then represents the resulting node embedding for mesh $\mathcal{M}_1^{ref}$, derived from aggregating the output of the MP block GNN_4 and the skip connector originating from ξ_s^1. As depicted in Figure 6.2b, "GNN_{U2}" employs the second Upscale operation, Up_2, to reconstruct mesh $\mathcal{M}_2^{ref}$. This operation is followed by the skip connector from ξ_s^2 to GNN_3.

$$\begin{aligned} \{\xi_s^{2'}\} &\leftarrow Up_2\{\xi_s^{1'}, e_{sr}^1, e_{sr}^2\} \qquad \{s \in \mathbf{U}^2\}, \{(s, r, b_{sr}) \in \mathbf{E}^2\}, \{(\mathbf{U}^2, \mathbf{E}^2) \in \mathcal{M}_2^{ref}\}, \\ \{\xi_s^{2''}\} &\leftarrow Agg\left[\xi_s^2, GNN_3\{\xi_s^{2'}, e_{sr}^2\}\right] \qquad \{s \in \mathbf{U}^2\}, \{(s, r, b_{sr}) \in \mathbf{E}^2\}, \{(\mathbf{U}^2, \mathbf{E}^2) \in \mathcal{M}_2^{ref}\}, \end{aligned} \tag{6.8}$$

In this equation, $\xi_s^{2'}$ represents the resulting reconstructed node embedding from the upscale operation, Up_2, involving the transition from $\mathcal{M}_1^{ref}$ to $\mathcal{M}_2^{ref}$. $\xi_s^{2''}$ represents the resulting node embedding for mesh $\mathcal{M}_2^{ref}$, which is derived from aggregating the output of the MP block

GNN_3 and the skip connector originating from ξ_s^2. As shown Figure 6.1, this process is repeated using two additional upscale blocks (GNN_{U3} and GNN_{U4}) until the initial refined mesh $\mathcal{M}^{ref}$ with mesh levels 0-4 is reconstructed, yielding the resulting aggregated node embedding $\xi_s^{4''}$.

The final generated reconstructed embedding, $\xi_s^{4''}$, serves as input to a Decoder MLP network, MLP_{out}, which transfers the embedding from the latent-space to the real-space for predicting the displacements and scalar damage field values at the future time-step, denoted as "$t+1$".

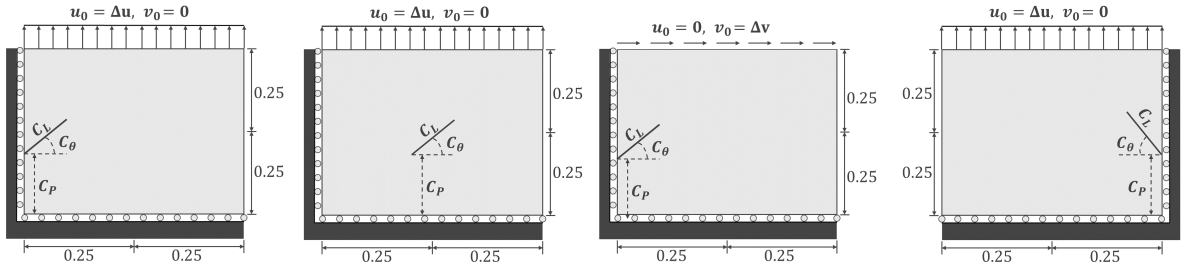
$$\{\xi_s^{t+1}\} \leftarrow MLP_{\text{out}}\{\xi_s^{4''}, e_{sr}^t\} \qquad \{s \in \mathbf{U}\}, \{(s, r, b_{sr}) \in \mathbf{E}\}, \{(\mathbf{U}, \mathbf{E}) \in \mathcal{M}^{ref}\}. \quad (6.9)$$

6.2.4 Single-Stage, Two-Stage, and Four-Stage Refinement GNNs

Although increasing the number of downscale/upscale operations and MP GNNs typically enhances the framework's ability to capture intricate features and relationships within the data, potentially leading to higher prediction accuracy, this comes at the expense of computational resources, including increased memory usage and longer training and inference times. Therefore, we evaluated three different architectures to balance prediction accuracy with computational cost following the methods described in Section 6.2.3. For each new architecture, we reduced the number of upscale/downscale operations, and MP blocks. These architectures are as follows.

1. Four-Stage Refinement (FSR) GNN: This framework involves four downscale and four upscale operations, resulting in multiple MP GNNs and intermediate mesh resolution levels. Detailed in Section 6.2.3.

2. Two-Stage Refinement (TSR) GNN: The TSR framework consists of two downscale and two upscale operations. Each operation accounts for two mesh resolution levels, effectively reducing the number of MP blocks and intermediate steps. For example, the first downscale operation, GNN_{D1}, removes mesh resolution levels 4 and 3, resulting in a downscaled node embedding ξ_s^2 for the graph $\mathcal{M}_2^{ref}$. Expected to be faster than FSR due to the reduced complexity.



(a) Left-edge crack in tension (b) Center crack in tension (c) Left-edge crack in shear (d) Right-edge crack in tension

Figure 6.3: The setup for the problem geometry and input parameters C_P, C_θ, C_L is as follows: a) Initial case involving left-edge cracks under tension. b) Center cracks under tension. c) Left-edge cracks under shear. c) Right-edge cracks under tension.

3. Single-Stage GNN (SSR): SSR involves only one downscale and upscale operation, significantly simplifying the architecture compared to FSR and TSR by removing three downscale/upscale operations, and six MP blocks. For example, the single downscale operation GNN_{D1} removes mesh resolution levels 1-4 directly from the initial node embedding ξ_s^4, resulting in the coarsest mesh level depicted in Figure 6.2c. SSR is the least computationally expensive as it eliminates the need for intermediate mesh resolution levels and additional MP Transformer GNNs.

These frameworks offer a trade-off between prediction accuracy and computational cost, with SSR being the most computationally efficient but potentially sacrificing some accuracy compared to FSR and TSR.

6.2.5 Transfer learning

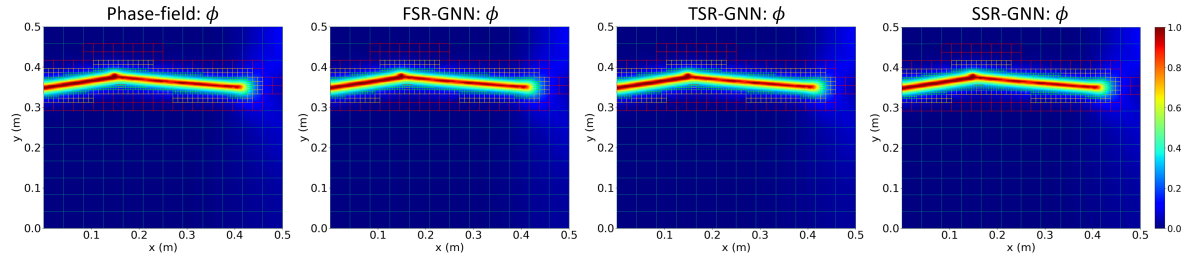
To train the adaptive multiscale GNN framework across different case studies, we collected a dataset comprising 1100 PF fracture simulations for single-edge notched systems subjected to tension (shown in Figure 6.3a) from previous research [5]. Each simulation represented a unique crack configuration with variations in initial crack orientation C_θ, crack length C_L, and edge position C_P. Next, we trained the multiscale GNN framework using the gathered dataset. During training, the model learned to predict displacement and damage fields based on input graph representations derived from the simulation data. To adapt the pretrained multiscale GNN to new case studies, we then employed TL. This involved transferring the weights of the encoder

MLP MLP_{in}, and the first MP model GNN_{D1}, from the pretrained framework for left-edge crack systems, into a new framework defined using similar architecture.

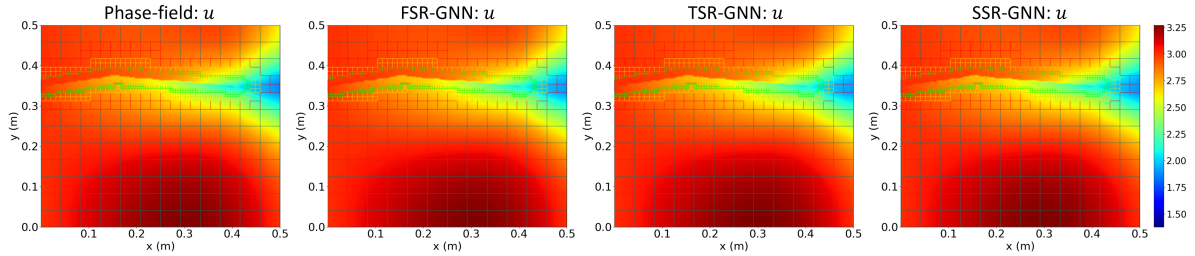
We then performed a sequence of TL update steps, transferring the pretrained weights to each new case study. This iterative process allowed the model to fine-tune its parameters and adapt to the specific characteristics of each case. Thus, ensuring that the adaptive multiscale GNN framework effectively leveraged knowledge learned from the initial training phase and adapted to new cases with varying crack configurations and loading conditions. Four case studies were considered for TL using this approach.

- Case 1: Left-edge cracks subjected to Mode I loading as shown in Figure 6.3a.
- Case 2: Center cracks subjected to Mode I loading as shown in Figure 6.3b.
- Case 3: Left-edge cracks subjected to Mode II loading as shown in Figure 6.3c.
- Case 4: Right-edge cracks subjected to Mode I loading as shown in Figure 6.3d.

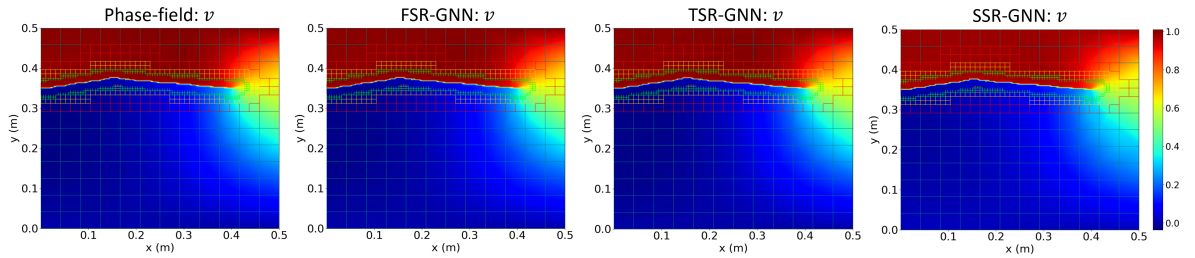
In Case 1, left-edge cracks propagate exclusively towards the right. In contrast, Case 2 involves center cracks that can propagate in both left and right directions, thereby enhancing the framework's adaptability. To address Cases 2 and 3, which entail center cracks and shear loading scenarios, we utilized the PF model introduced in [211] to generate new datasets. Each case comprised 30 simulations, divided equally as 15 simulations for training and 15 simulations for testing purposes. In Case 4, we capitalized on the symmetry observed in the left-edge crack scenario (Case 1) by mirroring 30 randomly selected simulations from the training data developed for Case 1. It's noteworthy that the adoption of TL led to a substantial reduction in the number of simulations required for the training data, shrinking from 1100 simulations in Case 1 to just 15 simulations per case in Cases 2-4. This reduction, approximately 70 times smaller, underscores the efficiency gains enabled by TL. The integration of TL not only facilitates the extension of the multiscale GNN framework to diverse crack problems but also enhances simulation efficiency and reduces computational costs significantly.



(a) Crack field, ϕ.



(b) X-displacement fields, u



(c) Y-displacement fields, v

Figure 6.4: Comparing the FSR, TSR, and SSR frameworks with the high-fidelity PF model in left-edge crack cases subjected to tension for predicting: a) The crack field, ϕ. b) X-displacement fields. c) Y-displacement fields.

6.3 Results and discussion

6.3.1 Prediction and error analysis for FSR, TSR and SSR

Here, we present a comparison of the performance of the FSR, TSR, and SSR architectures within the multiscale GNN framework. We obtained predictions for the crack variable ϕ, x-displacement, and y-displacement for left-edge crack cases. Figures 6.4a - 6.4c illustrate the predicted ϕ values and displacements for a randomly selected simulation from the test dataset of Case 1. The chosen simulation features a crack positioned near the top of the domain's left edge, with a positive angle, gradually approaching the right edge, indicating complete material failure. All three architectures exhibit nearly identical predictions for ϕ compared to PF (left-most case

shown from Figure 6.4a). Similarly, Figures 6.4b - 6.4c show that the FSR, TSR, and SSR architectures qualitatively predict displacements with remarkable accuracy. This qualitative analysis indicates that despite the reduced number of downsampling/upsampling steps, the TSR and SSR architectures are capable of maintaining prediction accuracy.

Next, we computed errors for each simulation in the test dataset. For each time-step of each simulation, we calculated the average percent error across all mesh points in $\mathcal{M}^{ref}$. For each simulation, we then averaged these percent errors across all time-steps. For instance, the error for the crack field was computed as:

$$\phi_{error} = \frac{1}{T_f} \sum_{t}^{T_f} \left[\frac{1}{M} \sum_{s}^{M} \frac{|\phi_s^{pred} - \phi_s^{true}|}{\phi_s^{true}} \times 100 \right]_t . \tag{6.10}$$

Here, ϕ_s^{true} defines the PF scalar damage field, ϕ_s^{pred} denotes the predicted scalar damage field, M defines the total number of mesh points in $\mathcal{M}^{ref}$, t represents the first predicted time-step, and T_f the final predicted time-step. We used equation (6.10) and compiled the errors in ϕ and displacements for each test simulation across the FSR, TSR, and SSR models. Figures 6.5a - 6.5c illustrate the errors in ϕ and displacements obtained for the FSR model. The errors in ϕ and displacements for the TSR model are shown in Figures 6.5d - 6.5f. Similarly, for the SSR model, Figure 6.5g - 6.5i depict the resulting errors for ϕ and displacements.

Comparing errors in ϕ (left-most), we observe that all architectures exhibited average errors below 0.3%. Similarly, errors in x-displacement and y-displacement remained under 0.35% for the FSR, TSR, and SSR models. Despite the reduced number downsampling/upsampling steps, these results demonstrate that the TSR and SSR architectures maintained high prediction accuracy, confirming the earlier results from the qualitative analysis.

Subsequently, for each architecture we computed the resulting average percent errors of the testing simulations. As shown in Figure 6.6a, although the FSR architecture exhibited the highest accuracy in predicting ϕ, SSR displayed lower errors compared to TSR. For predictions of x-displacement, the FSR and SSR models demonstrated close errors at around 0.08%. However, the TSR model yielded the lowest error x-displacement. For predictions of y-displacement, the FSR and TSR models showcased low errors of similar magnitude (roughly 0.08%). The SSR

model recorded the highest y-displacement error nearing 0.12%. Ultimately, the figure illustrates that for TSR and SSR, reducing the number of refinement steps did not notably escalate errors in predictions. We note that even the highest obtained error when predicting y-displacement using SSR remains considerably low at 0.12% compared to previous research [5].

6.3.2 Simulation time analysis for FSR, TSR, and SSR

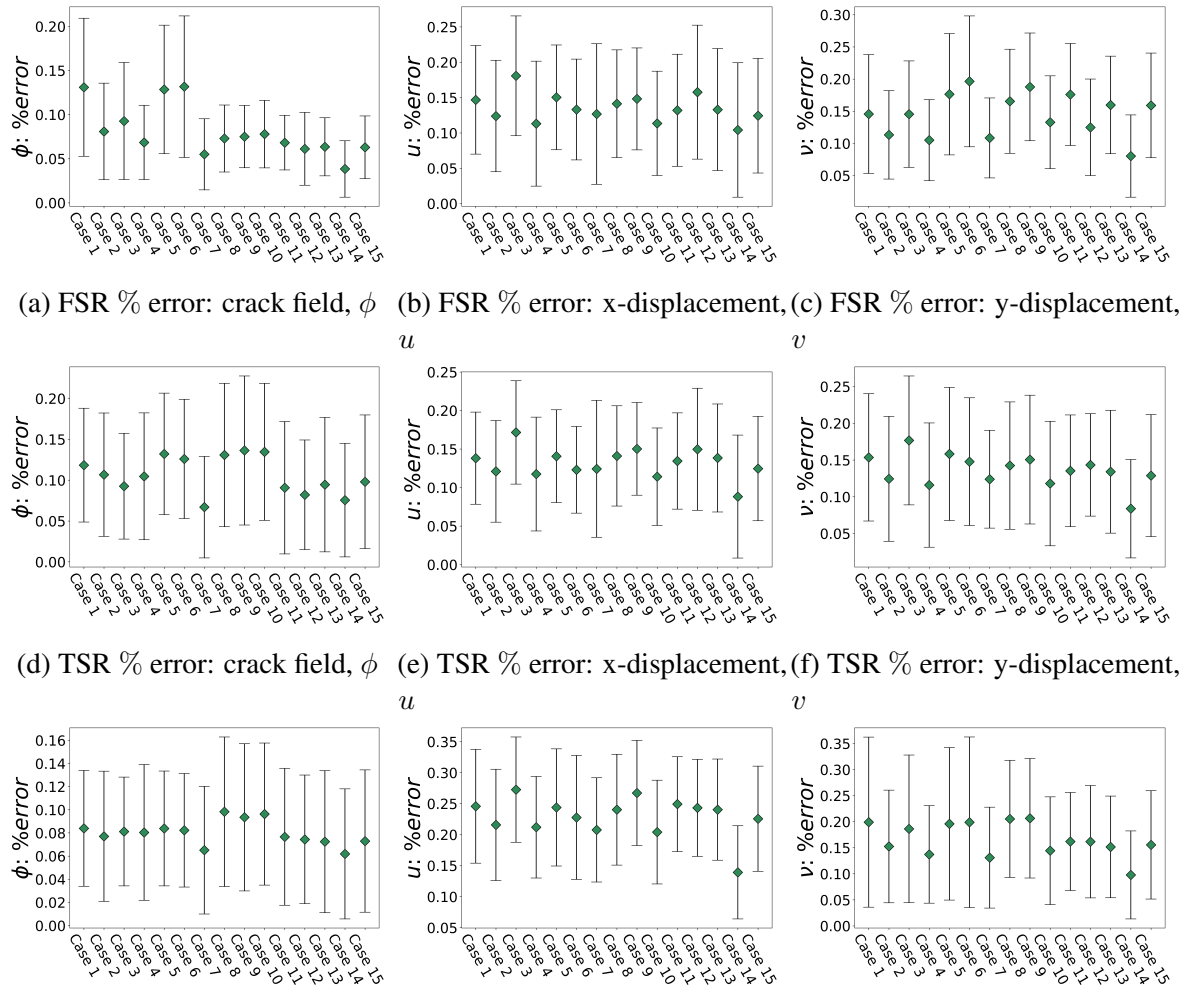
We conducted a computational cost comparison among the FSR, TSR, and SSR architectures by assessing the time needed to generate 30 simulations. In Section 6.3.1, we demonstrated that despite fewer downscaling and upscaling operations in the TSR and SSR architectures, prediction errors did not notably increase. However, each downscale/upscale operation adds to computational costs, necessitating storage of resulting mesh configurations and node embeddings, along with additional MP GNNs.

In Figure 6.6b we show the total simulation time (hours) required for each framework to generate the 30 randomly selected simulations. Notably, the FSR GNN architecture exhibited the longest simulation time, consuming 6.13 hours. Following the FSR model, the TSR architecture demanded 4.75 hours for the same task. As expected, the SSR architecture proved to be the most efficient in terms of computational costs, completing the task in just 3.91 hours. Comparatively, it's worth noting from [5] that the high-fidelity PF model took approximately 43.5 hours to generate the same 30 cases. Therefore, we opted for the SSR architecture for subsequent TL steps, given its lower simulation time coupled with high prediction accuracy.

6.3.3 Center crack cases

In Sections 6.3.1 through 6.3.2, we established that the SSR architecture offered high prediction accuracy at notably lower computational costs. Leveraging this architecture, we applied TL to extend the multiscale GNN framework to cases with center cracks, as depicted in Figure 6.3b. We evaluated the performance of the extended SSR framework in predicting crack and displacement fields for a randomly selected simulation from the test dataset.

First, we compared the PF versus predicted crack field (Figure 6.7a), the x-displacement field (Figure 6.7c), and the y-displacement field (Figure 6.7e) at the initial time-step t_0. Then, we

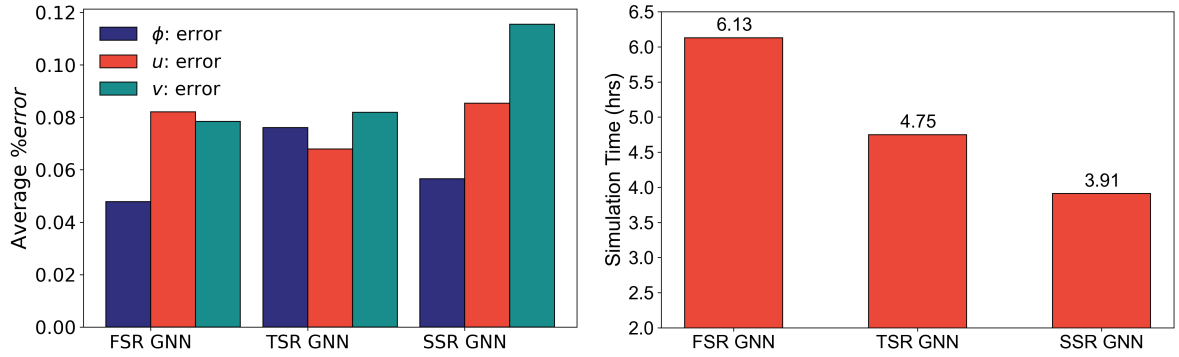


(a) FSR % error: crack field, ϕ (b) FSR % error: x-displacement, u (c) FSR % error: y-displacement, v

(d) TSR % error: crack field, ϕ (e) TSR % error: x-displacement, u (f) TSR % error: y-displacement, v

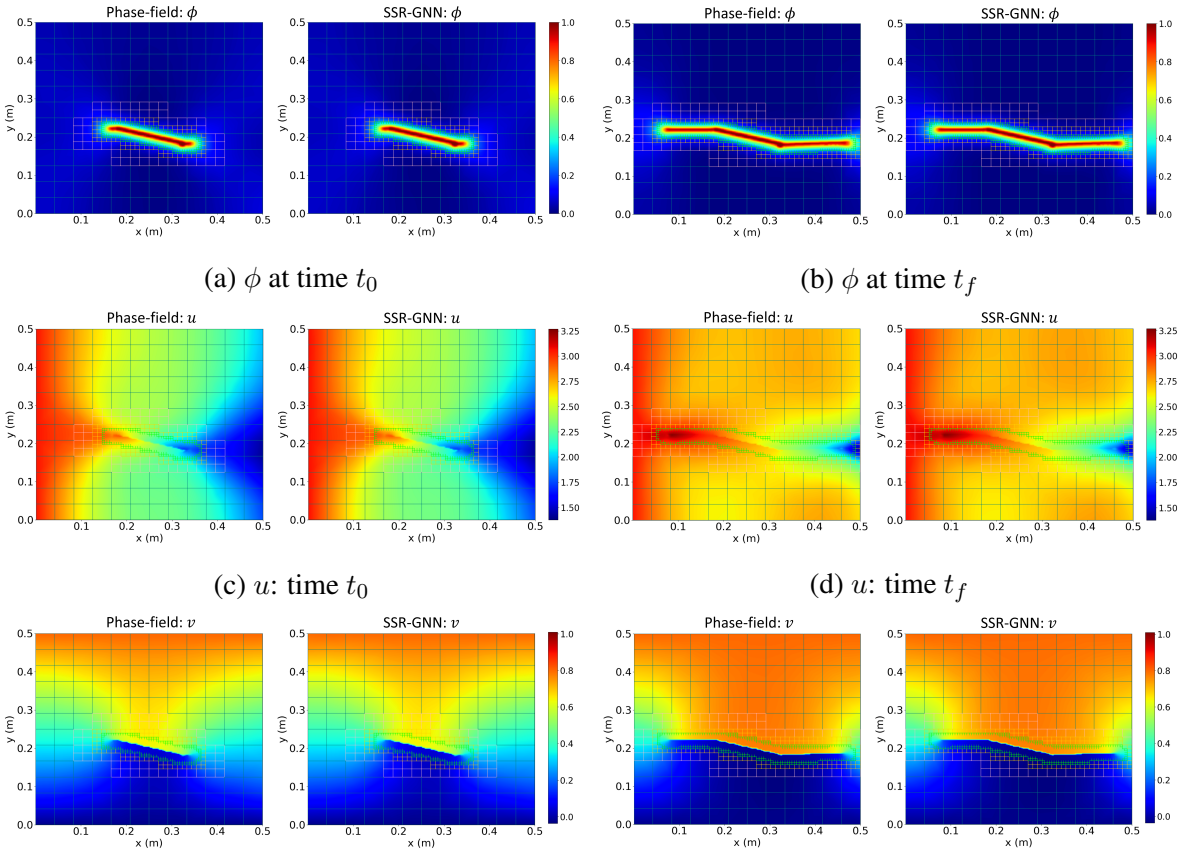
(g) SSR % error: crack field, ϕ (h) SSR % error: x-displacement, u (i) SSR % error: y-displacement, v

Figure 6.5: Comparing the average % errors for left-edge crack cases under tension for a) FSR GNN crack-field predictions. b) FSR GNN x-displacement predictions. c) FSR GNN y-displacement predictions. d) TSR GNN crack-field predictions. e) TSR GNN x-displacement predictions. f) TSR GNN y-displacement predictions. g) SSR GNN crack-field predictions. h) SSR GNN x-displacement predictions. i) SSR GNN y-displacement predictions.



(a) Average $\%$ errors for FSR, TSR, and SSR. (b) Simulation time (hrs) for FSR, TSR, and SSR.

Figure 6.6: a) Analysis of errors in the FSR, TSR, and SSR frameworks based on the average percentage error across all testing simulations. b) Evaluation of simulation time (in hours) required for the FSR, TSR, and SSR frameworks to generate 30 simulations.

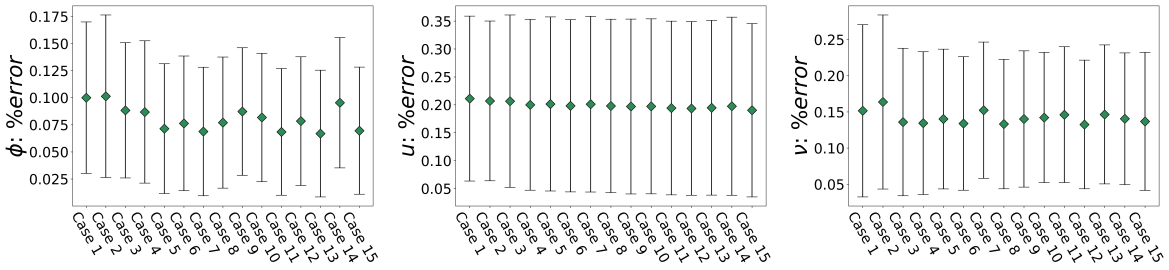


(a) ϕ at time t_0 (b) ϕ at time t_f

(c) u: time t_0 (d) u: time t_f

(e) v: time t_0 (f) v: time t_f

Figure 6.7: Comparison between the PF model and the SSR framework regarding the predicted evolution of the crack field, ϕ (a-b), x-displacement field, u (c-d), and y-displacement field, v (e-f), for a center crack test simulation at the initial time-step, t_0, and the final time-step, t_f.



(a) crack field, ϕ (b) X-displacement, u (c) Y-displacement, v

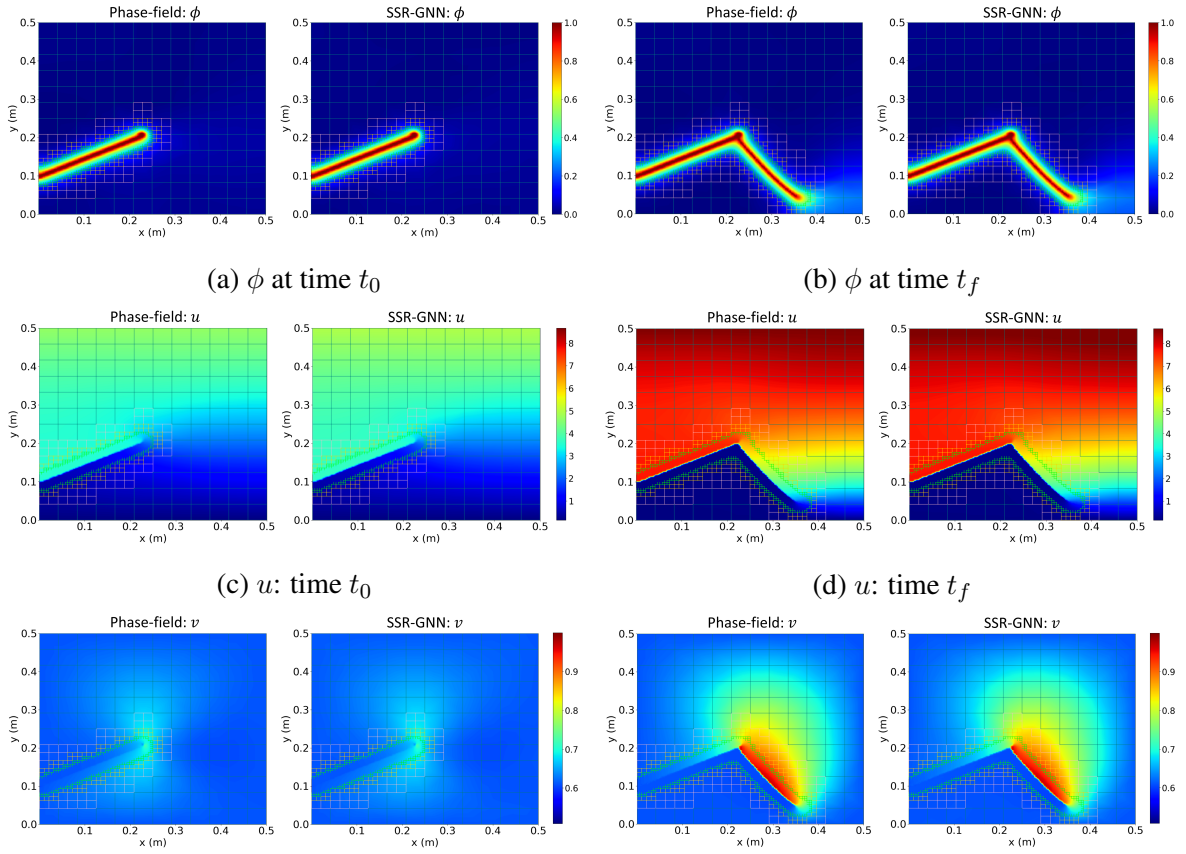
Figure 6.8: Average percentage errors across all simulation in the test dataset involving center crack cases subjected to tension for: a) Crack-field predictions, ϕ. b) X-displacement predictions, u. c) Y-displacement predictions, v.

propagated the simulation forward in time to near-complete material failure (t_f). We illustrate the resulting PF versus predicted crack field, x-displacement, and y-displacement at t_f in Figures 6.7b, 6.7d, and 6.7f, respectively. These results exhibit accuracy comparable to the left-edge crack cases, with predictions closely resembling those of the high-fidelity PF model at time-steps t_0 and t_f.

Furthermore, we conducted a quantitative analysis to corroborate these qualitative findings. For this, we obtained the average percent errors using equation (6.10) for each test simulation pertaining to center crack cases as shown in Figure 6.8. Across all test cases, the extended SSR model preserved high prediction accuracy, with crack field errors consistently below 0.125%, x-displacement errors below 0.25%, and y-displacement errors below 0.20%. These results underscore the SSR framework's high prediction accuracy achieved through TL, even with significantly reduced training datasets. Moreover, extending the SSR framework from left-edge cracks to center cracks enhances its predictive capabilities, particularly for cases where cracks propagate in both directions (i.e., left and right).

6.3.4 Shear load cases

In this next phase, we further implemented TL update steps to the extended SSR-based model acquired in Section 6.3.3. As detailed in Section 6.2.2, the integration of node features u_0, v_0 enables the framework to discern between tensile and shear loading scenarios. This entails setting $u_0 = \Delta u$ and $v_0 = 0$ for tensile loading, and $u_0 = 0$ and $v_0 = \Delta v$ for shear loading. Following a methodology similar to that for center cracks, we analyzed the extended SSR-based



(a) ϕ at time t_0

(b) ϕ at time t_f

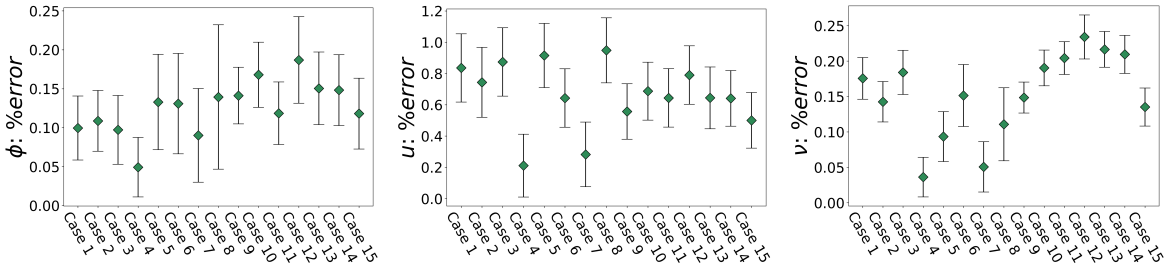
(c) u: time t_0

(d) u: time t_f

(e) v: time t_0

(f) v: time t_f

Figure 6.9: Comparison between the PF model and the SSR framework regarding the predicted evolution of the crack field, ϕ (a-b), x-displacement field, u (c-d), and y-displacement field, v (e-f), for a shear load test simulation at the initial time-step, t_0, and final time-step, t_f.



(a) crack field, ϕ (b) X-displacement, u (c) Y-displacement, v

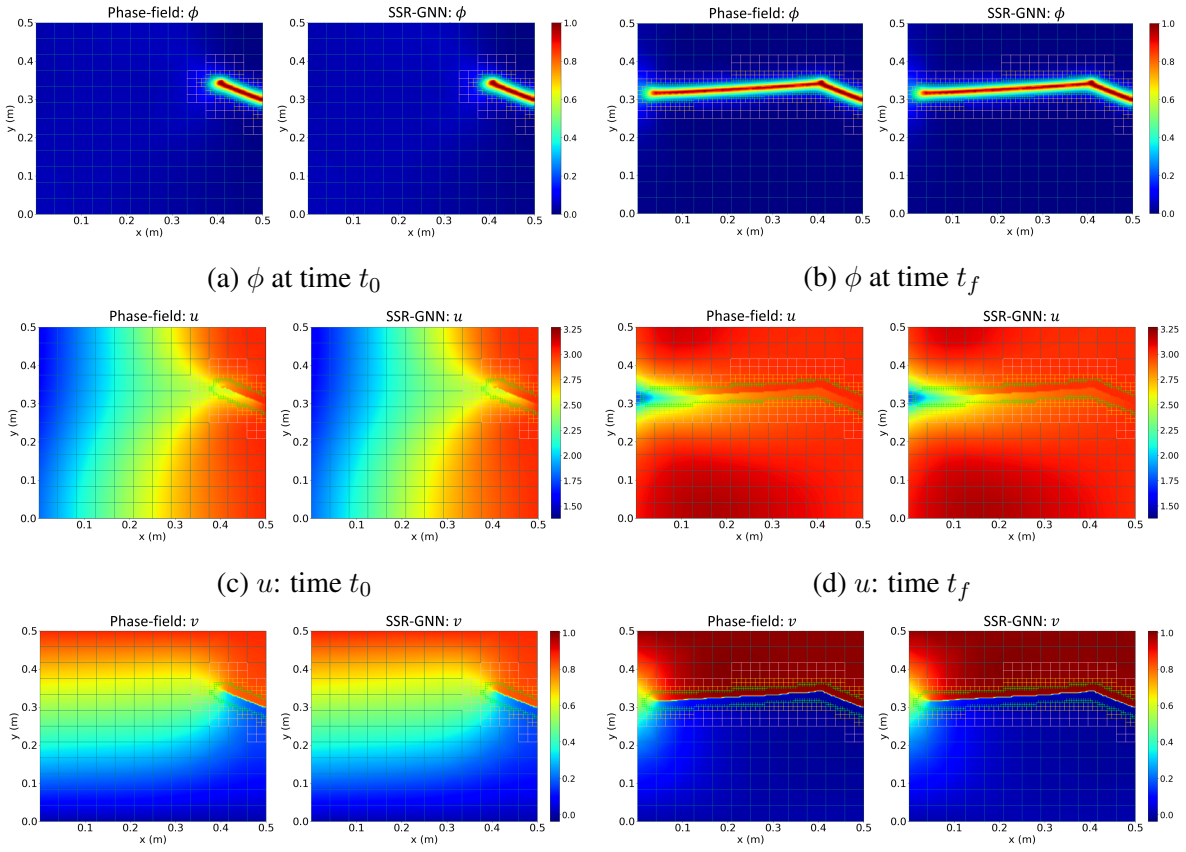
Figure 6.10: Average percentage errors across all simulations in the test dataset involving shear load cases for: a) Crack-field predictions, ϕ. b) X-displacement predictions, u. c) Y-displacement predictions, v.

model for shear load cases. This analysis involved predictions of crack field and displacement fields at both an initial time-step, t_0, and a future time-step near complete material failure, t_f.

For the initial time-step t_0, we compare the PF versus SSR predictions for crack field, x-displacement, and y-displacement fields in Figures 6.9a, 6.9c, and 6.9e, respectively. Similarly, for the final time-step t_f, Figures 6.9b, 6.9d, and 6.9f depict the PF versus SSR predictions for the crack field, x-displacement, and y-displacement fields, respectively. These visual representations demonstrate that the new extended SSR model adeptly captures shear load cases, yielding accurate predictions for both t_0 and t_f, virtually identical to the PF model. Additionally, we then followed the methodology outlined in Section 6.3.1 and computed the average errors for each test simulation to obtain a quantitative evaluation of the shear load cases. The resulting average errors are presented in Figure 6.10. The SSR framework also accurately predicts shear cases, with average percent errors for ϕ, x-displacement, and y-displacement under 0.25%, 1.20%, and 0.25%, respectively.

6.3.5 Right-edge crack cases

Finally, we trained the SSR framework towards right-edge crack cases subjected to tension. As outlined in Section 6.2.5 we generated the training and testing datasets for right-edge cracks by mirroring the cases involving left-edge cracks . We initially tested the trained SSR framework for right-edge cracks by simulating a randomly selected case from the test dataset. We show the predicted evolution of crack field from t_0 to t_f in Figures 6.11a. The predicted evolution of ϕ closely mirrors that of the PF predictions, indicating high qualitative prediction fidelity by SSR



(a) ϕ at time t_0

(b) ϕ at time t_f

(c) u: time t_0

(d) u: time t_f

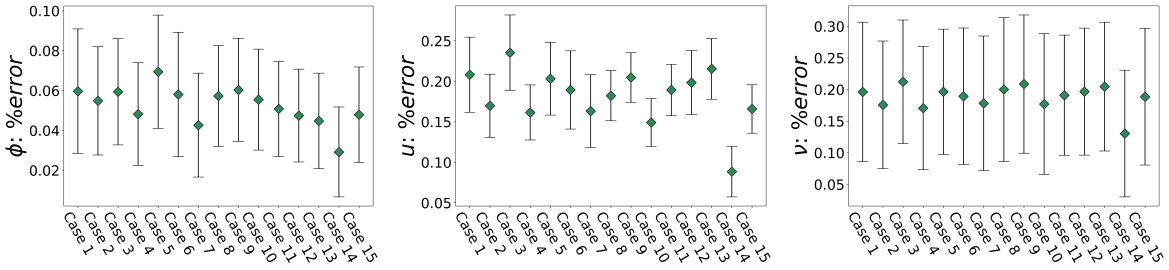
(e) v: time t_0

(f) v: time t_f

Figure 6.11: Comparison between the PF model and the SSR framework regarding the predicted evolution of the crack field, ϕ (a-b), x-displacement field, u (c-d), and y-displacement field, v (e-f), for a right-edge crack test simulation at the initial time-step, t_0, and final time-step, t_f.

for right-edge cracks. This observation is further supported by Figure 6.12a, where the average percent error in ϕ remains consistently below 0.10% for all test simulations.

For predictions related to x-displacement and y-displacement, we present a qualitative comparison for PF versus SSR in Figures 6.11c through 6.11f, respectively. Throughout the simulation, the SSR framework consistently demonstrates a high degree of prediction accuracy for the displacements. We show the resulting average percent errors in x-displacement in Figure 6.12b Examining the errors in x-displacement across all testing samples, we note that these errors remain comfortably under 0.25%. Similarly, in the case of y-displacement errors illustrated in Figure 6.12c, average percent errors consistently remain under 0.30%. These findings underscore the success of the SSR-based framework in simulating diverse problem configurations with remarkable accuracy through a series of sequential TL update steps.



(a) crack field, ϕ (b) X-displacement, u (c) Y-displacement, v

Figure 6.12: Average percentage errors across all simulation in the test dataset involving right-edge crack cases for: a) Crack-field predictions, ϕ. b) X-displacement predictions, u. c) Y-displacement predictions, v.

6.4 Conclusion

In summary, modeling complex multiphysics phenomena often necessitate computationally intensive methods, typically involving the solution of coupled multiphysics equations on intricate meshes. Recent advancements in ML, particularly mesh-based GNNs, offer a promising avenue to simulate such phenomena at a reduced computational cost. However, traditional mesh-based GNNs encounter challenges such as over-smoothing, particularly when dealing with fine meshes due to the high number of MP steps required. This study introduces a novel mesh-based multiscale GNN framework integrated with AMR to model multiphysics mesh-based problems with high accuracy, reduced number of MP steps, and accelerated simulation times. By systematically implementing sequential coarsening/upscaling operations which remove/add the highest level of mesh refinement at each step, the framework reduces the number of MP steps while maintaining high prediction accuracy and accelerating computational performance. Additionally, the framework incorporates state-of-the-art Graph Transformer MP networks along with skip-connectors linking coarsening and upscaling operations to prevent information loss. This technique provides new graphs with fewer resolution levels, facilitating the establishment of new distant connections and information transfer through broader local neighborhoods.

Given the high complexity and computational demands of multiphysics PF models, this study tests the multiscale GNN on PF fracture problems with near-singular operators and coupled equations. Initially focusing on single-edge notched systems (left-edge cracks) under tensile loading, the study develops and compares three coarsening/upscaling architectures (FSR, TSR,

and SSR). The SSR model, with the fewest operations, demonstrates the fastest simulation times while maintaining high accuracy.

Subsequently, TL is employed to extend the SSR framework for simulating various PF crack propagation problems, including center cracks, left-edge cracks under shear, and right-edge cracks, using only a fraction of the original training data. The developed SSR framework accurately predicts crack and displacement field evolution, with high accuracy above 98%, across different problem configurations. These results showcase the effectiveness of TL in leveraging smaller training datasets.

In conclusion, this work presents a novel mesh-based multiscale formulation that harnesses the computational efficiencies of AMR, algebraic multigrid scheme, and TL approaches. The resulting framework offers a powerful tool for simulating a variety of complex mesh-based multiphysics and engineering problems with integrated AMR, with high accuracy and accelerated simulation times.

6.5 Supplementary information

Additional information for (i) maximum % error analysis for the entire test datasets of left-edge crack, center crack, shear load, and right-edge crack cases, and (ii) generated sample simulations for each case can be found in https://github.com/rperera12/Adaptive-mesh-based-Multiscale-Graph-Neural-Network.

Chapter 7

Predicting critical impact velocity in heterogeneous explosives using Machine Learning techniques

7.1 Introduction and motivation

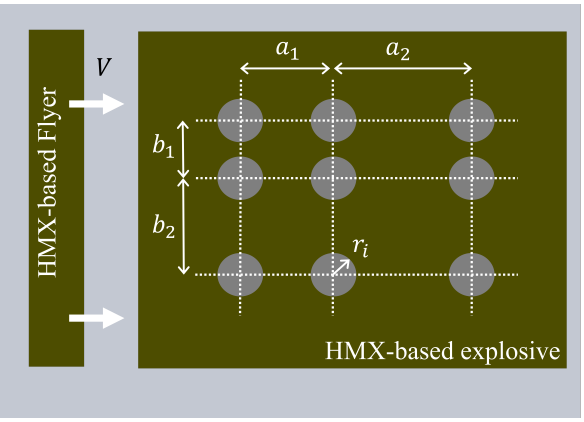
Heterogeneous Energetic Materials (HEMs) exhibit a complex array of microscale and macroscale defects, including pores, particles, voids, cracks, and grain interfaces, which give rise to localized temperature spikes known as hot spots [284, 285, 286]. These defects significantly influence the material's shock response and initiation behavior. For example, heterogeneous explosives involving elongated pores positioned parallel to the direction of the incoming shock result in higher sensitivity compared to circular pores[287]. Studies have also shown that the sensitivity of HEMs containing voids is directly dependent on the number of voids, their orientation, and their size distribution [288, 289].

Computational models have become indispensable tools for simulating impact and detonation in HEMs, providing insights into sensitivity properties that are challenging to obtain through experiments alone [290, 291, 292, 293, 294, 295, 296]. These models enable researchers to extend their understanding of the underlying physics and capture intricate material behaviors in regions and length scales where experimental techniques such as embedded gauge tests and large-scale gap tests [297, 298] are impractical. Recent advancements, such as the microstructure-explicit and void-explicit computational techniques introduced by Miller and Wei et al. [299, 300], have enabled the computation of probability equations for shock initiation thresholds in heterogeneous HEMs with varying grain size distributions. Despite their success, the computational costs and time associated with considering all possible heterogeneous

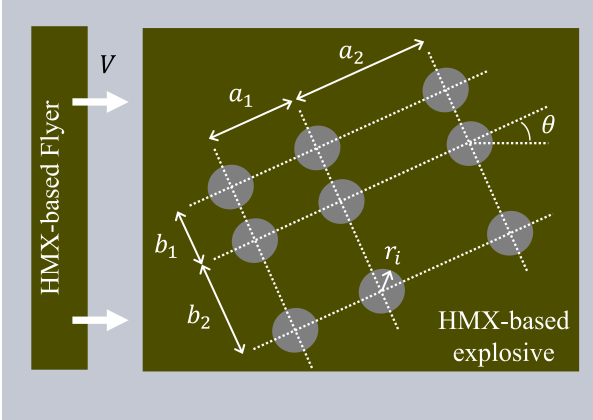
configurations remain prohibitively high for such models. For example, simulating the two-dimensional collapse of elongated voids in heterogeneous HMX involving pores required tens of thousands of processor hours on supercomputers [299]. Therefore, there is a pressing need for the development of novel computational techniques capable of rapidly predicting initiation in HEMs with diverse microstructural features. Such techniques would not only help reduce computational costs but also facilitate in-depth analyses to better understand the interactions between macro-scale pores in various configurations and their impact on initiation behavior.

Data-driven methods such as Machine Learning (ML) present a promising avenue to tackle these challenges. ML methods can provide high prediction accuracy comparable to traditional methods, while offering significant advantages in data processing efficiency and computational time [301]. Recent studies have extensively explored ML approaches for predicting chemical properties of energetic materials based on molecular configurations and properties [302, 303, 304, 305, 306, 307]. At the microscale and mesoscale, however, HEMs are typically characterized using techniques such as X-ray computed micro-tomography (x-ray μCT) or scanning electron microscopy (SEM), which result in image-based representations of the their heterogeneous material structure. To address this challenge through the use of ML methods, one common approach is to derive statistical descriptors of the microstructural features from these image-based representations [308]. Another prevalent image-based ML method involves regression Convolutional Neural Networks (CNNs). Unlike classification CNNs which focus on predicting binary or discreet numbers, regression CNNs are designed for predicting continuous numbers. Regression CNNs have been successfully applied in various domains, including detecting facial landmarks, estimating head or human pose landmarks, and predicting stock market trends [46, 47, 45]. In the context of ML methods for HEMs, CNNs have been utilized to predict properties such as the compressive strength of organic 2,4,6-triamino-1,3,5-trinitrobenzene (TATB) [309] and the peak stress in TATB materials with crystal microstructures [310]. Regression CNNs have also been recently employed by Casey et al. [311] to investigate the effects of microscale pore shape in HMX on the resulting critical impact velocity.

A more recent ML approach that has demonstrated strong performance in predicting various material properties in energetic materials is Graph Neural Networks (GNNs). The methodology



(a) Cartesian grid configuration



(b) Rotated configuration

Figure 7.1: Pore configurations comprising nine pores in a) Cartesian grid arrangement, and b) rotated grid for a PBX-9501 explosive subjected to impact by an impedance matched inert flyer plate.

of GNNs is described in detail in Chapters 3 - 6. Notably, in [312, 313], the skeletal molecular formula of energetic crystals was used to formulate graphs involving nodes and edges. Using the graph representations, the message-passing GNN D-MPNN was then employed for predicting properties such as detonation velocity, detonation pressure, density, and heat of formation. These works highlight the advantages of incorporating ML methods into HEM applications across different length scales, offering a means to mitigate the computational expenses linked with traditional approaches. However, although both mesoscale and microscale heterogeneities in HEMs have received considerable attention in previous ML-based research, the impacts of macroscale configurations (e.g., macroscale pores) have yet to be investigated using ML techniques.

Towards this objective, this study investigates the impact of multiple macroscale pores in heterogeneous PBX-9501 on their resulting critical impact velocity. Macroscale features in this context are defined within the range of 0.01 cm to 1.0 cm. Employing this approach, a total of 2556 distinct arrangements comprising 9 macroscale pores with varying sizes and positions were devised. Two types of configurations were devised: (i) utilizing a Cartesian grid where the radius of each pore, the vertical spacing between pore rows, and the horizontal spacing between pore columns were varied (see Figure 7.1a), and (ii) employing a rotated grid around a randomly determined angle, θ (see in Figure 7.1b). Simulations for each distinct pore configuration were

generated using the CTH hydrocode, with a bisection algorithm utilized to obtain the impact velocity resulting in bulk initiation, defined here as the critical impact velocity. Subsequently, two ML models were developed, each utilizing a different input representation for the pore configurations, in order to assess the most suitable ML approach for this problem. The first input representation employed a pixel-based binary image describing the bulk PBX-9501 material using black pixels, while the pore locations were represented using white pixels. Following a methodology similar to [311], the resulting pixel-based input was fed into a regression CNN model to predict the critical impact velocity. For the second input representation, a graph representation was developed, where nodes corresponded to the macroscale pore locations and edges connected the pores to all the remaining neighboring pores. The graph-based input method is especially suitable for GNNs, providing a more physical and spatial input of heterogeneous materials. To determine the best input representation for PBX-9501 explosives with different pore arrangements, the CNN and GNN models were evaluated on both Cartesian and rotated grid datasets. This study lays the groundwork for ML-driven models geared towards rapid predictions concerning the influence of various macroscale pore structures on shock sensitivity in HEMs.

7.2 Methods

7.2.1 Hydrocode simulations

The initial phase of this study involved creating an extensive dataset of two-dimensional macroscale simulations focusing on shock-induced detonation in heterogeneous HMX. PBX-9501 (comprising 95% HMX by weight) was selected as the explosive material, while the flyer plate was modeled using inert PBX-9501. Although typical flyer plate materials include steel or copper, PBX-9501 sufficed for this investigation to mitigate complex shock transition effects arising from impedance mismatch.

The hydrocode utilized a History Variable Reactive Burn (HVRB) model for PBX-9501, a Mie Grüneisen equation of state (EOS) for the flyer plate, and a SESAME EOS for air within the pores. It's worth noting that while the HVRB model is commonly calibrated via wedge

tests on pristine explosive samples, its application with pore-inclusive geometries may yield unrealistic shock behaviors. Future iterations of this work could fine-tune the ML models to incorporate more suitable reactive burn models and practical flyer plate materials.

The HVRB model is typically calibrated by performing wedge tests of pristine explosive samples which could lead to unrealistic shock behaviors when pores are added to the material geometry. The ML models developed in this work can be tuned in future work to more practical flyer plate materials and appropriate reactive burn models. As depicted in Figure 7.1, each simulation was defined by a distinct initial material structure featuring nine pores arranged in a Cartesian grid. From these CTH simulations, essential parameters such as temperature, pressure, particle velocity, and reaction progress variable λ were recorded over time along the explosive coordinates. To capture these variables, a 32 by 32 matrix of Eulerian tracers was employed along the explosive's coordinates, resulting in a training dataset comprising 2556 unique simulations. Notably, using CTH, approximately 250 compute-hours were required to run each simulation. Therefore, to generate the critical impact velocity of each distinct pore configuration (i.e., $\approx$ 10 simulations) a total of 2500 compute hours were required.

7.2.2 Problem geometry

The depicted sample simulation configuration in Figure 7.1 illustrates the macroscale spatial domain of PBX-9501, modeled using dimensions of $1.5'' \times 1.5''$, while dimensions of $1/8'' \times 1.5''$ were used to define the flyer plate. To vary the pore structure, adjustments were made to both the size of each pore and the spatial distribution between columns and rows of pores. Specifically, pore diameters (d_i) were chosen from $0.1, 0.2, 0.3, 0.4, 0.5$ cm, while horizontal and vertical spacing between columns and rows (a_1, a_2, b_1, b_2) were selected from $0.95, 1.90, 2.85$ cm.

Two distinct arrangements of pores were explored: (i) a Cartesian grid, as depicted in Figure 7.1a, and (ii) a rotated grid, obtained by randomly generating a rotation angle, θ, as illustrated in Figure 7.1b. In both arrangements, the radii of pores within a single column were maintained identical to minimize parameter complexity. For configuration (ii), the rotation angle (θ) ranged randomly from 0 to 2π. Using this approach, the range of impact velocity was considered spanning 500 to 1500 m/s.

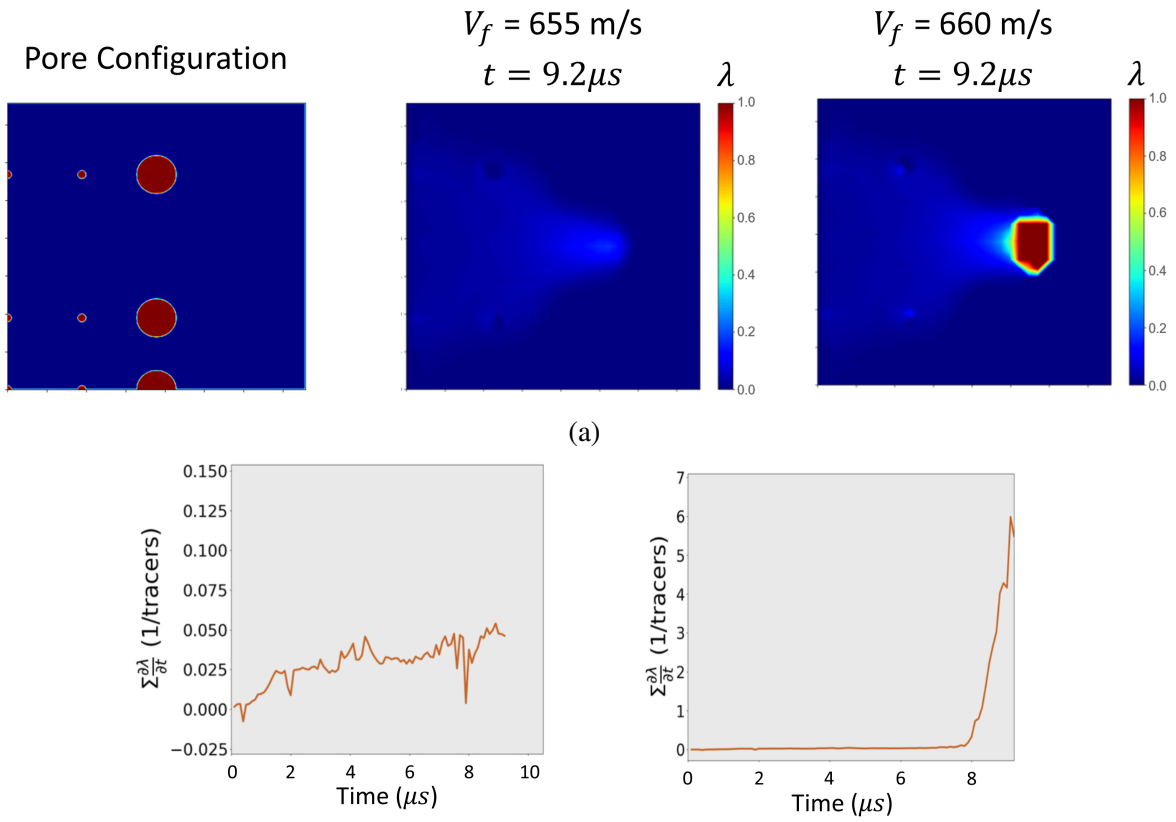


Figure 7.2: a) Distributions of reaction progress variable in a sample exposed to impact velocities of $655 m/s$ and $660 m/s$ at time $t = 9.2\mu s$. b) Rate of change of reaction progress variable over time in a sample under impact velocities of $655 m/s$ and $660 m/s$, spanning from $t = 0$ to $t = 9.2\mu s$.

While real-world HEMs exhibit random variations in pore quantity, shape, and position throughout the explosive, this study opted for simplified two-dimensional configurations featuring nine perfectly circular pores. Two-dimensional simulations may overlook rarefaction waves and shock interactions in the out-of-plane direction (i.e., result in higher pressures compared to three-dimensional analyses). However, this study is geared towards developing the methodology of a predictive ML framework. Thus, using two-dimensional domains involving 9 perfectly circular pores is sufficient to achieve this methodology. This framework lays the groundwork for extending to more intricate and realistic scenarios in the future.

7.2.3 Initiation criteria

As detailed in Section 7.2.1, every CTH simulation was conducted with 32 by 32 Eulerian tracers distributed throughout the explosive, facilitating the capture of the reaction progress variable, temperature, pressure, and particle velocity data for each time-step. The reaction progress variable, denoted as λ, spans the interval $[0, 1]$ and is defined at each cell i in the mesh as:

$$\lambda_i = \left\{ \frac{m_U}{m_R} \right\}_i \qquad i \in \{\mathcal{D}\}. \tag{7.1}$$

Here, m_U represents the unreacted mass, m_R represents the reacted mass, and $\mathcal{D}$ denotes the cells of the mesh. The critical impact velocity, V_c, was determined by computing the rate of change of the reaction progress variable with respect to time ($\frac{d\lambda}{dt}$) for each time-step in the simulation. Using this, the initiation criteria was defined as

$$\sum_i \left\{ \frac{d\lambda}{dt} \right\}_i \geq 1 \qquad i \in \{\mathcal{D}\}. \tag{7.2}$$

The threshold value of 1 was selected following an analysis of the rate of change of λ across various cases. An illustrative example of this analysis is depicted in Figure 7.2. Here, the same pore configuration depicted in the left-most image of Figure 7.2a, was subjected to impacts at velocities of $655m/s$ (middle image) and $660m/s$ (right-most image). The distribution of the reaction progress variable across the explosive at time $t = 9.2\mu s$ (i.e., when initiation occurred for the case of $660m/s$ impact velocity) is presented in Figure 7.2a. We note that for the $655m/s$ impact velocity case, the reaction progress failed to reach $\lambda = 1$ in any mesh cell within the explosive, indicating that initiation did not occur throughout the simulation. However, as illustrated in the right-most image of Figure 7.2a, for the $660m/s$ impact velocity case, the reaction progress variable reached values of 1 in multiple mesh cells of the PBX-9501 at this specific time-step ($t = 9.2\mu s$).

Furthermore, for both impact velocity cases of $655m/s$ and $660m/s$, we present the corresponding $\frac{d\lambda}{dt}$ in Figure 7.2b. To obtain a unit-less $\frac{d\lambda}{dt}$, we initially normalized $\sum(\lambda)$ by dividing by the total number of tracers used (i.e., 1024), resulting in $\sum(\lambda) \in \{0, 1\}$. Subsequently, we

made $\frac{d\lambda}{dt}$ unit-less by multiplying $\frac{d\lambda}{dt}$ times the total number of time-steps, $t_f = 15\mu s$, used for each simulation. For the case with $655m/s$ impact velocity, the magnitude of $\sum(\frac{d\lambda}{dt})$ peaked at approximately 0.055. However, upon increasing the impact velocity to $660m/s$ (i.e., only $5m/s$ higher), $\sum(\frac{d\lambda}{dt})$ notably surged to values exceeding 6.

Due to this considerable rise in $\sum(\frac{d\lambda}{dt})$ during initiation, we chose $\sum(\frac{d\lambda}{dt}) \geq 1$ as the threshold criterion to determine the critical impact velocity of each porous PBX-9501 sample. Notably, we conducted this analysis across 20 additional porous samples. Across all 20 samples, the porous PBX-9501 samples with the lowest and highest $\sum(\frac{d\lambda}{dt})$ where initiation did not occur, resulted in $\sum(\frac{d\lambda}{dt}) \in \{0.03, 0.09\}$. Additionally, the porous PBX-9501 samples where initiation occurred resulted in lowest and highest values of $\sum(\frac{d\lambda}{dt}) \in \{6, 15\}$. Thus, demonstrating that the threshold value of $\sum(\frac{d\lambda}{dt}) \geq 1$ is sufficient to define the initiation criteria for this problem.

Lastly, we emphasize that due to limitations in the selected chemical reaction models, unrealistic shock reflections were seen after the shock reached the right edge of the PBX-9501 samples. As a result, we used an additional constraint by obtaining the threshold criteria prior to the shock reaching the right edge of the domain.

7.2.4 Threshold velocity determination

Utilizing the criterion outlined in Section 7.2.3, we then developed a bisection algorithm for estimating the critical velocity of distinct porous sample. For each pore configuration, the bisection algorithm initially conducts CTH for the minimum and maximum impact velocities (defined as $V_{min} = 500m/s$, $V_{max} = 1500m/s$). The updated impact velocity is then calculated as:

$$V_{new} = \frac{V_{max} + V_{min}}{2} \quad . \tag{7.3}$$

If V_{new} results in the initiation of the explosive before the shock reaches the right edge of the sample, then we set V_{max} to V_{new}, and update V_{new} using Equation (7.3). Conversely, if V_{new} does not lead to initiation before the shock reaches the right edge of the domain, then we set V_{min} to V_{new}, and use Equation (7.3) to update V_{new}. This iterative process continues until V_{new}

is within $\pm 5 m/s$ of the final V_{min} and V_{max} values, where we choose the critical velocity as V_{max}.

Lastly, for both pore configurations (i.e., Cartesian grid and rotated configurations) we employed this bisection algorithm to construct the training, validation, and test datasets. The resulting training, validation, and test dataset involved 1000, 140, and 138 cases, respectively, totaling 1278 cases for each grid type (Cartesian grid and rotated configurations).

7.2.5 Regression Convolutional Neural Networks

To build upon previous research, which utilized 2D CNNs to forecast the critical threshold velocity necessary for triggering reactions in HMX crystals with a single embedded pore of varying shape [311], this study endeavors to broaden the scope by examining the impact of multiple macroscale pores. Here, we developed a 2D CNN to analyze how multiple pores, characterized by diverse sizes and spatial distributions, influence the resulting critical threshold velocity. In this pursuit, the critical threshold velocity obtained from each CTH simulation served as the target variable for training the regression CNN model. The input representation for the CNN comprised a pixel-based binary image, where black-colored pixels denoted the locations of the pores, while white-colored pixels represented the bulk PBX-9501 material. To standardize the inputs, we normalized and converted each image into a single binary channel, sized at 256 by 256 pixels.

The architecture of the regression CNN model comprised two 2D convolution layers. Each 2D convolution layer was followed using activation functions of Rectified Linear Unit (ReLU), and 2D maximum pooling operations. Subsequently, to flatten the output tensors from the 2D convolutions and generate a one-hot encoded feature vector, we employed a linear layer followed by a ReLU activation function. Finally, to make the final prediction we used an output linear layer.

For the first convolutional layer, we used 12 channels, 3 kernels, and a stride and padding of size 1. We then used 2 kernels and a stride of size 4 for the first maximum pooling layer. Similarly, for the second convolutional layer we utilized 24 channels, 5 kernels, and a stride and padding of size 1, followed by the second maximum pooling layer with equal configuration as

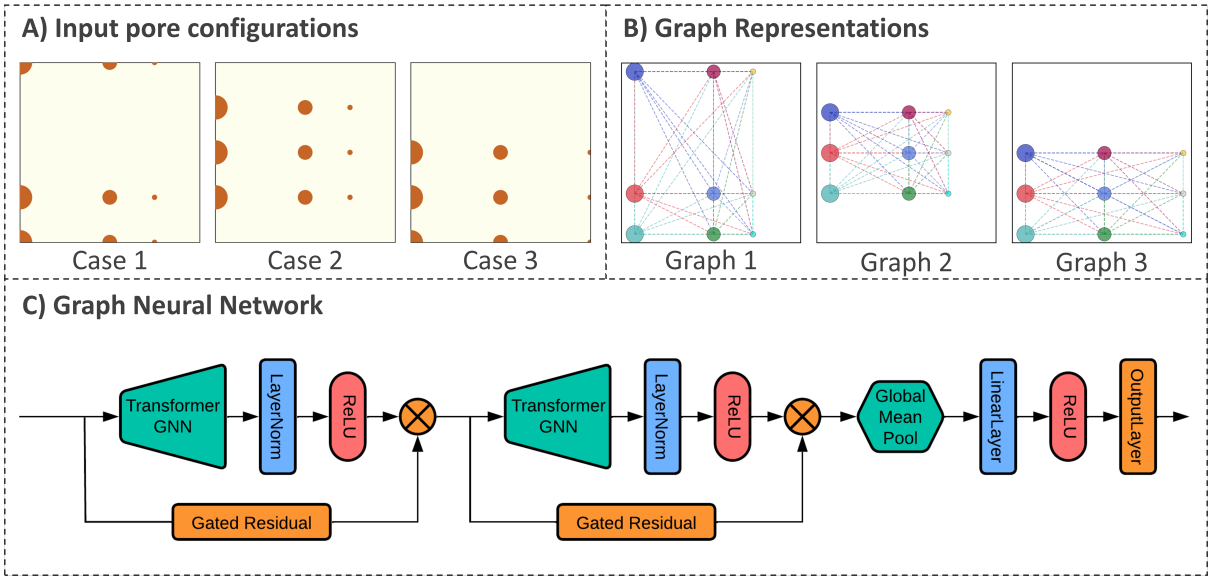


Figure 7.3: Flowchart illustrating the GNN framework: A) Initial configurations of three sample input pores. B) Generated graph representation for each sample, with nodes positioned at each pore and edges connecting all pores. C) Architecture of the developed Transformer GNN-based model.

the previous layer. Ultimately, leveraging this pixel-based (i.e., image-based) input configuration, the regression CNN model facilitated rapid predictions for the critical impact velocity.

7.2.6 Graph Neural Network

The next ML model established in this chapter involved a GNN to predict critical velocities of the porous PBX-9501 samples. In Figure 7.3, we present a flowchart for the developed GNN framework. First, we defined the graphs of the pore configurations as $\langle \mathbf{V}, \mathbf{E} \rangle$. Here, $\mathbf{V}$ encompassed all nine pores in the domains as nodes (or vertices), and $\mathbf{E}$ encompassed all the connecting edges in the graphs. This methodology presented a spatial graph-based representation of the pore configurations to the GNN model. For instance, we depict three distinct pore arrangements in Figure 7.3A, and show their resulting unique graph representations in Figure 7.3B. For each node in the domain, we defined the nodal features based on several factors, including their positions denoted as $\hat{\mathcal{P}}s$, their radii represented by $\hat{\mathcal{R}}s$, and their vertical

and horizontal distances to the boundaries of the PBX-9501 acceptor denoted as $\hat{\mathcal{L}}_s$.

$$\begin{aligned} \hat{\mathcal{P}}_s &= \{(x_s, y_s)\} \qquad \{s \in \mathbf{V}\}, \\ \hat{\mathcal{R}}_s &= \{r_s\} \qquad \{s \in \mathbf{V}\}, \\ \hat{\mathcal{L}}_s &= \{(d_{l_s}, d_{r_s}, d_{b_s}, d_{t_s})\} \qquad \{s \in \mathbf{V}\}, \\ \{v_s\} &= \left(\hat{\mathcal{P}}_s, \hat{\mathcal{R}}_s, \hat{\mathcal{L}}_s\right) \qquad \{s \in \mathbf{V}\}. \end{aligned} \tag{7.4}$$

Here, we defined the x- and y-positions of each sender node as (x_s, y_s), the pore radius of the sender node as r_s, and the distances from the the sender node to the left edge, right edge, bottom edge, and top edge of the acceptor using $(d_{l_s}, d_{r_s}, d_{b_s}$, and $d_{t_s})$, respectively. Furthermore, as depicted in Figure 7.3B, to define the edges within the domain (**E**), given the small scale of each graph comprising only 9 pores, we linked all sender nodes ($v_s \in \mathbf{V}$) to all other receiver nodes (i.e., neighboring pores defined as $v_r \in \mathbf{V}$).

We then defined the connections between edges by $(v_s, v_r, b_{sr}) \in \mathbf{E}$. Here, $b_{sr} \in 0, 1$ was a binary value indicating whether the sender node, v_s, and the receiver node, v_r, shared an edge connection. In this study, given that all sender nodes were linked to all neighboring nodes within the domain, b_{sr} was set to 1 for all pore samples. Furthermore, to supply the GNN model with additional spatial context, we introduced seven features for each edge in the domain. For the first edge feature, we included the effective angle from the sender to the receiver node with respect to the horizontal axis, denoted as θ_{sr}. We also included the vertical, horizontal, and the equivalent distances from the centroid of the sender node to the centroid of the receiver node, represented as $\delta\mathcal{Y}sr = yr - y_s$, $\delta\mathcal{X}sr = xr - x_s$, and $\delta\mathcal{D}sr = \sqrt{\delta\mathcal{X}^2 + \delta\mathcal{Y}^2}$, respectively. Lastly, to incorporate the influence of pore size in each pair-wise connection, we included the vertical, horizontal, and equivalent distances from the closest edge of the sender node to the closest edge of the receiver node, expressed as $\delta\mathcal{Y}^{'}sr = yr - y_s$, $\delta\mathcal{X}^{'}sr = x_r - x_s$, and $\delta\mathcal{D}^{'}_{sr} = \sqrt{\delta\mathcal{X}^2 + \delta\mathcal{Y}^2}$, respectively. We defined the resulting edge feature vector as:

$$\{e_{sr}\} = \left(b_{sr}, \theta_{sr}, \delta\mathcal{X}_{sr}, \delta\mathcal{Y}_{sr}, \delta\mathcal{D}_{sr}, \delta\mathcal{X}^{'}_{sr}, \delta\mathcal{Y}^{'}_{sr}, \delta\mathcal{D}^{'}_{sr}\right) \qquad \{s \in \mathbf{V}\} \; ; \; \{(s, r, b_{sr}) \in \mathbf{E}\}. \tag{7.5}$$

Upon generating the graph representation, we utilized it as the input to the GNN to predict the critical impact velocity. In Figure 7.3C, we show the architecture of the presented GNN model. We developed the message-passing graph blocks by utilizing the Graph Transformer network from [282]. We describe the Graph Transformers network in detail in Section 6.2.3 of Chapter 6.

As illustrated in Figure 7.3C, we made use of two Transformer message-passing GNN layers for the GNN architecture. Following each Transformer GNN layer, we update the graph weights by implementing a LayerNorm operation and a ReLU activation function. We designed these Transformer GNNs using 16 attention heads, each having a hidden dimension of size 32. To prevent over-smoothing during backward propagation, we also incorporated Gated Residual (aggregation skip connectors) to each message-passing network. Subsequently, we used a global average pooling layer to yield a graph-level output from the output from the Transformer message-passing networks. The global average pooling layer achieves this through averaging of the node features across all node dimensions. We then followed the global average pooling layer with a Linear operator and a ReLU activation function. Here, the Linear operator involved dimension of 32 hidden nodes. Finally, as shown in Figure 7.3C, the output layer consisted of a linear operator responsible for generating predictions of critical impact velocities.

7.3 Results and discussion

To illustrate the impact of multiple macroscale pores with diverse size and spatial distributions on the material's sensitivity, we conducted an analysis of the distribution of critical impact velocities (V_c) within the generated datasets. This analysis concentrated on understanding the range of V_c values across both the dataset comprising 1278 Cartesian grid configurations and the dataset comprising 1278 rotated configurations. Subsequently, we evaluated the performance of both the CNN and GNN developed models in predicting V_c for four randomly selected Cartesian grid arrangements from the test dataset. We assessed the resulting errors for each model on these simulations, and present the mean and standard distribution of errors across the entire test dataset to provide an overview of model performance. Furthermore, we repeated the aforementioned model error analyses on four randomly selected rotated pore arrangements. Lastly, we also obtained the CNN and GNN errors for the entire test dataset on the rotated

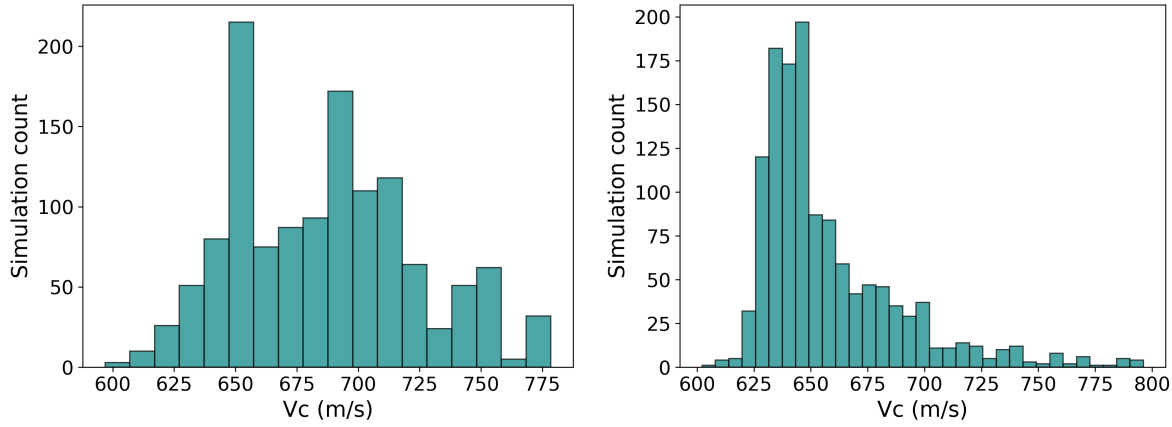
arrangements, offering insights into the predictive capabilities of the models across different configurations.

7.3.1 Distribution of critical impact velocities

The first step in our analysis involved examining the distribution of critical impact velocities (V_c) across the entire datasets of (i) Cartesian grid arrangements and (ii) rotated arrangements. This analysis aimed to determine the range of sensitivities exhibited by varying pore radius and spatial distributions, thereby highlighting the significant role played by different macroscale pore configurations in the material's sensitivity. In Figure 7.4, we illustrate two histograms for the distributions of V_c of both the Cartesian grid samples, and the rotated samples obtained from CTH simulations and the calculation of initiation using the criteria defined in Sections 7.2.3 and 7.2.4. We show the distributions of V_c for the 1278 Cartesian grid samples in Figure 7.4a, and the 1278 rotated pore samples in Figure 7.4b. These histograms provide insights into the variability of critical impact velocities across different pore configurations and orientations.

The analysis of critical impact velocities for Cartesian grid arrangements reveals a range from approximately 600m/s to 775m/s, underscoring the significant sensitivity dependence on the material's initial pore structure. Notably, the distribution illustrates varying occurrences across different V_c values, with 50 to 100 cases seen with values from 625m/s to 725m/s. Additionally, we observe the highest distribution of 160 and 210 cases with V_c values peaking at 650m/s and 690m/s, respectively, indicating distinct sensitivity patterns influenced by pore configurations. This highlights the necessity of developing efficient ML models capable of capturing these underlying patterns and relationships without relying on computationally demanding simulations. For instance, we note from Section 7.2.1 that each CTH simulation demanded approximately 250 compute-hours, thus, obtaining the critical velocity for each configuration required around 2500 compute-hours.

Furthermore, from Figure 7.4b the distribution of critical impact velocities for the rotated arrangements presents a distinct behavior compared to the Cartesian grid configurations. With V_c ranging from approximately 605m/s to 790m/s, the rotated cases exhibit a broader sensitivity range. However, the distribution appears skewed, with fewer instances observed at lower and



(a) Cartesian grid distribution

(b) Rotated distribution

Figure 7.4: Histograms displaying the distributions of critical impact velocities obtained from CTH for: a) 1278 samples of Cartesian grid pore arrangements. b) 1278 samples of rotated pore arrangements.

higher V_c values. Specifically, there are fewer than 5 cases for each V_c value between 605m/s and 615m/s, and less than 25 cases for V_c values between 705m/s and 790m/s. This skew in the V_c distribution of rotated pore arrangements may be attributed to the rotation angles used to rotate each original Cartesian grid configuration. From Section 7.2.1, we defined the rotation angles (θ) by randomly generating values of $\theta \in [0, 2\pi]$. Unlike the Cartesian grid which consistently had nine pores, this rotation angle generated cases where a subset of pores lie outside the PBX-9501 acceptor, thus, resulting in samples with fewer than nine pores. Consequently, in simulations with fewer than nine pores, the critical impact velocities may cluster around the higher end of the range, between 625m/s to 660m/s. We note that for a homogeneous acceptor (pristine PBX-9501), the critical velocity was determined as 646m/s, falling within this high-spike range. This suggests a direct relationship between the number (or density) of pores within the acceptor and the critical impact velocity, indicating a tendency towards homogeneous behavior when the number of pores within the acceptor is low. We also emphasize that although the rotated grid cases results in lower number of pores in some cases, these may serve as additional stress test for the ML models towards cases with varying number of initial pores.

A comprehensive analysis focusing on the impact of macroscale pore arrangements and critical impact velocity, considering factors such as pore density and spatial distribution, would be a valuable avenue for future research. By delving deeper into these relationships, a more

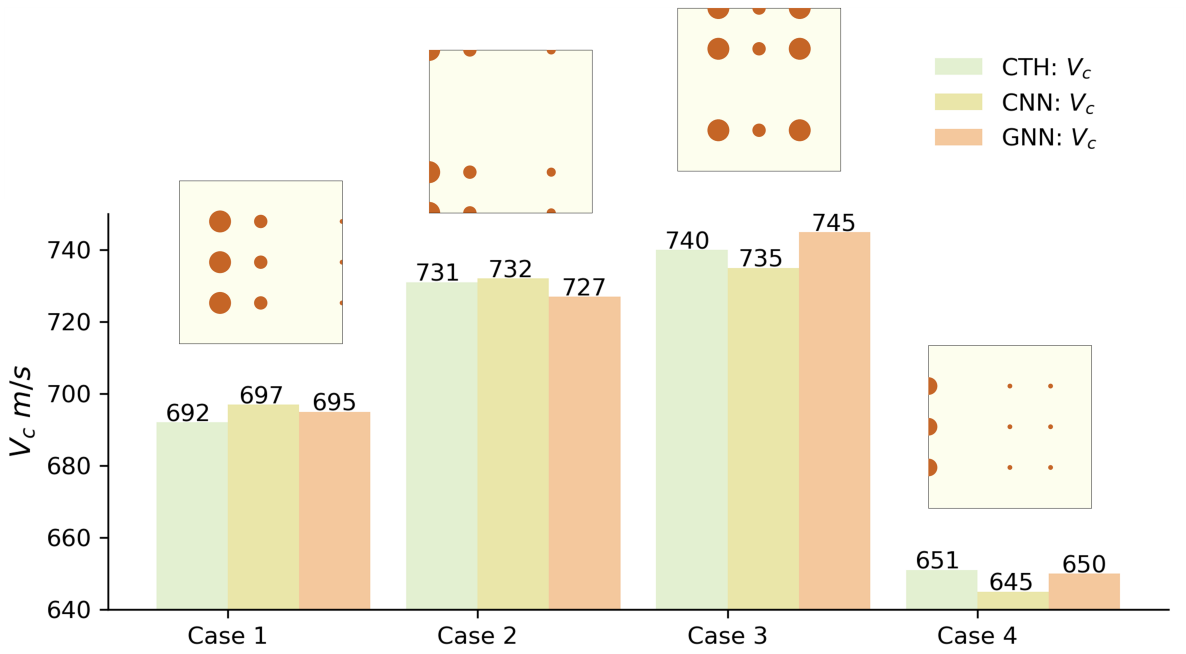


Figure 7.5: Comparison of critical impact velocities predicted by CTH and ML models for four pore configurations from the Cartesian grid test dataset.

nuanced understanding of how different pore characteristics influence material sensitivity can be gained. However, despite the variations introduced by the rotated pore configurations, in the context of developing accelerated predictive ML models for sensitivity of explosives, the key aspiration of this study is that the CNN and GNN models will effectively learn the underlying relations of these problems. By leveraging these ML models, it is hoped that the complex relationships between the number of pores, their varying radii, and spatial distribution can be accurately modeled and predicted, facilitating faster and more efficient assessments of material sensitivity.

7.3.2 Predictions on Cartesian grid configurations

We note from Section 7.2.3 that the test dataset for Cartesian grid configurations consisted of 138 unique samples. In Figure 7.5, we compare the critical impact velocities obtained from CTH versus the critical impact velocities predicted by the CNN and GNN ML models for four randomly chosen cases from the test set. Notably, despite not being trained on these specific simulations, the CNN and GNN models closely approximate the CTH velocities, with predicted V_c values falling within a range of $\pm 6 m/s$ of CTH. The largest deviation between CTH and the

predicted V_c value by the CNN model occurs in Case 4. Here, we observe that the hydrocode estimates $V_c = 651m/s$, and the CNN model predicts $V_c = 645m/s$.

In addition to the CNN model, we also assess the prediction accuracy of the Transformer GNN model for the same four Cartesian grid cases. Remarkably, similar to the CNN model, the Transformer GNN model demonstrated high accuracy in predicting the critical impact velocity of Cartesian grid configurations. As illustrated in Figure 7.5, the largest deviation occurred in Case 3, with only a $\pm 5m/s$ difference. We note that this error falls within the chosen threshold for the bisection algorithm of $\pm$5m/s. Consequently, the GNN model surpassed the CNN model in accuracy for these four randomly selected samples of Cartesian grid pore configurations.

We then compare the performance of the CNN model versus the Transformer GNN model by computing their errors on the entire test dataset (138 simulations). As depicted in Figure 7.6, we represent the errors for the CNN model using orange dots, and represent the errors for the GNN model as light blue. From the plot (Figure 7.6), it is evident that the Transformer GNN model consistently outperforms the CNN model for Cartesian grid pores, as it exhibits lower error rates across the entire test dataset. Specifically, the GNN model demonstrates an average error of $0.678 \pm 0.621\%$, while the CNN model shows an average error of $0.902 \pm 0.799\%$. We note that the highest error for the GNN model is obtained for Simulation No. 44, with approximately 2.9% error, while for the CNN model, it occurs for Simulation No. 96, with approximately 5.1% error. Although the GNN model generally exhibits better accuracy overall, it's essential to acknowledge that the CNN model can yield lower errors compared to the GNN model in certain scenarios, as observed in Case 2 from Figure 7.5. Ultimately, both models demonstrate robust performance to predict the critical impact velocities in Cartesian grid arrangements, with a superior overall performance achieved by the GNN model.

7.3.3 Predictions on rotated configurations

The subsequent analysis focused on evaluating the accuracy and errors of the CNN and GNN models for the rotated arrangements, considering the uneven distribution of V_c values observed in this arrangement, as discussed in Section 7.3.1. Similar to the evaluation conducted for Cartesian grid configurations, the initial step involved randomly selecting four samples from the

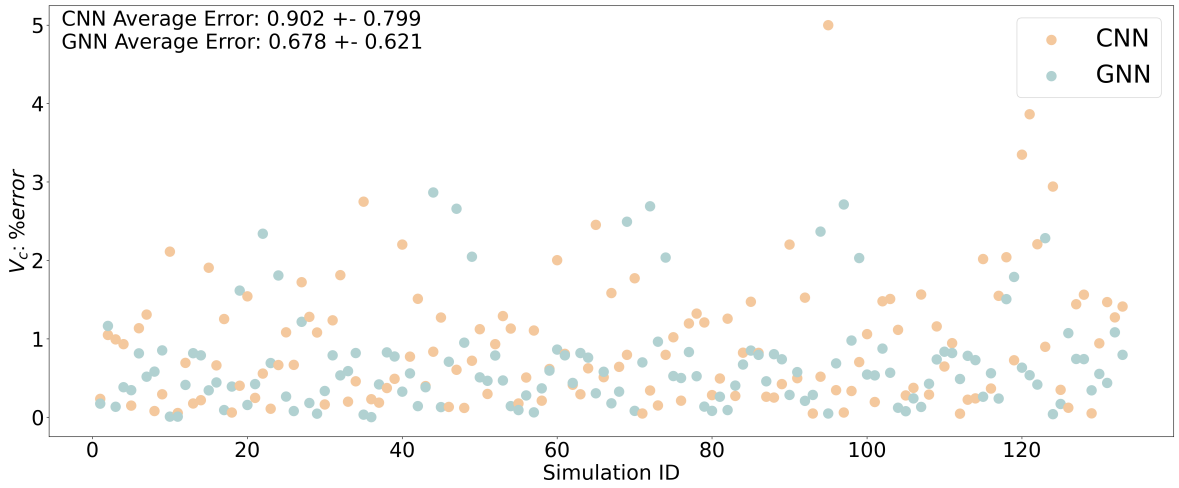


Figure 7.6: Percent error comparison between CNN (red) and GNN (blue) models across the entire test dataset of Cartesian grid comprising 138 simulations.

test set to obtain the critical impact velocities using CTH, CNN, and GNN models. In Figure 7.7, we illustrate the resulting critical impact velocities for these four random configurations. An intriguing observation is noted in Cases 2 and 4, where due to the rotation angle θ, configurations were formed with a column or row of pores positioned outside of the acceptor. The CNN model demonstrated high prediction accuracy for Cases 1, 3, and 4, achieving accuracies from 99.7% to 99.9%. However, the accuracy dropped for Case 2, reaching 96.8% due to the particular configuration resulting from the rotation. On the other hand, the GNN model exhibited lower accuracy for some of these four random cases compared to the CNN. We obtained the highest accuracies for the GNN for Cases 2, 3, and 4 ranging from 98.2% to 99.8%. Notably, the GNN model displayed lower accuracy for Case 1 at 93.58%, resulting in a difference of 45m/s from the CTH V_c. This analysis underscores the challenges posed by the rotated arrangements and highlights the varying performance of the CNN and GNN models in accurately predicting critical impact velocities for such configurations.

We further compare and analyze the performance of both ML models by computing and visualizing the overall percentage error across the entire test dataset. We show the resulting error for both the CNN and GNN on the entire test dataset of rotated pores in Figure 7.8. Contrary to the observations in Figure 7.6, where the CNN model exhibited higher error for the test dataset of Cartesian grid pores, the analysis of the entire test dataset of rotated grid revealed that the CNN model outperformed the GNN model. We obtained an average error for the GNN model

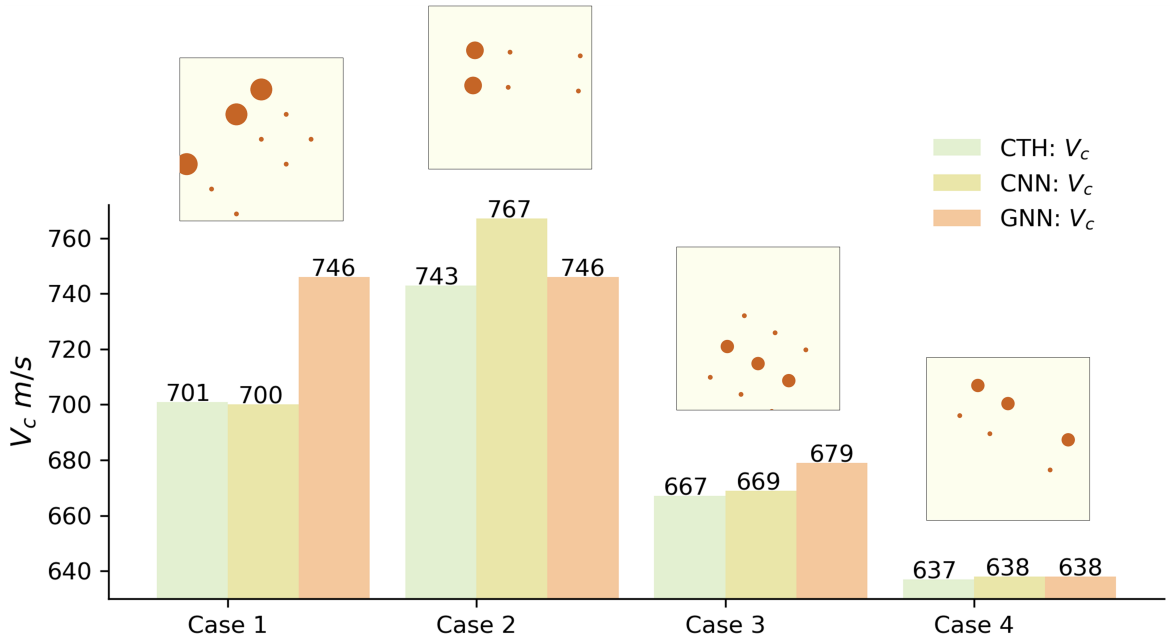


Figure 7.7: Comparison between critical impact velocities predicted by CTH and ML models for four pore configurations from the rotated pores test dataset.

over the entire test dataset of rotated pores of $0.967\% \pm 1.09\%$, while for the CNN model we obtained a slightly lower average error of $0.782\% \pm 1.107\%$. These results suggest that the CNN model may possess better adaptability in capturing the uneven distribution of V_c values observed in the rotated cases with greater accuracy compared to the GNN model. Despite the CNN's superior performance for the rotated arrangements, we note that both ML models maintained an overall error below 1%, indicating their efficacy in predicting critical impact velocities for various pore configurations.

7.4 Conclusions

To conclude, extensive research has focused on predicting how microscale and mesoscale pores affect material sensitivity using machine learning (ML) techniques. However, the impact of multiple macroscale pores remains largely unexplored. In this study, we highlight the pivotal role played by configurations comprising nine macroscale pores, each varying in size and position, in determining material sensitivity. To quantify material sensitivity, we devise a methodology to derive critical impact velocities. Employing a bisection algorithm alongside the CTH hydrocode, we generate a substantial dataset of critical impact velocities for diverse porous configurations.

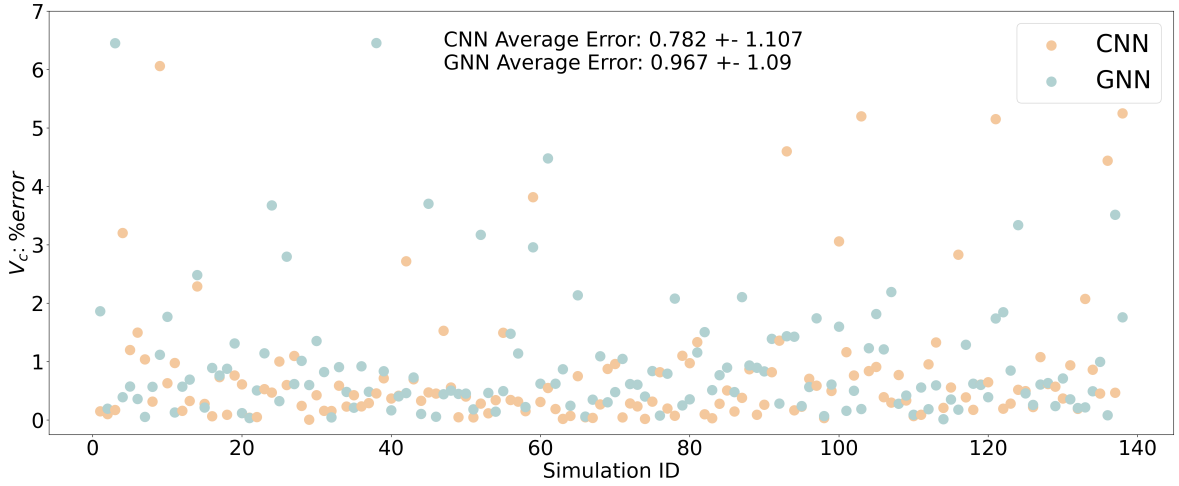


Figure 7.8: Percent error comparison between CNN (red) and GNN (blue) models across the entire test dataset of rotated grid comprising 138 simulations

Two distinct pore configurations were delineated: one employing a Cartesian grid of nine pores, and the other a rotated grid featuring nine or fewer pores. The radii of these pores and the inter-column and inter-row distances were systematically varied across each configuration.

We then developed two ML models to forecast the critical impact velocity based on the initial configuration. The initial model, a CNN, utilized a pixel-based binary image as input. This image represented the pore locations using black-colored pixels and the bulk material using white-colored pixels. Meanwhile, the second model adopted a Transformer GNN, which employed a graph-based input representation. This graph representation included nodes/vertices positioned at the pore locations and edges linking each pore to its neighboring pores. Subsequently, we assessed the average accuracy of both models across both Cartesian grid and rotated configurations of pores. Notably, in the case of Cartesian grid arrangements, the GNN model demonstrated superior performance over the CNN model, boasting an average percent error overall of $0.678 \pm 0.621\%$ compared to $0.902 \pm 0.799\%$ for the CNN. The GNN model also exhibited a maximum error of approximately 2.9%, while the CNN model showed a slightly higher maximum error of approximately 5.1%. Conversely, for rotated arrangements, the CNN model displayed higher accuracy than the GNN, with an average error overall of $0.782 \pm 1.107\%$ for the CNN and $0.967 \pm 1.09\%$ for the GNN. Interestingly, both models showcased similar maximum errors, with the CNN at approximately 6.0% and the GNN at 6.4%.

In conclusion, despite the higher errors observed in the rotated cases, both ML models achieved an average error below 1%, indicating overall strong performance. These findings underscore the significance of leveraging ML models for rapid material sensitivity predictions, eliminating the need for costly hydrocode simulations. Employing the CTH hydrocode to obtain critical impact velocities for each macroscale pore configuration is computationally intensive, demanding approximately 2500 compute-hours per pore configuration when utilizing the bisection algorithm iteratively. This study lays the groundwork for ML-driven approaches to expedite the understanding of how diverse macroscale pore structures influence shock sensitivity in HEMs. Future endeavors could enhance this methodology and the ML models by expanding the training dataset, refining model architectures, considering pore shape effects, and incorporating validated equations and reactive burn models into hydrocodes for more precise characterization of shock response in HEMs with several macroscale pores.

Chapter 8

Conclusions and Future Work

In this concluding chapter, we first present a summary for the entirety of the research detailed in this dissertation. We then describe some of the current limitations of each work. Lastly, we propose promising avenues for future research and extensions of the current ML methods.

8.1 Summary

8.1.1 Microstructure Characterization Framework

In the initial segment of this dissertation, our focus was on enhancing aerospace structures and materials, as well as preparing for future space exploration endeavors where emerging manufacturing technologies like 3D printing play a crucial role. With the inherent heterogeneity of microstructures in AM materials, it becomes imperative for 3D printing technologies to predict resulting material properties rapidly, based on the heterogeneous configurations. These capabilities enable the customization of material properties by controlling the initial defect arrangements. Hence, our first objective revolved around the development of an autonomous and optimized ML framework for expedited detection and extraction of microstructural defects in 3D printed materials, including particles, pores, grains, and GBs. The framework incorporated various ML models: a Classifier CNN for defect classification, a binary CEDN for particle or pore segmentation, an RGB CEDN for grain and GB segmentation, an object detection YOLOv5 network for predicting particle/pore size and location, and two Regression CNNs for grain size distribution histograms. A notable aspect of this work was the utilization of the Neural Architecture Search (NAS) algorithm, specifically DENSE, to optimize existing state-of-the-art

CEDN models while preserving high prediction accuracy. Through DENSE, we managed to reduce the GPU usage of the models by approximately 716 MB, roughly four times lower than state-of-the-art U-Net, thereby making our framework less CPU-intensive and easily portable. Ultimately, the developed ML framework yielded a remarkable fivefold acceleration in analysis time compared to conventional image-processing tools for microstructure characterization.

8.1.2 Microcrack-GNN

From Chapter 3 to Chapter 6, our focus shifted towards the development of novel ML frameworks utilizing GNNs to simulate the failure response of heterogeneous materials, offering faster turnaround times compared to conventional high-fidelity models. These endeavors were built upon the initial ML framework developed for microstructure characterization of 3D printed materials, where the extracted defects and features could be directly utilized to estimate the resulting material properties. Subsequently, these estimated properties could serve as inputs to computational models for simulating material failure behavior and defect propagation under various loads. Moreover, the integration of neural networks and graph theory in dynamic ML models like GNNs presents a promising avenue for enhancing the computational efficiency of existing high-fidelity computational models. The first dynamic GNN framework developed, Microcrack-GNN, was tailored to model the structural failure of a satellite panel featuring multiple initial microcracks. To achieve this, we addressed problems involving a brittle domain with varying numbers of initial microcracks ranging from 5 to 19. Introducing a novel graph representation for the initial system, we placed nodes at each crack-tip and established edges connecting each crack-tip to its nearest neighbors. The Microcrack-GNN framework demonstrated high accuracy in simulating stresses, with a maximum relative error of 4.80%, and exhibited good accuracy in simulating crack propagation, with maximum errors of 1.01% and 4.29%. Furthermore, a significant contribution of Microcrack-GNN was its remarkable computational speed-up in simulation time, achieving up to 20 times faster performance compared to a high-fidelity XFEM model. This work showed that integrating graph theory along with GNNs shows a promising approach for speeding-up existing high-fidelity fracture mechanics models in materials with initial defects.

8.1.3 ACCURATE

We then proceeded to enhance and broaden the capabilities of Microcrack-GNN to accommodate new problem-specific inputs. This endeavor involved implementing a sequence of five TL update steps to the pre-trained Microcrack-GNN model, considering scenarios with arbitrary domain sizes, crack lengths, orientations, and shear loads. Notably, each TL update step substantially reduced the required size of training data to merely 20 simulations, a stark contrast to the 960 simulations needed for Microcrack-GNN. The resulting GNN framework, named ACCURATE, acquired generalized knowledge of fracture mechanics pertinent to the problem, enabling accurate simulation of stress evolution and crack propagation for previously unseen cases featuring distinct domain dimensions, crack lengths, and orientations, under both tensile and shear loading conditions. Furthermore, an impressive feature of the ACCURATE framework was its significant improvement in simulation speed, achieving a remarkable speed-up of 2 orders of magnitude, approximately 200 times faster than the high-fidelity XFEM model. Ultimately, this development showcased the efficacy of leveraging ML techniques like GNNs and TL in crafting rapid reduced-order computational models, trainable with minimal datasets, and adaptable to unseen scenarios with new initial conditions.

8.1.4 ADAPT-GNN

Next, we delved into mesh-based GNNs, a different breed of dynamic GNNs where the mesh configuration itself serves as the graph representation for the model. Modeling complex multiphysics phenomena often entails employing computationally intensive methods that require solving coupled multiphysics equations on a mesh. While Microcrack-GNN and ACCURATE showcased accelerated emulation times and high accuracies compared to XFEM, their predictions were confined to nodes in the graph, specifically crack-tips. In contrast, mesh-based GNNs offer predictions across the entire domain akin to conventional computational models. Thus, we embarked on crafting an adaptive mesh-based GNN framework, ADAPT-GNN, adept at emulating nultiphysics problems such as phase field fracture models for single-edge notched cracks under tensile loading conditions. A standout characteristic of ADAPT-GNN was its fusion

of AMR with ML, harnessing the computational efficiencies of both techniques. This innovative framework facilitated simulation of x- and y-displacement fields, followed by the scalar damage field and stress field at subsequent time-steps. Leveraging this dynamic mesh-based graph implementation resulted in significant simulation speed-ups, up to 36 times faster compared to a traditional phase field fracture model, while maintaining high prediction accuracies of approximately 98%. Ultimately, phase field fracture models, particularly for crack propagation, pose formidable computational challenges. Through this endeavor, we showcased the development of an adaptive mesh-based GNN approach capable of accurately predicting phase field fracture models of crack propagation, while achieving noteworthy computational speed-ups.

8.1.5 Multiscale GNN framework

Furthermore, while ADAPT-GNN showed promise in simulating multiphysics problems at a reduced computational cost, it encountered challenges with over-smoothing, especially when dealing with fine meshes requiring a high number of required message-passing steps. To address these challenges, the subsequent project introduced a mesh-based multiscale GNN framework with adaptive mesh refinement (AMR) for simulating multiphysics problems with improved efficiency, accuracy, and performance. Drawing inspiration from recent studies integrating algebraic multigrid schemes with GNNs, this novel approach utilized a multigrid formulation involving sequential coarsening and upscaling operations. By dynamically adding or removing the highest mesh refinement resolution level at each step, the framework generated new graphs with fewer mesh resolution levels, fostering increased connectivity and larger local neighborhoods. The effectiveness of the multiscale GNN framework was demonstrated through simulations of phase field crack problems featuring near-singular operators and coupled equations. Various problem-specific inputs and initial conditions, such as left-edge crack systems, center crack systems, and right-edge crack systems under tension, and left-edge crack systems under shear, were explored, with TL employed to reduce the required training data size to only 15 samples per new input. Results showcased significant acceleration in simulation time compared to both the phase field model and ADAPT-GNN, while maintaining high prediction accuracies exceeding 98% for displacement fields and crack fields across all cases. Ultimately,

this work introduced a versatile mesh-based multiscale formulation leveraging AMR, mirroring conventional iterative multigrid schemes, and integrating TL, paving the way for efficient simulation of a wide range of complex mesh-based engineering and multiphysics problems with exceptional accuracy and performance.

8.1.6 Predicting sensitivity of HEMs

Finally, another critical aerospace challenge, particularly in military applications exploring additive manufacturing, lies in the 3D printing of novel explosives. The resulting structures often contain defects like macroscale pores, significantly influencing their sensitivity and shock response. While past research has delved into predicting material sensitivity due to microscale and mesoscale pores using ML techniques, the impact of multiple macroscale pores had remained unexplored. Thus, the subsequent study in this dissertation focused on investigating the effects of macroscale pores on shock sensitivity. We generated various distinct material configurations featuring nine macroscale pores of different sizes and positions and determined their critical impact velocities using the CTH hydrocode. Subsequently, we developed two ML models to predict these critical impact velocities: (i) a CNN utilizing pixel-based binary images as inputs, and (ii) a Transformer GNN framework employing a graph-based representation of nodes and edges for the pores. While the GNN model exhibited higher overall prediction accuracy compared to the CNN model, both models maintained an average accuracy exceeding 99%. These findings underscore the significance of ML models in swiftly and accurately predicting material sensitivity without relying on computationally expensive hydrocodes. Leveraging the CTH hydrocode for obtaining critical impact velocities for each potential macroscale pore configuration necessitated approximately 2500 compute-hours per configuration, whereas the GNN and CNN models required less than 1 second per configuration. Ultimately, this work lays the foundation for ML-guided models facilitating accelerated comprehension of how diverse heterogeneous structures influence shock sensitivity in HEMs.

8.2 Limitations

While this dissertation introduced novel ML techniques for rapid microstructure characterization of AM materials and accelerated emulation of dynamic fracture in materials with defects, several limitations merit consideration, beyond those already discussed in each chapter. A general limitation across all works involves threats to validity. Each ML framework presented was empirically compared with conventional computational tools based on their required wall time. For example, the microstructure characterization framework exhibited analysis times approximately five times faster than the standard image processing tool. Additionally, Microcrack-GNN and ACCURATE reduced simulation times significantly compared to XFEM-based models, as did mesh-based ADAPT-GNN and Multiscale GNN compared to serial execution of the phase field model. While empirical measures of wall time sufficed for this dissertation, future work should include a more comprehensive analysis of the algorithms' computational complexity. Potential approaches may include employing Big O notation, Big-omega notation, or Big-theta notation for algorithm analysis. Furthermore, the empirical comparisons of wall time assumed that the ML models were fully trained prior to usage. It's worth noting that if these models require training from scratch, conventional tools may outperform the developed ML models significantly. Such considerations should be addressed in future research to provide a more comprehensive understanding of the relative performance of ML-based approaches compared to traditional methods.

Another general limitation of these works is the inherent nature of supervised ML techniques, which necessitate pre-labelled data for training the models. While this thesis endeavors to reduce the computational costs of existing conventional computational tools, it is noteworthy that extensive use of these tools was required to gather the training datasets for each ML framework. Thus, the development of new conventional tools remains crucial, and this dissertation does not seek to supplant them, but rather to complement and expedite their applications.

Next, a key limitation of the microstructure characterization framework discussed in Chapter 2 pertains to the scope of materials, types of defects, and length scales on which the framework was trained. Specifically, the framework focused solely on AM metallic materials and addressed

defects such as pores, particles, and GBs at the microscale level. This limitation raises pertinent questions such as whether the framework can identify defects in transparent materials and if it can be adapted for different tasks like detecting craters and flood plains from satellite images. Future work should involve extensive testing of the framework across new non-metallic materials, varied size regimes, and different types of problems to address these inquiries effectively.

For Microcrack-GNN and ACCURATE, an important limitation of both frameworks is their reliance on a graph representation where nodes are situated solely at each crack-tip. This representation constrains the frameworks to predict crack propagation and stresses exclusively at these crack-tips, unlike XFEM which offers insights throughout the entire domain. Furthermore, the models are tailored solely for 2D brittle materials, disregarding 3D scenarios where crack bifurcation might occur in ductile materials, necessitating the representation of two or more nodes to capture the formation of two crack-tips in three-dimensional spaces. As a result, the current graph representation in these frameworks is limited to 2D brittle materials and crack-related problems, highlighting the need for a different graph formulation in future endeavors to accommodate diverse defect types and three-dimensional materials, particularly those exhibiting ductile behavior.

Ultimately, moving to mesh-based GNNs, these models showed to provide accurate predictions for multiphysics problems such as phase field fracture models. However, it's worth noting that the developed frameworks, although formulated based on the phase field energy functional, still do not provide a physics-based interpretation of the results, unlike conventional computational models. This is a key limitation in supervised ML models where interpreting their predictions remains a "black box". Additionally, the developed mesh-based GNNs did not include any information regarding boundary conditions. These limitations can be addressed in future work using multiple types of nodes for the boundary and material region, and by investigating into physics-informed neural networks and interpretable models.

Ultimately, while mesh-based GNNs have demonstrated accurate predictions for multi-physics problems like phase field fracture models, it's essential to acknowledge their limitations. Unlike conventional computational models, the developed frameworks lack a physics-based interpretation of results, characteristic of supervised ML models where predictions remain a

"black box." Additionally, these GNNs are restricted to two-dimensional domains and do not incorporate information on boundary conditions, which could affect their applicability in more complex and realistic scenarios. Addressing these limitations in future work could involve incorporating three-dimensional realistic mesh-based domains, various node types for boundary and material regions, and exploring physics-informed neural networks and interpretable models to enhance the interpretability of results.

8.3 Future Work

The ML microstructure characterization framework presented in this dissertation opens up several avenues for future research to further enhance its applicability in aerospace and beyond. First, the framework can be expanded to detect and characterize new types of materials beyond AM metals, encompassing various length scales and material compositions. Additionally, extending the framework to predict multiple defects within the same image, using techniques like the object detection YOLOv5 network, could enhance its versatility. Moreover, integrating additional ML models to predict material properties based on the extracted features would provide a more comprehensive understanding of material behavior. This generalized structure-property extraction capability could then be utilized as input for computational models to simulate material failure responses accurately. Ultimately, while the framework was initially developed for defect characterization, its techniques hold potential for applications beyond structural mechanics, such as crater extraction in satellite imagery, which could be explored in future studies.

Next, the GNN frameworks developed to accelerate simulation time also open up numerous avenues for future projects, and I will discuss just a handful of them here. The Microcrack-GNN and ACCURATE frameworks offered a simplified graph representation tailored towards XFEM problems which only considered cracks in two-dimensional brittle materials. These frameworks can be extended in future work towards more complex and realistic three-dimensional domains involving ductile materials. A possible approach to account for dynamic effects from ductile fracture such as crack bifurcation, is to develop a dynamic graph representation where in the event of crack bifurcation, an additional node and edge is added to the graph to account for

the newly formed crack-tip. Additionally, a similar methodology can be applied to extend the framework for different types of defects such as voids. Lastly, the ML optimization algorithm, DENSE, can be implemented to these models iteratively at each training time-step. The DENSE algorithm can be used for optimizing the models' architecture, training parameters, number of simulations required for TL, and number of initial layers to be used for TL.

For future projects involving mesh-based GNNs, a crucial area of exploration is the development of interpretable models to elucidate the predictions made by these frameworks, thereby transcending the "black box" nature of current models. One avenue to achieve this is through the integration of physical laws and constraints into the model architecture, allowing for the extraction of physics knowledge from predictions. One approach could involve formulating a physics-informed loss function based on the energy functional of the phase field fracture model [314]. The developed mesh-based GNNs predict quantities directly included in the phase field energy functional such as scalar damage field, x-displacement field, and y-displacement field. With these predictions, we can then directly informing the loss function based on the residual from the predicted energy functional and the actual energy functional. Another approach involves introducing inductive bias into the model architecture, embedding governing physics and constraints directly into the structure of the ML model [315]. The graph representation used in the developed mesh-based GNNs provides an ideal platform for incorporating this inductive bias formulation where even the gradients of the crack and displacement fields are included as edge features, enabling enforcement of governing differential equations and constraints for each node and edge feature in the mesh. Ultimately, an interpretable mesh-based GNN approach would facilitate the identification of significant node and edge features and their impact on the resulting physics of the problem, offering deeper insights into material behavior and failure mechanisms.

Moreover, in future investigations, a critical aspect to address for all dynamic GNNs discussed in this dissertation is their scalability concerning domain size, defect quantity, and mesh resolution. The ability to train models on smaller domains and then apply them to larger-scale systems with more defects and finer meshes would significantly enhance their computational efficiency. Multiscale GNNs may prove pivotal in this regard, as their approach

facilitates the learning of underlying physics or patterns across large graphs from individual smaller subgraphs. These proposed future endeavors aim to establish a generalized, scalable, and interpretable ML framework for accelerating simulation time across various engineering and physics mesh-based problems. The mesh-based GNNs developed in this dissertation provide a versatile methodology that can be readily extended to address other multiphysics challenges, such as simulating detonation or shock response, fluid dynamics, deflagration, and material fragmentation. Furthermore, this methodology can be expanded to encompass three-dimensional problems, incorporating multiple types of initial defects akin to those found in AM materials, as well as diverse node types to accommodate various boundary conditions.

Ultimately, each study in this thesis delved into the realm of structure-property relations, spanning from the creation of an ML microstructure characterization framework for defect extraction and material property estimation to ML mesh-based GNNs for expediting fracture simulation in materials with initial defects. Another crucial frontier for future exploration lies in reversing this process to achieve optimal material design: starting with desired material properties and dynamic failure behavior, then predicting the corresponding initial defect arrangement needed to attain such behavior. This approach would ultimately revolutionize the development of space structures and military materials by tailoring material defects on-the-fly to withstand specific anticipated loads. Generative ML models like Generative Adversarial Networks (GANs) and Reinforcement Learning (RL) models may offer promising avenues for tackling this task in future research endeavors.

References

[1] Z. Wang, W. Yang, Q. Liu, Y. Zhao, P. Liu, D. Wu, M. Banu, and L. Chen, "Data-driven modeling of process, structure and property in additive manufacturing: A review and future directions," *Journal of Manufacturing Processes*, vol. 77, pp. 13–31, 2022.

[2] R. Perera, D. Guzzetti, and V. Agrawal, "Optimized and autonomous machine learning framework for characterizing pores, particles, grains and grain boundaries in microstructural images," *Computational Materials Science*, vol. 196, p. 110524, 2021.

[3] R. Perera, D. Guzzetti, and V. Agrawal, "Graph neural networks for simulating crack coalescence and propagation in brittle materials," *Computer Methods in Applied Mechanics and Engineering*, vol. 395, p. 115021, 2022.

[4] R. Perera and V. Agrawal, "A generalized machine learning framework for brittle crack problems using transfer learning and graph neural networks," *Mechanics of Materials*, vol. 181, p. 104639, 2023.

[5] R. Perera and V. Agrawal, "Dynamic and adaptive mesh-based graph neural network framework for simulating displacement and crack fields in phase field models," *Mechanics of Materials*, vol. 186, p. 104789, 2023.

[6] R. Perera and V. Agrawal, "Multiscale graph neural networks with adaptive mesh refinement for accelerating mesh-based simulations," *arXiv preprint arXiv:2402.08863*, 2024.

[7] X. bin SU, Y. qiang YANG, P. YU, and J. feng SUN, "Development of porous medical implant scaffolds via laser additive manufacturing," *Transactions of Nonferrous Metals Society of China*, vol. 22, pp. s181 – s187, 2012.

[8] A. Gasser, G. Backes, I. Kelbassa, A. Weisheit, and K. Wissenbach, "Laser additive manufacturing," *Laser Technik Journal*, vol. 7, no. 2, pp. 58–63, 2010.

[9] R. Russell, D. Wells, J. Waller, B. Poorganji, E. Ott, T. Nakagawa, H. Sandoval, N. Shamsaei, and M. Seifi, "3 - qualification and certification of metal additive manufactured hardware for aerospace applications*," in *Additive Manufacturing for the Aerospace Industry* (F. Froes and R. Boyer, eds.), pp. 33 – 66, Elsevier, 2019.

[10] R. van Woensel, T. van Oirschot, M. Burgmans, M. Mohammadi, and K. Hermans, "Printing architecture: An overview of existing and promising additive manufacturing methods and their application in the building industry," *The International Journal of the Constructed Environment*, vol. 9, pp. 57–81, 06 2018.

[11] B. L. DeCost and E. A. Holm, "Characterizing powder materials using keypoint-based computer vision methods," *Computational Materials Science*, vol. 126, pp. 438 – 445, 2017.

[12] L. Murr, "A metallographic review of 3D printing/additive manufacturing of metal and alloy products and components," *Metallography, Microstructure, and Analysis*, vol. 7, pp. 103–132, 2018.

[13] L. Sheridan, O. E. Scott-Emuakpor, T. George, and J. E. Gockel, "Relating porosity to fatigue failure in additively manufactured alloy 718," *Materials Science and Engineering: A*, vol. 727, pp. 170 – 176, 2018.

[14] T. Furumoto, A. Koizumi, M. R. Alkahari, R. Anayama, A. Hosokawa, R. Tanaka, and T. Ueda, "Permeability and strength of a porous metal structure fabricated by additive manufacturing," *Journal of Materials Processing Technology*, vol. 219, pp. 10 – 16, 2015.

[15] J. Choren, S. Heinrich, and B. Silver-Thorn, "Young's modulus and volume porosity relationships for additive manufacturing applications," *Journal of Materials Science*, vol. 48, pp. 5103–5112, 08 2013.

[16] S. Tammas-Williams, P. Withers, I. Todd, and P. Prangnell, "The influence of porosity on fatigue crack initiation in additively manufactured titanium components," *Scientific Reports*, vol. 7, 08 2017.

[17] C. Todaro, M. Easton, D. Qiu, D. Zhang, M. Bermingham, E. Lui, M. Brandt, D. Stjohn, and M. Qian, "Grain structure control during metal 3D printing by high-intensity ultrasound," *Nature Communications*, vol. 11, p. 142, 01 2020.

[18] M. Chapetti, H. Miyata, T. Tagawa, T. Miyata, and M. Fujioka, "Fatigue strength of ultra-fine grained steels," *Materials Science and Engineering: A*, vol. 381, pp. 331–336, 09 2004.

[19] J. Slotwinski, E. Garboczi, P. Stutzman, C. Ferraris, S. Watson, and M. Peltz, "Characterization of metal powders used for additive manufacturing," *Journal of research of the National Institute of Standards and Technology*, vol. 119, pp. 460–493, 09 2014.

[20] A. Rabbani, S. Jamshidi, and S. Salehi, "An automated simple algorithm for realistic pore network extraction from micro-tomography images," *Journal of Petroleum Science and Engineering*, vol. 123, pp. 164 – 171, 2014. Neural network applications to reservoirs: Physics-based models and data models.

[21] G. Lo Re, F. Lopresti, G. Petrucci, and R. Scaffaro, "A facile method to determine pore size distribution in porous scaffold by using image processing," *Micron*, vol. 76, pp. 37 – 45, 2015.

[22] H. J. G. Gundersen and E. B. Jensen, "Stereological estimation of the volume-weighted mean volume of arbitrary particles observed on random sections*," *Journal of Microscopy*, vol. 138, no. 2, pp. 127–142, 1985.

[23] U. Semiautomatic, "Automatic image analysis," *ASTM International*, 2015.

[24] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, 05 2016.

[25] A. Chowdhury, E. Kautz, B. Yener, and D. Lewis, "Image driven machine learning methods for microstructure recognition," *Computational Materials Science*, vol. 123, pp. 176 – 187, 2016.

[26] D. Steinberger, H. Song, and S. Sandfeld, "Machine learning-based classification of dislocation microstructures," *Frontiers in Materials*, vol. 6, p. 141, 2019.

[27] B. DeCost, H. Jain, A. Rollett, and E. Holm, "Computer vision and machine learning for autonomous characterization of AM powder feedstocks," *JOM*, vol. 69, 12 2016.

[28] M. Furukawa, Z. Horita, M. Nemoto, R. Valiev, and T. Langdon, "Microhardness measurements and the hall-petch relationship in an Al-Mg alloy with submicrometer grain size," *Acta Materialia*, vol. 44, no. 11, pp. 4619 – 4629, 1996.

[29] S. Tachibana, S. Kawachi, K. Yamada, and T. Kunio, "Effect of grain refinement on the endurance limit of plain carbon steels at various strength levels.," *Transactions of the Japan Society of Mechanical Engineers. A*, vol. 54, pp. 1956–1961, 1988.

[30] W. Li, K. G. Field, and D. Morgan, "Automated defect analysis in electron microscopic images," *npj Computational Materials*, vol. 4, p. 36, Jul 2018.

[31] C. M. Anderson, J. Klein, H. Rajakumar, C. D. Judge, and L. K. Béland, "Automated detection of helium bubbles in irradiated X-750," *Ultramicroscopy*, vol. 217, p. 113068, 2020.

[32] M. Li, D. Chen, S. Liu, and D. Guo, "Online learning method based on support vector machine for metallographic image segmentation," *Signal, Image and Video Processing*, 09 2020.

[33] A. Baskaran, G. Kane, K. Biggs, R. Hull, and D. Lewis, "Adaptive characterization of microstructure dataset using a two stage machine learning approach," *Computational Materials Science*, vol. 177, p. 109593, 2020.

[34] J. Jang, D. Van, H. Jang, D. H. Baik, S. D. Yoo, J. Park, S. Mhin, J. Mazumder, and S. H. Lee, "Residual neural network-based fully convolutional network for microstructure segmentation," *Science and Technology of Welding and Joining*, vol. 25, no. 4, pp. 282–289, 2020.

[35] G. Roberts, S. Y. Haile, R. Sainju, D. J. Edwards, B. Hutchinson, and Y. Zhu, "Deep learning for semantic segmentation of defects in advanced STEM images of steels," *Scientific Reports*, vol. 9, p. 12744, Sep 2019.

[36] M. F. Kasim, D. Watson-Parris, L. Deaconu, S. Oliver, P. Hatfield, D. H. Froula, G. Gregori, M. Jarvis, S. Khatiwala, J. Korenaga, J. Topp-Mugglestone, E. Viezzer, and S. M. Vinko, "Building high accuracy emulators for scientific simulations with deep neural architecture search," 2020.

[37] P. Lehto, H. Remes, T. Saukkonen, H. Hänninen, and J. Romanoff, "Influence of grain size distribution on the Hall-Petch relationship of welded structural steel," *Materials Science and Engineering: A*, vol. 592, pp. 28 – 39, 2014.

[38] P. Lehto, H. Remes, T. Sarikka, and J. Romanoff, "Characterisation of local grain size variation of welded structural steel," *Welding in the World*, vol. 60, pp. 673 – 688, 2016.

[39] E. Shang and H. Zhang, "Image spam classification based on convolutional neural network," in *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 1, pp. 398–403, 2016.

[40] H. Al-Wzwazy, "Handwritten digit recognition using convolutional neural networks," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, 02 2016.

[41] S. Bhatnagar, D. Ghosal, and M. H. Kolekar, "Classification of fashion article images using convolutional neural networks," in *2017 Fourth International Conference on Image Information Processing (ICIIP)*, pp. 1–6, 2017.

[42] N. Sharma, V. Jain, and A. Mishra, "An analysis of convolutional neural networks for image classification," *Procedia Computer Science*, vol. 132, pp. 377 – 384, 2018. International Conference on Computational Intelligence and Data Science.

[43] G. Levi and T. Hassner, "Age and gender classification using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

[44] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, and A. N. Poo, "Multi-category classification by soft-max combination of binary classifiers," in *Multiple Classifier Systems* (T. Windeatt and F. Roli, eds.), (Berlin, Heidelberg), pp. 125–134, Springer Berlin Heidelberg, 2003.

[45] S. Chen and H. He, "Stock prediction using convolutional neural network," *IOP Conference Series: Materials Science and Engineering*, vol. 435, p. 012026, nov 2018.

[46] J. Xia, L. Cao, G. Zhang, and J. Liao, "Head pose estimation in the wild assisted by facial landmarks based on convolutional neural networks," *IEEE Access*, vol. 7, pp. 48470–48483, 2019.

[47] C. Yan, C. Lang, T. Wang, X. Du, and C. Zhang, "Age estimation based on convolutional neural network," in *Advances in Multimedia Information Processing – PCM 2014* (W. T. Ooi, C. G. M. Snoek, H. K. Tan, C.-K. Ho, B. Huet, and C.-W. Ngo, eds.), (Cham), pp. 211–220, Springer International Publishing, 2014.

[48] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[49] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[51] G. Philipp, D. Song, and J. G. Carbonell, "Gradients explode - Deep networks are shallow - ResNet explained," 2018.

[52] H. Xu, H. Xie, Y. Liu, C. Cheng, C. Niu, and Y. Zhang, "Deep cascaded attention network for multi-task brain tumor segmentation," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019* (D. Shen, T. Liu, T. M. Peters, L. H. Staib, C. Essert, S. Zhou, P.-T. Yap, and A. Khan, eds.), (Cham), pp. 420–428, Springer International Publishing, 2019.

[53] X. Xiao, S. Lian, Z. Luo, and S. Li, "Weighted Res-UNet for high-quality retina vessel segmentation," in *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, pp. 327–331, 2018.

[54] H. Zhang, X. Hong, S. Zhou, and Q. Wang, "Infrared image segmentation for photovoltaic panels based on Res-UNet," in *Pattern Recognition and Computer Vision* (Z. Lin, L. Wang, J. Yang, G. Shi, T. Tan, N. Zheng, X. Chen, and Y. Zhang, eds.), (Cham), pp. 611–622, Springer International Publishing, 2019.

[55] J. Feng, J. Deng, Z. Li, Z. Sun, H. Dou, and K. Jia, "End-to-end Res-Unet based reconstruction algorithm for photoacoustic imaging," *Biomed. Opt. Express*, vol. 11, pp. 5321–5340, Sep 2020.

[56] N. Hansen, "The CMA evolution strategy: A tutorial," *CoRR*, vol. abs/1604.00772, 2016.

[57] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[58] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[59] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016.

[60] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[61] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *ArXiv*, 2020.

[62] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, A. Hogan, lorenzomammana, yxNONG, AlexWang1900, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, F. Ingham, Frederik, Guilhen, Hatovix, J. Poznanski, J. Fang, L. Yu, changyu98, M. Wang, N. Gupta, O. Akhtar, PetrDvoracek, and P. Rai, "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," *Towards AI - Multidisciplinary Science Journal*, Oct. 2020.

[63] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," 2022.

[64] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.

[65] E. Bisong, *Google Colaboratory*, pp. 59–64. Berkeley, CA: Apress, 2019.

[66] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[67] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," *NIPS*, 2017.

[68] B. L. DeCost and E. A. Holm, "A large dataset of synthetic SEM images of powder materials and their ground truth 3D structures," *Data in Brief*, vol. 9, pp. 727 – 731, 2016.

[69] X. Li, H. J. Willy, S. Chang, W. Lu, T. S. Herng, and J. Ding, "Selective laser melting of stainless steel and alumina composite: Experimental and simulation studies on processing

parameters, microstructure and mechanical properties," *Materials and Design*, vol. 145, pp. 1 – 10, 2018.

[70] A. Kalra and R. L. Chhokar, "A hybrid approach using sobel and canny operator for digital image edge detection," in *2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE)*, pp. 305–310, 2016.

[71] S. Garg and M. Pant, "Meshfree methods: A comprehensive review of applications," *International Journal of Computational Methods*, vol. 15, no. 04, p. 1830001, 2018.

[72] K. Park and G. Paulino, "Cohesive Zone Models: A Critical Review of Traction-Separation Relationships Across Fracture Surfaces," *Applied Mechanics Reviews*, vol. 64, pp. 1002–, 11 2011.

[73] K.-H. Schwalbe, I. Scheider, and A. Cornec, *Guidelines for applying cohesive models to the damage behaviour of engineering materials and structures*. Springer Science & Business Media, 2012.

[74] H. Yuan and X. Li, "Critical remarks to cohesive zone modeling for three-dimensional elastoplastic fatigue crack propagation," *Engineering Fracture Mechanics*, vol. 202, pp. 311–331, 2018.

[75] B. A. Moore, E. Rougier, D. O'Malley, G. Srinivasan, A. Hunter, and H. Viswanathan, "Predictive modeling of dynamic fracture growth in brittle materials with machine learning," *Computational Materials Science*, vol. 148, pp. 46–53, 2018.

[76] C. Song and J. P. Wolf, "The scaled boundary finite-element method—alias consistent infinitesimal finite-element cell method—for elastodynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 147, no. 3, pp. 329–355, 1997.

[77] J. P. Wolf and C. Song, "The scaled boundary finite-element method - a primer: derivations," *Computers I& Structures*, vol. 78, no. 1, pp. 191–210, 2000.

[78] C. Song and J. P. Wolf, "The scaled boundary finite-element method - a primer: solution procedures," *Computers I& Structures*, vol. 78, no. 1, pp. 211–225, 2000.

[79] C. Song, E. T. Ooi, and S. Natarajan, "A review of the scaled boundary finite element method for two-dimensional linear elastic fracture mechanics," *Engineering Fracture Mechanics*, vol. 187, pp. 45–73, 2018.

[80] G. A. Francfort and J.-J. Marigo, "Revisiting brittle fracture as an energy minimization problem," *Journal of the Mechanics and Physics of Solids*, vol. 46, no. 8, pp. 1319–1342, 1998.

[81] M. Ambati, T. Gerasimov, and L. De Lorenzis, "A review on phase-field models of brittle fracture and a new fast hybrid formulation," *Computational Mechanics*, vol. 55, no. 2, pp. 383–405, 2015.

[82] M. Ambati, R. Kruse, and L. De Lorenzis, "A phase-field model for ductile fracture at finite strains and its experimental verification," *Computational Mechanics*, vol. 57, no. 1, pp. 149–167, 2016.

[83] N. Moës, J. Dolbow, and T. Belytschko, "A finite element method for crack growth without remeshing," *International Journal for Numerical Methods in Engineering*, vol. 46, no. 1, pp. 131–150, 1999.

[84] N. Sukumar, N. Moës, B. Moran, and T. Belytschko, "Extended finite element method for three-dimensional crack modelling," *International Journal for Numerical Methods in Engineering*, vol. 48, no. 11, pp. 1549–1570, 2000.

[85] H. Li, J. Li, and H. Yuan, "A review of the extended finite element method on macrocrack and microcrack growth simulations," *Theoretical and Applied Fracture Mechanics*, vol. 97, pp. 236–249, 2018.

[86] A. Egger, U. Pillai, K. Agathos, E. Kakouris, E. Chatzi, I. A. Aschroft, and S. P. Triantafyllou, "Discrete and phase field methods for linear elastic fracture mechanics: A comparative study and state-of-the-art review," *Applied Sciences*, vol. 9, no. 12, p. 2436, 2019.

[87] A. Sedmak, "Computational fracture mechanics: An overview from early efforts to recent achievements," *Fatigue & Fracture of Engineering Materials & Structures*, vol. 41, no. 12, pp. 2438–2474, 2018.

[88] A. Hunter, B. A. Moore, M. Mudunuru, V. Chau, R. Tchoua, C. Nyshadham, S. Karra, D. O'Malley, E. Rougier, H. Viswanathan, and G. Srinivasan, "Reduced-order modeling through machine learning and graph-theoretic approaches for brittle fracture applications," *Computational Materials Science*, vol. 157, pp. 87 – 98, 2019.

[89] D. J. Lucia, P. S. Beran, and W. A. Silva, "Reduced-order modeling: New approaches for computational physics," *Progress in aerospace sciences*, vol. 40, no. 1-2, pp. 51–117, 2004.

[90] J. Oliver, M. Caicedo, A. E. Huespe, J. Hernández, and E. Roubin, "Reduced order modeling strategies for computational multiscale fracture," *Computer Methods in Applied Mechanics and Engineering*, vol. 313, pp. 560–595, 2017.

[91] Z. Zhang and G. X. Gu, "Finite-Element-Based Deep-Learning Model for Deformation Behavior of Digital Materials," *Advanced Theory and Simulations*, vol. 3, no. 7, p. 2000031, 2020.

[92] C. Settgast, M. Abendroth, and M. Kuna, "Constitutive modeling of plastic deformation behavior of open-cell foam structures using neural networks," *Mechanics of Materials*, vol. 131, pp. 1–10, 2019.

[93] A. Danoun, E. Prulière, and Y. Chemisky, "Thermodynamically consistent Recurrent Neural Networks to predict non linear behaviors of dissipative materials subjected to non-proportional loading paths," *Mechanics of Materials*, vol. 173, p. 104436, 2022.

[94] J. Weng, R. Lindvall, K. Zhuang, J.-E. Ståhl, H. Ding, and J. Zhou, "A machine learning based approach for determining the stress-strain relation of grey cast iron from nanoindentation," *Mechanics of Materials*, vol. 148, p. 103522, 2020.

[95] F. Teng, G. Menary, S. Malinov, S. Yan, and J. B. Stevens, "Predicting the multiaxial stress-strain behavior of polyethylene terephthalate (PET) at different strain rates and temperatures above Tg by using an artificial neural network," *Mechanics of Materials*, vol. 165, p. 104175, 2022.

[96] D. Park, J. Jung, G. X. Gu, and S. Ryu, "A generalizable and interpretable deep learning model to improve the prediction accuracy of strain fields in grid composites," *Materials & Design*, vol. 223, p. 111192, 2022.

[97] C. Yang, Y. Kim, S. Ryu, and G. X. Gu, "Prediction of composite microstructure stress-strain curves using convolutional neural networks," *Materials & Design*, vol. 189, p. 108509, 2020.

[98] J. K. Wilt, C. Yang, and G. X. Gu, "Accelerating auxetic metamaterial design with deep learning," *Advanced Engineering Materials*, vol. 22, no. 5, p. 1901266, 2020.

[99] Z. Zhang, Z. Zhang, F. Di Caprio, and G. X. Gu, "Machine learning for accelerating the design process of double-double composite structures," *Composite Structures*, vol. 285, p. 115233, 2022.

[100] G. X. Gu, C.-T. Chen, D. J. Richmond, and M. J. Buehler, "Bioinspired hierarchical composite design using machine learning: simulation, additive manufacturing, and experiment," *Mater. Horiz.*, vol. 5, pp. 939–945, 2018.

[101] G. X. Gu, C.-T. Chen, and M. J. Buehler, "De novo composite design based on machine learning algorithm," *Extreme Mechanics Letters*, vol. 18, pp. 19–28, 2018.

[102] M. Bessa, R. Bostanabad, Z. Liu, A. Hu, D. W. Apley, C. Brinson, W. Chen, and W. Liu, "A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality," *Computer Methods in Applied Mechanics and Engineering*, vol. 320, pp. 633–667, 2017.

[103] N. N. Vlassis and W. Sun, "Component-Based Machine Learning Paradigm for Discovering Rate-Dependent and Pressure-Sensitive Level-Set Plasticity Models," *Journal of Applied Mechanics*, vol. 89, 11 2021. 021003.

[104] J. N. Fuhg, L. van Wees, M. Obstalecki, P. Shade, N. Bouklas, and M. Kasemer, "Machine-learning convex and texture-dependent macroscopic yield from crystal plasticity simulations," *Materialia*, vol. 23, p. 101446, 2022.

[105] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, and M. A. Bessa, "Deep learning predicts path-dependent plasticity," *Proceedings of the National Academy of Sciences*, vol. 116, no. 52, pp. 26414–26420, 2019.

[106] A. Chandrashekar, P. Belardinelli, M. A. Bessa, U. Staufer, and F. Alijani, "Quantifying nanoscale forces using machine learning in dynamic atomic force microscopy," *Nanoscale Adv.*, vol. 4, pp. 2134–2143, 2022.

[107] T. Kadeethum, D. O'Malley, J. N. Fuhg, Y. Choi, J. Lee, H. S. Viswanathan, and N. Bouklas, "A framework for data-driven solution and parameter estimation of PDEs using conditional generative adversarial networks," *Nature Computational Science*, vol. 1, 12 2021.

[108] T. Kadeethum, D. O'Malley, Y. Choi, H. Viswanathan, N. Bouklas, and H. Yoon, "Continuous conditional generative adversarial networks for data-driven solutions of poroelasticity with heterogeneous material properties," *Computers & Geosciences*, vol. 167, p. 105212, 2022.

[109] D. Lizama, T. Kadeethum, D. O'Malley, Y. Choi, H. Viswanathan, H. Yoon, and N. Bouklas, "A framework for data-driven solution of linear poroelasticity using continuous conditional generative adversarial networks," in *AGU Fall Meeting Abstracts*, vol. 2021, pp. H12E–03, Dec. 2021.

[110] X. He, Q. He, and J.-S. Chen, "Deep autoencoders for physics-constrained data-driven nonlinear materials modeling," *Computer Methods in Applied Mechanics and Engineering*, vol. 385, p. 114034, 2021.

[111] Z. Yang, C.-H. Yu, and M. J. Buehler, "Deep learning model to predict complex stress and strain fields in hierarchical composites," *Science Advances*, vol. 7, no. 15, 2021.

[112] A. Mangal and E. A. Holm, "Applied machine learning to predict stress hotspots I: Face centered cubic materials," *International Journal of Plasticity*, vol. 111, pp. 122–134, 2018.

[113] A. Mangal and E. A. Holm, "Applied machine learning to predict stress hotspots II: Hexagonal close packed materials," *International Journal of Plasticity*, vol. 114, pp. 1–14, 2019.

[114] Z. Gao, W. Guo, and X. Yue, "Optimal integration of supervised tensor decomposition and ensemble learning for in situ quality evaluation in friction stir blind riveting," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 1, pp. 19–35, 2021.

[115] Y. Wang, W. G. Guo, and X. Yue, "Tensor decomposition to compress convolutional layers in deep learning," *IISE Transactions*, vol. 0, no. 0, pp. 1–60, 2021.

[116] A. Pandolfi, K. Weinberg, and M. Ortiz, "A comparative accuracy and convergence study of eigenerosion and phase-field models of fracture," *Computer Methods in Applied Mechanics and Engineering*, vol. 386, p. 114078, 2021.

[117] X. Liu, C. E. Athanasiou, N. P. Padture, B. W. Sheldon, and H. Gao, "A machine learning approach to fracture mechanics problems," *Acta Materialia*, vol. 190, pp. 105–112, 2020.

[118] E. Haghighat, A. C. Bekar, E. Madenci, and R. Juanes, "A nonlocal physics-informed deep learning framework using the peridynamic differential operator," *Computer Methods in Applied Mechanics and Engineering*, vol. 385, p. 114012, 2021.

[119] Y. Zhang, Z. Wen, H. Pei, J. Wang, Z. Li, and Z. Yue, "Equivalent method of evaluating mechanical properties of perforated Ni-based single crystal plates using artificial

neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112725, 2020.

[120] S. Saha, Z. Gan, L. Cheng, J. Gao, O. L. Kafka, X. Xie, H. Li, M. Tajdari, H. A. Kim, and W. K. Liu, "Hierarchical Deep Learning Neural Network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering," *Computer Methods in Applied Mechanics and Engineering*, vol. 373, p. 113452, 2021.

[121] Y. Wang, D. Oyen, W. Guo, A. Mehta, C. B. Scott, N. Panda, M. G. Fernández-Godino, G. Srinivasan, and X. Yue, "StressNet - Deep learning to predict stress with fracture propagation in brittle materials," *npj Materials Degradation*, vol. 5, 2 2021.

[122] D. Sharma, V. Pandey, I. Singh, S. Natarajan, J. Kumar, and S. Ahmad, "A polygonal FEM and continuum damage mechanics based framework for stochastic simulation of fatigue life scatter in duplex microstructure titanium alloys," *Mechanics of Materials*, vol. 163, p. 104071, 2021.

[123] L. Zhang and X. Wei, "Prediction of fatigue crack growth under variable amplitude loading by artificial neural network-based Lagrange interpolation," *Mechanics of Materials*, vol. 171, p. 104309, 2022.

[124] A. Ouladbrahim, I. Belaidi, S. Khatir, E. Magagnini, R. Capozucca, and M. Abdel Wahab, "Experimental crack identification of API X70 steel pipeline using improved Artificial Neural Networks based on Whale Optimization Algorithm," *Mechanics of Materials*, vol. 166, p. 104200, 2022.

[125] C. Cao, G. Li, X. Jin, H. Ding, and J. Zhao, "Continuous fracture of soft tissue under high-speed waterjet impact and its quantification method," *Mechanics of Materials*, vol. 151, p. 103631, 2020.

[126] V. Daghigh, J. Thomas E Lacy, H. Daghigh, G. Gu, K. T. Baghaei, M. F. Horstemeyer, and J. Charles U Pittman, "Machine learning predictions on fracture toughness of multiscale

bio-nano-composites," *Journal of Reinforced Plastics and Composites*, vol. 39, no. 15-16, pp. 587–598, 2020.

[127] C. Gebhardt, T. Trimborn, F. Weber, A. Bezold, C. Broeckmann, and M. Herty, "Simplified ResNet approach for data driven prediction of microstructure-fatigue relationship," *Mechanics of Materials*, vol. 151, p. 103625, 2020.

[128] A. J. Lew, C.-H. Yu, Y.-C. Hsu, and M. J. Buehler, "Deep learning model to predict fracture mechanisms of graphene," *npj 2D Materials and Applications*, vol. 5, no. 1, pp. 1–8, 2021.

[129] S. Im, J. Lee, and M. Cho, "Surrogate modeling of elasto-plastic problems via long short-term memory neural networks and proper orthogonal decomposition," *Computer Methods in Applied Mechanics and Engineering*, vol. 385, p. 114030, 2021.

[130] S. Feng, Y. Xu, X. Han, Z. Li, and A. Incecik, "A phase field and deep-learning based approach for accurate prediction of structural residual useful life," *Computer Methods in Applied Mechanics and Engineering*, vol. 383, p. 113885, 2021.

[131] E. Knight, E. Rougier, Z. Lei, B. Euser, V. Chau, S. Boyce, K. Okubo, and M. Froment, "HOSS: an implementation of the combined finite-discrete element method," *Computational Particle Mechanics*, vol. 7, 07 2020.

[132] B. Euser, E. Rougier, Z. Lei, E. E. Knight, L. P. Frash, J. W. Carey, H. Viswanathan, and A. Munjiza, "Simulation of fracture coalescence in granite via the combined finite-discrete element method," *Rock Mechanics and Rock Engineering*, vol. 52, pp. 3213–3227, Mar 2019.

[133] Y.-C. Hsu, C.-H. Yu, and M. J. Buehler, "Using Deep Learning to Predict Fracture Patterns in Crystalline Solids," *Matter*, vol. 3, no. 1, pp. 197–211, 2020.

[134] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[135] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pp. 177–186, Springer, 2010.

[136] N. Ketkar and N. Ketkar, "Stochastic gradient descent," *Deep learning with Python: A hands-on introduction*, pp. 113–132, 2017.

[137] J. Gasteiger, J. Groß, and S. Günnemann, "Directional message passing for molecular graphs," *arXiv preprint arXiv:2003.03123*, 2020.

[138] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *International Conference on Machine Learning*, pp. 8459–8468, PMLR, 2020.

[139] Q. Wang and L. Zhang, "Inverse design of glass structure with deep graph neural networks," *Nature communications*, vol. 12, no. 1, pp. 1–11, 2021.

[140] G. Gu, J. Jang, J. Noh, A. Walsh, and Y. Jung, "Perovskite synthesizability using graph neural networks," *npj Computational Materials*, vol. 8, p. 71, 04 2022.

[141] A. L. Frankel, C. Safta, C. Alleman, and R. Jones, "Mesh-based graph convolutional neural networks for modeling materials with microstructure," *Journal of Machine Learning for Modeling and Computing*, vol. 3, no. 1, 2022.

[142] M. Dai, M. F. Demirel, Y. Liang, and J.-M. Hu, "Graph neural networks for an accurate and interpretable prediction of the properties of polycrystalline materials," *npj Computational Materials*, vol. 7, no. 1, pp. 1–9, 2021.

[143] C. Shu, J. He, G. Xue, and C. Xie, "Grain knowledge graph representation learning: A new paradigm for microstructure-property prediction," *Crystals*, vol. 12, no. 2, 2022.

[144] K. Choudhary and B. DeCost, "Atomistic line graph neural network for improved materials property predictions," *npj Computational Materials*, vol. 7, no. 1, pp. 1–8, 2021.

[145] S. Tsopanidis and S. Osovski, "A graph-based workflow for extracting grain-scale toughness from meso-scale experiments," *Materials & Design*, vol. 213, p. 110272, 2022.

[146] V. Fung, J. Zhang, E. Juarez, and B. G. Sumpter, "Benchmarking graph neural networks for materials chemistry," *npj Computational Materials*, vol. 7, no. 1, pp. 1–8, 2021.

[147] A. S. Rosen, V. Fung, P. Huck, C. T. O'Donnell, M. K. Horton, D. G. Truhlar, K. A. Persson, J. M. Notestein, and R. Q. Snurr, "High-throughput predictions of metal–organic framework electronic properties: theoretical challenges, graph neural networks, and data exploration," *npj Computational Materials*, vol. 8, no. 1, pp. 1–10, 2022.

[148] N. N. Vlassis, R. Ma, and W. Sun, "Geometric deep learning for computational mechanics Part I: Anisotropic hyperelasticity," *Computer Methods in Applied Mechanics and Engineering*, vol. 371, p. 113299, 2020.

[149] Y. Heider, K. Wang, and W. Sun, "SO(3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials," *Computer Methods in Applied Mechanics and Engineering*, vol. 363, p. 112875, 2020.

[150] X. Lin, H. Jiang, L. Wang, Y. Ren, W. Ma, and S. Zhan, "3D-structure-attention graph neural network for crystals and materials," *Molecular Physics*, vol. 120, no. 11, p. e2077258, 2022.

[151] K. Choudhary, T. Yildirim, D. W. Siderius, A. G. Kusne, A. McDannald, and D. L. Ortiz-Montalvo, "Graph neural network predictions of metal organic framework CO2 adsorption properties," *Computational Materials Science*, vol. 210, p. 111388, 2022.

[152] S. Tsopanidis and S. Osovski, "A graph-based workflow for extracting grain-scale toughness from meso-scale experiments," *Materials & Design*, vol. 213, p. 110272, 2022.

[153] O. Sidorov and J. Y. Hardeberg, "Craquelure as a graph: Application of image processing and graph neural networks to the description of fracture patterns," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 1429–1436, 2019.

[154] B. Zhang, M. Zhou, J. Wu, and F. Gao, "Predicting the materials properties using a 3D graph neural network with invariant representation," *IEEE Access*, vol. 10, pp. 62440–62449, 2022.

[155] C. W. Park, M. Kornbluth, J. Vandermause, C. Wolverton, B. Kozinsky, and J. P. Mailoa, "Accurate and scalable graph neural network force field and molecular dynamics with direct force architecture," *npj Computational Materials*, vol. 7, no. 1, pp. 1–9, 2021.

[156] N. Black and A. R. Najafi, "Learning finite element convergence with the multi-fidelity graph neural network," *Computer Methods in Applied Mechanics and Engineering*, vol. 397, p. 115120, 2022.

[157] N. N. Vlassis and W. Sun, "Geometric deep learning for computational mechanics Part II: Graph embedding for interpretable multiscale plasticity," *arXiv preprint arXiv:2208.00246*, 2022.

[158] Z. Li and A. B. Farimani, "Graph neural network-accelerated Lagrangian fluid simulation," *Computers & Graphics*, vol. 103, pp. 201–211, 2022.

[159] R. Bhattoo, S. Ranu, and N. Krishnan, "Learning articulated rigid body dynamics with lagrangian graph neural network," *Advances in Neural Information Processing Systems*, vol. 35, pp. 29789–29800, 2022.

[160] A. Mayr, S. Lehner, A. Mayrhofer, C. Kloss, S. Hochreiter, and J. Brandstetter, "Boundary graph neural networks for 3D simulations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 9099–9107, Jun. 2023.

[161] Z. Zhuang, Z. Liu, B. Cheng, and J. Liao, "Chapter 2 - Fundamental Linear Elastic Fracture Mechanics," in *Extended Finite Element Method* (Z. Zhuang, Z. Liu, B. Cheng, and J. Liao, eds.), pp. 13–31, Oxford: Academic Press, 2014.

[162] D. Sutula, P. Kerfriden, T. van Dam, and S. P. Bordas, "Minimum energy multiple crack propagation. Part I: Theory and state of the art review," *Engineering Fracture Mechanics*, vol. 191, pp. 205–224, 2018.

[163] D. Sutula, P. Kerfriden, T. van Dam, and S. P. Bordas, "Minimum energy multiple crack propagation. Part-II: Discrete solution with XFEM," *Engineering Fracture Mechanics*, vol. 191, pp. 225–256, 2018.

[164] D. Sutula, P. Kerfriden, T. van Dam, and S. P. Bordas, "Minimum energy multiple crack propagation. Part III: XFEM computer implementation and applications," *Engineering Fracture Mechanics*, vol. 191, pp. 257–276, 2018.

[165] V. F. González-Albuixech, E. Giner, J. E. Tarancón, F. J. Fuenmayor, and A. Gravouil, "Domain integral formulation for 3-D curved and non-planar cracks with the extended finite element method," *Computer Methods in Applied Mechanics and Engineering*, vol. 264, pp. 129–144, 2013.

[166] X.-K. Zhu, "Improved Incremental J-Integral Equations for Determining Crack Growth Resistance Curves," *Journal of Pressure Vessel Technology*, vol. 134, 09 2012. 051404.

[167] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020.

[168] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[169] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[170] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *CoRR*, vol. abs/1806.01261, 2018.

[171] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in *NIPS*, pp. 4502–4510, 2016.

[172] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks," *CoRR*, vol. abs/2002.09405, 2020.

[173] Y. Li, T. Lin, K. Yi, D. Bear, D. Yamins, J. Wu, J. Tenenbaum, and A. Torralba, "Visual Grounding of Learned Physical Models," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 5927–5936, PMLR, 13–18 Jul 2020.

[174] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning Mesh-Based Simulation with Graph Networks," *CoRR*, vol. abs/2010.03409, 2020.

[175] A. F. Bower, *Applied Mechanics of Solids*, ch. Chapter 9: Modeling Material Failure. CRC Press, 1991.

[176] J. Klicpera, J. Groß, and S. Günnemann, "Directional message passing for molecular graphs," *CoRR*, vol. abs/2003.03123, 2020.

[177] R. Venkatesan and M. J. Er, "A novel progressive learning technique for multi-class classification," *Neurocomputing*, vol. 207, pp. 310–321, 2016.

[178] P. Refaeilzadeh, L. Tang, H. Liu, L. Liu, and M. Özsu, "Encyclopedia of database systems," *Cross-validation*, vol. 5, pp. 532–538, 2009.

[179] W. Zhang, Y. Shen, Z. Lin, Y. Li, X. Li, W. Ouyang, Y. Tao, Z. Yang, and B. Cui, "GMLP: Building scalable and flexible graph neural networks with feature-message passing," 2021.

[180] Z. Li and A. B. Farimani, "Learning lagrangian fluid dynamics with graph neural networks," 2021.

[181] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," *CoRR*, vol. abs/1903.02428, 2019.

[182] M. Tiezzi, G. Marra, S. Melacci, M. Maggini, and M. Gori, "A Lagrangian Approach to Information Propagation in Graph Neural Networks," *arXiv preprint arXiv:2002.07684*, 2020.

[183] M. Willjuice Iruthayarajan and S. Baskar, "Covariance matrix adaptation evolution strategy based design of centralized PID controller," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5775–5781, 2010.

[184] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, "Advances in neural information processing systems," *Curr Assoc*, vol. 27, pp. 3320–3328, 2014.

[185] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264, IGI global, 2010.

[186] X. Liu, C. E. Athanasiou, N. P. Padture, B. W. Sheldon, and H. Gao, "Knowledge extraction and transfer in data-driven fracture mechanics," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 118, p. e2104765118, June 2021.

[187] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[188] M. Imad, O. Doukhi, and D.-J. Lee, "Transfer learning based semantic segmentation for 3D object detection from point cloud," *Sensors*, vol. 21, no. 12, 2021.

[189] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019–1034, 2015.

[190] J. Talukdar, S. Gupta, P. S. Rajpura, and R. S. Hegde, "Transfer learning for object detection using state-of-the-art deep neural networks," in *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 78–83, 2018.

[191] B. Bamne, N. Shrivastava, L. Parashar, and U. Singh, "Transfer learning-based object detection by using convolutional neural networks," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 328–332, 2020.

[192] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, pp. 270–279, Springer, 2018.

[193] N. Saeed, N. King, Z. Said, and M. A. Omar, "Automatic defects detection in CFRP thermograms, using convolutional neural networks and transfer learning," *Infrared Physics & Technology*, vol. 102, p. 103048, 2019.

[194] Y. Gong, J. Luo, H. Shao, and Z. Li, "A transfer learning object detection model for defects detection in X-ray images of spacecraft composite structures," *Composite Structures*, vol. 284, p. 115136, 2022.

[195] B. Zheng, Z. Zheng, and G. X. Gu, "Scalable graphene defect prediction using transferable learning," *Nanomaterials*, vol. 11, no. 9, p. 2341, 2021.

[196] N. N. Vlassis, P. Zhao, R. Ma, T. Sewell, and W. Sun, "Molecular dynamics inferred transfer learning models for finite-strain hyperelasticity of monoclinic crystals: Sobolev training and validations against physical constraints," *International Journal for Numerical Methods in Engineering*, vol. 123, no. 17, pp. 3922–3949, 2022.

[197] X. Li, Y. Zhang, H. Zhao, C. Burkhart, L. C. Brinson, and W. Chen, "A transfer learning approach for microstructure reconstruction and structure-property predictions," *Scientific reports*, vol. 8, no. 1, pp. 1–13, 2018.

[198] B. L. DeCost, T. Francis, and E. A. Holm, "Exploring the microstructure manifold: Image texture representations applied to ultrahigh carbon steel microstructures," *Acta Materialia*, vol. 133, pp. 30–40, 2017.

[199] Y. Kim, Y. Kim, C. Yang, K. Park, G. X. Gu, and S. Ryu, "Deep learning framework for material design space exploration using active transfer learning and data augmentation," *npj Computational Materials*, vol. 7, no. 1, pp. 1–7, 2021.

[200] A. Kolluru, N. Shoghi, M. Shuaibi, S. Goyal, A. Das, C. L. Zitnick, and Z. Ulissi, "Transfer learning using attentions across atomic systems with graph neural networks (TAAG)," *The Journal of Chemical Physics*, vol. 156, no. 18, p. 184702, 2022.

[201] C. Chen and S. P. Ong, "AtomSets as a hierarchical transfer learning framework for small and large materials datasets," *npj Computational Materials*, vol. 7, no. 1, pp. 1–9, 2021.

[202] J. Lee and R. Asahi, "Transfer learning for materials informatics using crystal graph convolutional neural network," *Computational Materials Science*, vol. 190, p. 110314, 2021.

[203] T. Xie and J. C. Grossman, "Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties," *Physical review letters*, vol. 120, no. 14, p. 145301, 2018.

[204] R. Wang, Y. Zou, C. Zhang, X. Wang, M. Yang, and D. Xu, "Combining crystal graphs and domain knowledge in machine learning to predict metal-organic frameworks performance in methane adsorption," *Microporous and Mesoporous Materials*, vol. 331, p. 111666, 2022.

[205] R. Perera, D. Guzzetti, and V. Agrawal, "Graph neural networks for simulating crack coalescence and propagation in brittle materials," *Computer Methods in Applied Mechanics and Engineering*, vol. 395, p. 115021, 2022.

[206] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," 2019.

[207] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105524, 2020.

[208] P. Ferreira, D. C. Le, and N. Zincir-Heywood, "Exploring feature normalization and temporal information for machine learning based insider threat detection," in *2019 15th International Conference on Network and Service Management (CNSM)*, pp. 1–7, 2019.

[209] A. Sedmak, "Computational fracture mechanics: An overview from early efforts to recent achievements," *Fatigue & Fracture of Engineering Materials & Structures*, vol. 41, no. 12, pp. 2438–2474, 2018.

[210] F. Ernesti, M. Schneider, and T. Böhlke, "Fast implicit solvers for phase-field fracture problems on heterogeneous microstructures," *Computer Methods in Applied Mechanics and Engineering*, vol. 363, p. 112793, 2020.

[211] S. Goswami, C. Anitescu, and T. Rabczuk, "Adaptive fourth-order phase field analysis for brittle fracture," *Computer Methods in Applied Mechanics and Engineering*, vol. 361, p. 112808, 2020.

[212] G. Zhang, T. F. Guo, K. I. Elkhodary, S. Tang, and X. Guo, "Mixed Graph-FEM phase field modeling of fracture in plates and shells with nonlinearly elastic solids," *Computer Methods in Applied Mechanics and Engineering*, vol. 389, p. 114282, 2022.

[213] G. Francfort and J.-J. Marigo, "Revisiting brittle fracture as an energy minimization problem," *Journal of the Mechanics and Physics of Solids*, vol. 46, no. 8, pp. 1319–1342, 1998.

[214] A. Egger, U. Pillai, K. Agathos, E. Kakouris, E. Chatzi, I. A. Aschroft, and S. P. Triantafyllou, "Discrete and phase field methods for linear elastic fracture mechanics: A comparative study and state-of-the-art review," *Applied Sciences*, vol. 9, no. 12, 2019.

[215] B. Runnels and V. Agrawal, "Phase field disconnections: A continuum method for disconnection-mediated grain boundary motion," *Scripta Materialia*, vol. 186, pp. 6–10, 2020.

[216] W. Xu, H. Yu, J. Zhang, C. Lyu, Q. Wang, M. Micheal, and H. Wu, "Phase-field method of crack branching during SC-CO2 fracturing: A new energy release rate criterion coupling pore pressure gradient," *Computer Methods in Applied Mechanics and Engineering*, vol. 399, p. 115366, 2022.

[217] S. A. Vajari, M. Neuner, P. K. Arunachala, A. Ziccarelli, G. Deierlein, and C. Linder, "A thermodynamically consistent finite strain phase field approach to ductile fracture considering multi-axial stress states," *Computer Methods in Applied Mechanics and Engineering*, vol. 400, p. 115467, 2022.

[218] W. Li, M. Ambati, N. Nguyen-Thanh, H. Du, and K. Zhou, "Adaptive fourth-order phase-field modeling of ductile fracture using an isogeometric-meshfree approach," *Computer Methods in Applied Mechanics and Engineering*, vol. 406, p. 115861, 2023.

[219] J. Han, S. Matsubara, S. Moriguchi, and K. Terada, "Variational crack phase-field model for ductile fracture with elastic and plastic damage variables," *Computer Methods in Applied Mechanics and Engineering*, vol. 400, p. 115577, 2022.

[220] V. Agrawal and B. Runnels, "Robust, strong form mechanics on an adaptive structured grid: Efficiently solving variable-geometry near-singular problems with diffuse interfaces," *Computational Mechanics*, pp. 1–19, 2023.

[221] V. Agrawal and B. Runnels, "Block structured adaptive mesh refinement and strong form elasticity approach to phase field fracture with applications to delamination, crack branching and crack deflection," 2021.

[222] Y. Chen and Y. Shen, "A "parallel universe" scheme for crack nucleation in the phase field approach to fracture," *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115708, 2023.

[223] S. Brach, E. Tanné, B. Bourdin, and K. Bhattacharya, "Phase-field study of crack nucleation and propagation in elastic–perfectly plastic bodies," *Computer Methods in Applied Mechanics and Engineering*, vol. 353, pp. 44–65, 2019.

[224] O. Gültekin, H. Dal, and G. A. Holzapfel, "Numerical aspects of anisotropic failure in soft biological tissues favor energy-based criteria: A rate-dependent anisotropic crack phase-field model," *Computer methods in applied mechanics and engineering*, vol. 331, pp. 23–52, 2018.

[225] M. Marulli, A. Valverde-González, A. Quintanas-Corominas, M. Paggi, and J. Reinoso, "A combined phase-field and cohesive zone model approach for crack propagation in layered structures made of nonlinear rubber-like materials," *Computer Methods in Applied Mechanics and Engineering*, vol. 395, p. 115007, 2022.

[226] B. Yin and M. Kaliske, "An anisotropic phase-field model based on the equivalent crack surface energy density at finite strain," *Computer Methods in Applied Mechanics and Engineering*, vol. 369, p. 113202, 2020.

[227] B. Runnels, V. Agrawal, W. Zhang, and A. Almgren, "Massively parallel finite difference elasticity using block-structured adaptive mesh refinement with a geometric multigrid solver," *Journal of Computational Physics*, vol. 427, p. 110065, 2021.

[228] V. Agrawal and B. Runnels, "Block structured adaptive mesh refinement and strong form elasticity approach to phase field fracture with applications to delamination, crack branching and crack deflection," *Computer Methods in Applied Mechanics and Engineering*, vol. 385, p. 114011, 2021.

[229] J. G. Ribot, V. Agrawal, and B. Runnels, "A new approach for phase field modeling of grain boundaries with strongly nonconvex energy," *Modelling and Simulation in Materials Science and Engineering*, vol. 27, p. 084007, oct 2019.

[230] C. D. Norton, T. A. Cwik, and J. Z. Lou, "Status and directions for the PYRAMID Parallel Unstructured AMR library," in *Parallel and Distributed Processing Symposium, International*, vol. 4, (Los Alamitos, CA, USA), p. 30120b, IEEE Computer Society, apr 2001.

[231] P. MacNeice, K. M. Olson, C. Mobarry, R. de Fainchtein, and C. Packer, "PARAMESH: A parallel adaptive mesh refinement community toolkit," *Computer Physics Communications*, vol. 126, no. 3, pp. 330–354, 2000.

[232] Y. Feng, Q. Wang, D. Wu, W. Gao, and F. Tin-Loi, "Stochastic nonlocal damage analysis by a machine learning approach," *Computer Methods in Applied Mechanics and Engineering*, vol. 372, p. 113371, 2020.

[233] G. Capuano and J. J. Rimoli, "Smart finite elements: A novel machine learning application," *Computer Methods in Applied Mechanics and Engineering*, vol. 345, pp. 363–381, 2019.

[234] S. Lee, Z. Zhang, and G. X. Gu, "Generative machine learning algorithm for lattice structures with superior mechanical properties," *Mater. Horiz.*, vol. 9, pp. 952–960, 2022.

[235] J. M. Hanna, J. V. Aguado, S. Comas-Cardona, R. Askri, and D. Borzacchiello, "Residual-based adaptivity for two-phase flow simulation in porous media using physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 396, p. 115100, 2022.

[236] P. Ren, C. Rao, Y. Liu, J.-X. Wang, and H. Sun, "PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs," *Computer Methods in Applied Mechanics and Engineering*, vol. 389, p. 114399, 2022.

[237] M. Wang, S. Feng, A. Incecik, G. Królczyk, and Z. Li, "Structural fatigue life prediction considering model uncertainties through a novel digital twin-driven approach," *Computer Methods in Applied Mechanics and Engineering*, vol. 391, p. 114512, 2022.

[238] A. Mangal and E. A. Holm, "Applied machine learning to predict stress hotspots I: Face centered cubic materials," *International Journal of Plasticity*, vol. 111, pp. 122–134, 2018.

[239] A. Mangal and E. A. Holm, "Applied machine learning to predict stress hotspots II: Hexagonal close packed materials," *International Journal of Plasticity*, vol. 114, pp. 1–14, 2019.

[240] X. He, Q. He, and J.-S. Chen, "Deep autoencoders for physics-constrained data-driven nonlinear materials modeling," *Computer Methods in Applied Mechanics and Engineering*, vol. 385, p. 114034, 2021.

[241] Y. Wang, D. Oyen, W. G. Guo, A. Mehta, C. B. Scott, N. Panda, M. G. Fernández-Godino, G. Srinivasan, and X. Yue, "StressNet-Deep learning to predict stress with fracture propagation in brittle materials," *Npj Materials Degradation*, vol. 5, no. 1, pp. 1–10, 2021.

[242] Z. Yang, C.-H. Yu, and M. J. Buehler, "Deep learning model to predict complex stress and strain fields in hierarchical composites," *Science Advances*, vol. 7, no. 15, p. eabd7416, 2021.

[243] S. Saha, Z. Gan, L. Cheng, J. Gao, O. L. Kafka, X. Xie, H. Li, M. Tajdari, H. A. Kim, and W. K. Liu, "Hierarchical deep learning neural network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering," *Computer Methods in Applied Mechanics and Engineering*, vol. 373, p. 113452, 2021.

[244] A. Rovinelli, M. D. Sangid, H. Proudhon, and W. Ludwig, "Using machine learning and a data-driven approach to identify the small fatigue crack driving force in polycrystalline materials," *npj Computational Materials*, vol. 4, no. 1, pp. 1–10, 2018.

[245] M. S. Elapolu, M. I. R. Shishir, and A. Tabarraei, "A novel approach for studying crack propagation in polycrystalline graphene using machine learning algorithms," *Computational Materials Science*, vol. 201, p. 110878, 2022.

[246] K. Zhang, J. Wang, Y. Huang, L.-Q. Chen, and Y. Cao, "High-throughput phase-field simulations and machine learning of resistive switching in resistive random-access memory," *npj Computational Materials*, vol. 6, 12 2020.

[247] Y. Zhu, T. Xu, Q. Wei, J. Mai, H. Yang, H. Zhang, T. Shimada, T. Kitamura, and T.-Y. Zhang, "Linear-superelastic Ti-Nb nanocomposite alloys with ultralow modulus via high-throughput phase-field design and machine learning," *npj Computational Materials*, vol. 7, 12 2021.

[248] E. Samaniego, C. Anitescu, S. Goswami, V. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk, "An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 362, p. 112790, 2020.

[249] G. H. Teichert and K. Garikipati, "Machine learning materials physics: Surrogate optimization and multi-fidelity algorithms predict precipitate morphology in an alternative to phase field dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 344, pp. 666–693, 2019.

[250] V. Oommen, K. Shukla, S. Goswami, R. Dingreville, and G. E. Karniadakis, "Learning two-phase microstructure evolution using neural operators and autoencoder architectures," *arXiv preprint arXiv:2204.07230*, 2022.

[251] D. Montes de Oca Zapiain, J. A. Stewart, and R. Dingreville, "Accelerating phase-field-based microstructure evolution predictions via surrogate models trained by machine learning methods," *npj Computational Materials*, vol. 7, no. 1, pp. 1–11, 2021.

[252] Z. Jin, B. Zheng, C. Kim, and G. X. Gu, "Leveraging graph neural networks and neural operator techniques for high-fidelity mesh-based physics simulations," *APL Machine Learning*, vol. 1, p. 046109, 11 2023.

[253] C. Jiang and N.-Z. Chen, "Graph neural networks (GNNs) based accelerated numerical simulation," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106370, 2023.

[254] J. C. Wong, C. C. Ooi, J. Chattoraj, L. Lestandi, G. Dong, U. Kizhakkinan, D. W. Rosen, M. H. Jhon, and M. H. Dao, "Graph neural network based surrogate model of physics simulations for geometry design," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1469–1475, 2022.

[255] X. Shao, Z. Liu, S. Zhang, Z. Zhao, and C. Hu, "PIGNN-CFD: A physics-informed graph neural network for rapid predicting urban wind field defined on unstructured mesh," *Building and Environment*, vol. 232, p. 110056, 2023.

[256] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning mesh-based simulation with graph networks," *arXiv preprint arXiv:2010.03409*, 2020.

[257] L. Zhang, D. Xu, A. Arnab, and P. H. Torr, "Dynamic Graph Message Passing Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[258] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, PMLR, 06–11 Aug 2017.

[259] J. Zhu, Y. Song, L. Zhao, and H. Li, "A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting," 2020.

[260] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2020.

[261] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," 2017.

[262] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[263] T. Fushiki, "Estimation of prediction error by using k-fold cross-validation," *Statistics and Computing*, vol. 21, pp. 137–146, apr 2011.

[264] Q. Li, Z. Han, and X.-m. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, Apr. 2018.

[265] T. K. Rusch, M. M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," *arXiv preprint arXiv:2303.10993*, 2023.

[266] C. Cai and Y. Wang, "A note on over-smoothing for graph neural networks," *arXiv preprint arXiv:2006.13318*, 2020.

[267] M. Fortunato, T. Pfaff, P. Wirnsberger, A. Pritzel, and P. Battaglia, "Multiscale MeshGraphNets," *arXiv preprint arXiv:2210.00612*, 2022.

[268] K. Stüben *et al.*, "An introduction to algebraic multigrid," *Multigrid*, pp. 413–532, 2001.

[269] J. Xu and L. Zikatanov, "Algebraic multigrid methods," *Acta Numerica*, vol. 26, pp. 591–721, 2017.

[270] M. Eliasof and E. Treister, "DiffGCN: Graph convolutional networks via differential operators and algebraic multigrid pooling," *Advances in neural information processing systems*, vol. 33, pp. 18016–18027, 2020.

[271] Z. Yang, Y. Dong, X. Deng, and L. Zhang, "AMGNET: Multi-scale graph neural networks for flow field prediction," *Connection Science*, vol. 34, no. 1, pp. 2500–2519, 2022.

[272] M. Lino, C. Cantwell, A. A. Bharath, and S. Fotiadis, "Simulating continuum mechanics with multi-scale graph neural networks," *arXiv preprint arXiv:2106.04900*, 2021.

[273] M. Lino, S. Fotiadis, A. A. Bharath, and C. Cantwell, "Towards fast simulation of environmental fluid mechanics with multi-scale graph neural networks," *arXiv preprint arXiv:2205.02637*, 2022.

[274] R. J. Gladstone, H. Rahmani, V. Suryakumar, H. Meidani, M. D'Elia, and A. Zareei, "GNN-based physics solver for time-independent PDEs," *arXiv preprint arXiv:2303.15681*, 2023.

[275] W. Liu, M. Yagoubi, and M. Schoenauer, "Multi-resolution graph neural networks for PDE approximation," in *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part III 30*, pp. 151–163, Springer, 2021.

[276] I. Luz, M. Galun, H. Maron, R. Basri, and I. Yavneh, "Learning algebraic multigrid using graph neural networks," in *International Conference on Machine Learning*, pp. 6489–6499, PMLR, 2020.

[277] Y. Cao, M. Chai, M. Li, and C. Jiang, "Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network," in *International Conference on Machine Learning*, pp. 3541–3558, PMLR, 2023.

[278] H. Gao and S. Ji, "Graph U-Nets," in *International Conference on Machine Learning*, pp. 2083–2092, PMLR, 2019.

[279] S. Barwey, V. Shankar, V. Viswanathan, and R. Maulik, "Multiscale graph neural network autoencoders for interpretable scientific machine learning," *Journal of Computational Physics*, vol. 495, p. 112537, 2023.

[280] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in neural information processing systems*, vol. 27, 2014.

[281] G. Francfort, "Variational fracture: twenty years after," *International Journal of Fracture*, vol. 237, no. 1-2, pp. 3–13, 2022.

[282] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," *arXiv preprint arXiv:2009.03509*, 2020.

[283] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[284] F. P. Bowden and A. D. Yoffe, "Initiation and growth of explosion in liquids and solids," *American Journal of Physics*, vol. 20, no. 4, pp. 250–251, 1952.

[285] M. Baer, "Modeling heterogeneous energetic materials at the mesoscale," *Thermochimica Acta*, vol. 384, no. 1, pp. 351–367, 2002. ENERGETIC MATERIALS.

[286] M. R. Baer, "Mesoscale modeling of shocks in heterogeneous reactive materials," in *ShockWave Science and Technology Reference Library*, pp. 321–356, Springer, 2007.

[287] N. K. Rai, M. J. Schmidt, and H. S. Udaykumar, "Collapse of elongated voids in porous energetic materials: Effects of void orientation and aspect ratio on initiation," *Phys. Rev. Fluids*, vol. 2, p. 043201, Apr 2017.

[288] H. K. Springer, S. Bastea, A. L. Nichols III, C. M. Tarver, and J. E. Reaugh, "Modeling the effects of shock pressure and pore morphology on hot spot mechanisms in HMX," *Propellants, Explosives, Pyrotechnics*, vol. 43, no. 8, pp. 805–817, 2018.

[289] Y. Nguyen, P. Seshadri, O. Sen, D. B. Hardin, C. D. Molek, and H. S. Udaykumar, "Multi-scale modeling of shock initiation of a pressed energetic material I: The effect of void shapes on energy localization," *Journal of Applied Physics*, vol. 131, p. 055906, 02 2022.

[290] J. Zhang and T. Jackson, "Effect of microstructure on the detonation initiation in energetic materials," *Shock Waves*, vol. 29, no. 2, pp. 327–338, 2019.

[291] J. Baum, H. Luo, E. Mestreau, and R. Lohner, *A coupled CFD/CSD methodology for modeling weapon detonation and fragmentation*, ch. 794, pp. 1–23. American Institute of Aeronautics and Astronautics, 1999.

[292] F. A. Allahdadi, F. Medina, E. T. Olson, and S. R. Jeffers, "Simulation of impact induced detonation of AIM–120. A novel approach," *Rep. No. AFSC-TR-2000–0002, Air Force Safety Center, Weapons, Space, and Nuclear Safety Division, Kirtland Air Force Base, NM*, 1998.

[293] N. K. Rai and H. Udaykumar, "Mesoscale simulation of reactive pressed energetic materials under shock loading," *Journal of Applied Physics*, vol. 118, no. 24, p. 245905, 2015.

[294] S. Kim, Y. Wei, Y. Horie, and M. Zhou, "Prediction of shock initiation thresholds and ignition probability of polymer-bonded explosives using mesoscale simulations," *Journal of the Mechanics and Physics of Solids*, vol. 114, pp. 97–116, 2018.

[295] M. A. Wood, D. E. Kittell, C. D. Yarrington, and A. P. Thompson, "Multiscale modeling of shock wave localization in porous energetic material," *Physical Review B*, vol. 97, no. 1, p. 014109, 2018.

[296] C. Handley, B. Lambourn, N. Whitworth, H. James, and W. Belfield, "Understanding the shock and detonation response of high explosives at the continuum and meso scales," *Applied Physics Reviews*, vol. 5, no. 1, 2018.

[297] J. Felts, H. Sandusky, and R. Granholm, "Development of the small-scale shock sensitivity test (SSST)," in *AIP Conference Proceedings*, vol. 1195, pp. 233–236, American Institute of Physics, 2009.

[298] P. A. Urtiew and C. M. Tarver, "Shock initiation of energetic materials at different initial temperatures," *Combustion, Explosion and Shock Waves*, vol. 41, pp. 766–776, 2005.

[299] C. Miller, D. Kittell, C. Yarrington, and M. Zhou, "Prediction of probabilistic detonation threshold via millimeter-scale microstructure-explicit and void-explicit simulations," *Propellants, Explosives, Pyrotechnics*, vol. 45, no. 2, pp. 254–269, 2020.

[300] Y. Wei, C. Miller, D. Olsen, and M. Zhou, "Prediction of Probabilistic Shock Initiation Thresholds of Energetic Materials Through Evolution of Thermal-Mechanical Dissipation and Reactive Heating," *Journal of Applied Mechanics*, vol. 88, 05 2021. 091005.

[301] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–16, 2016.

[302] M. Fathollahi and H. Sajady, "Prediction of density of energetic cocrystals based on QSPR modeling using artificial neural network," *Structural Chemistry*, vol. 29, pp. 1119–1128, 2018.

[303] M. Fathollahi and H. Sajady, "QSPR modeling of decomposition temperature of energetic cocrystals using artificial neural network," *Journal of Thermal Analysis and Calorimetry*, vol. 133, pp. 1663–1672, 2018.

[304] M. Zhongliang, X. Fangliang, L. Haiyan, and Z. Wencai, "Predicting the detonating velocity of explosives based on artificial neural network and hybrid genetic algorithm," *chinese journal of energetic materials*, vol. 15, no. 6, pp. 637–640, 2011.

[305] N. Chandrasekaran, C. Oommen, V. S. Kumar, A. N. Lukin, V. S. Abrukov, and D. A. Anufrieva, "Prediction of detonation velocity and N-O composition of high energy C-H-N-O explosives by means of artificial neural networks," *Propellants, Explosives, Pyrotechnics*, vol. 44, no. 5, pp. 579–587, 2019.

[306] M. H. Keshavarz and M. Jaafari, "Investigation of the various structure parameters for predicting impact sensitivity of energetic molecules via artificial neural network," *Propellants, Explosives, Pyrotechnics: An International Journal Dealing with Scientific and Technological Aspects of Energetic Materials*, vol. 31, no. 3, pp. 216–225, 2006.

[307] A. D. Casey, S. F. Son, I. Bilionis, and B. C. Barnes, "Prediction of energetic material properties from electronic structure using 3D convolutional neural networks," *Journal of Chemical Information and Modeling*, vol. 60, no. 10, pp. 4457–4473, 2020. PMID: 33054184.

[308] D. Walters, N. Rai, O. Sen, and W. Lee Perry, "Toward a machine-guided approach to energetic material discovery," *Journal of Applied Physics*, vol. 131, p. 234902, 06 2022.

[309] B. Gallagher, M. Rever, D. Loveland, T. N. Mundhenk, B. Beauchamp, E. Robertson, G. G. Jaman, A. M. Hiszpanski, and T. Y.-J. Han, "Predicting compressive strength of consolidated molecular solids using computer vision and deep learning," *Materials & Design*, vol. 190, p. 108541, 2020.

[310] S. Liu, B. Kailkhura, J. Zhang, A. M. Hiszpanski, E. Robertson, D. Loveland, X. Zhong, and T. Y.-J. Han, "Attribution-driven explanation of the deep neural network model via conditional microstructure image synthesis," *ACS Omega*, vol. 7, no. 3, pp. 2624–2637, 2022.

[311] A. D. Casey, *Predicting energetic material properties and investigating the effect of pore morphology on shock sensitivity via machine learning*. PhD thesis, Purdue University Graduate School, 2020.

[312] B. C. Barnes, "Deep learning for energetic material detonation performance," in *AIP Conference Proceedings*, vol. 2272, AIP Publishing, 2020.

[313] J. L. Lansford, B. C. Barnes, B. M. Rice, and K. F. Jensen, "Building chemical property models for energetic materials from small datasets using a transfer learning approach," *Journal of Chemical Information and Modeling*, vol. 62, no. 22, pp. 5397–5410, 2022.

[314] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[315] P. C. Nguyen, Y.-T. Nguyen, J. B. Choi, P. K. Seshadri, H. Udaykumar, and S. S. Baek, "PARC: Physics-aware recurrent convolutional neural networks to assimilate meso scale reactive mechanics of energetic materials," *Science advances*, vol. 9, no. 17, p. eadd6868, 2023.