Computers and Structures 270 (2022) 106825

Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

Learning mechanically driven emergent behavior with message passing neural networks

Peerasait Prachaseree, Emma Lejeune *

Department of Mechanical Engineering, Boston University, Boston, MA 02215, United States

ARTICLE INFO

Article history:
Received 9 February 2022
Accepted 8 May 2022
Available online 15 June 2022

Keywords:
Machine learning
Graph neural networks
Benchmark data
Open science
Metamodel
Surrogate model

ABSTRACT

From designing architected materials to connecting mechanical behavior across scales, computational modeling is a critical tool for understanding and predicting the mechanical response of deformable bodies. In particular, computational modeling is an invaluable tool for predicting global emergent phenomena, such as the onset of geometric instabilities, or heterogeneity induced symmetry breaking. Recently, there has been a growing interest in both using machine learning based computational models to learn mechanical behavior directly from experimental data, and using machine learning (ML) methods to reduce the computational cost of physics-based simulations. Notably, machine learning approaches that rely on Graph Neural Networks (GNNs) have recently been shown to effectively predict mechanical behavior in multiple examples of particle-based and mesh-based simulations. However, despite this initial promise, the performance of graph based methods have yet to be investigated on a myriad of solid mechanics problems. In this work, we examine the ability of neural message passing to predict a fundamental aspect of mechanically driven emergent behavior: the connection between a column's geometric structure and the direction that it buckles. To accomplish this, we introduce the Asymmetric Buckling Columns (ABC) dataset, a dataset comprised of three types of asymmetric and heterogeneous column geometries (sub-dataset 1, sub-dataset 2, and sub-dataset 3) where the goal is to classify the direction of symmetry breaking (left or right) under compression after the onset of the buckling instability. Notably, it is difficult to parameterize these structures into a feature vector for typical ML methods. Essentially, because the geometry of these columns is discontinuous and intricate, local geometric patterns will be distorted by the low-resolution "image-like" data representations that are required to implement convolutional neural network based metamodels. Instead, we present a pipeline to learn global emergent properties while enforcing locality with message passing neural networks. Specifically, we take inspiration from point cloud based classification problems from the computer vision research field and use PointNet++ layers to perform classification on the ABC dataset. In addition to investigating GNN model architecture, we study the effect of different input data representation approaches, data augmentation, and combining multiple models as an ensemble. Overall, we were able to achieve good performance with this approach, ranging from 0.952 prediction accuracy on sub-dataset 1, to 0.913 prediction accuracy on sub-dataset 2, to 0.856 prediction accuracy on sub-dataset 3 for training dataset sizes of 20,000 points each. However, these results also clearly indicate that predicting solid mechanics based emergent behavior with these methods is non-trivial. Because both our model implementation and dataset are distributed under open-source licenses, we hope that future researchers can build on our work to create enhanced mechanics-specific machine learning methods. Furthermore, we also intend to provoke discussion around different methods for representing complex mechanical structures when applying machine learning to mechanics research.

1. Introduction

Advances in additive manufacturing have made it possible to create architected materials with unprecedented control and intricacy [7,30,48,52,66,77]. To design these materials, many researchers have turned to biologically-inspired design [48,73], where

* Correspoding author.
E-mail addresses: pprachas@bu.edu (P. Prachaseree), elejeune@bu.edu (E. Lejeune).

https://doi.org/10.1016/j.compstruc.2022.106825
0045-7949/

naturally occurring microstructure such as bones [48], and complex geometry such as spiderwebs [73] are the starting point for achieving mechanical properties that exceeds what is possible with homogeneous domains and simple geometry. To complement this broad interest in exploring the design space of architected materials, multiple researchers have developed computational frameworks to both better understand and optimize architected material mechanical behavior, often through computationally expensive direct numerical simulations [69,72]. More recently, there has been a surge in applying machine learning (ML) methods to replace either computationally expensive simulations or resource demanding experiments with computationally cheap metamodels, also known as surrogate models, that directly map input parameters to output quantities of interest (QoIs) [35,36,74]. Within the broader context of ML based approaches for computational design and analysis frameworks, graph-based neural network approaches have received substantial recent attention for problems in recommendation systems [55], point cloud classification [64,78], molecule property predictions [16,21], and targeted drug design [22,84]. Fundamentally, when it comes to designing complex structures, graph based techniques are appealing because they offer a natural strategy for representing domains that are not conveniently captured as ‘‘image-like” arrays. In the context of mechanics problems, this includes problems that involve structures such as trusses [27], disordered fiber networks [67], and polycrystal microstructures [76]. In this manuscript, our goal is to further explore the efficacy of these graph-based techniques for problems in solid mechanics, where global mechanically driven emergent behavior is highly influenced by local geometric properties.

Many examples of prior work on applying ML methods to complex structural design focus on structures that can be parameterized into a feature vector [8,23,24,38,42,46,73]. For domain information that is not conveniently represented as a low dimensional feature vector, treating domain geometry as ‘‘image-like” arrays that serve as input to a Convolutional Neural Network (CNN) has become standard practice [81]. For example, CNNs have been applied to predict change in strain energy [41] and rank toughness from heterogeneous material distributions [25], predict full-field mechanical QoIs [53,82], aid in inverse design [18,28], and quantify uncertainty of partial differential equations (PDEs) [85]. For these and related examples, CNNs are an effective approach because they naturally enforce locality and spatial invariance through inductive bias with the convolution operation [5]. However, this strategy faces the fundamental limitation that arrays are often not an efficient approach for representing complex geometries, especially given the computational limitations associated with training a model on a dataset of large arrays [34]. In another recent approach, collocation methods have been used in conjunction with Physics-Informed Neural Networks (PINNs) where points are sampled within the domain as inputs to the neural networks [65,87]. Alternatively, unordered and non-grid like data structures such as point clouds [12] and meshes [9] can be readily used to store geometric information for broader applications in both computer vision [64] and mechanical simulation [61]. Furthermore, there is broad interest in the ML research community on geometric learning, or extending ML methods to graphs and non-Euclidean data structures [5,11], where non-grid like datasets are represented as graph structures. Neural networks that directly take in graphs and operate on graph structures are called graph neural networks (GNNs), and with GNNs, locality and combinatorial generalization (i.e., constructing new inferences from known relations) can be induced through spatial graph convolutions [5]. In this work, we are interested in using GNNs to predict mechanically driven global emergent behavior from geometric structures.

One of the most interesting aspects of solid mechanics is that components of a structure’s local geometry and/or material properties often interact collectively to produce global emergent behavior such as symmetry breaking [58], auxetic behavior [6,58,67], and mechanical resilience [52]. Compellingly, these extreme behaviors often *emerge* simply from the fundamental set of equations governing mechanical deformation [30]. As data driven approaches to capturing mechanical behavior gain traction [8,23,24,28,38,51,73], we anticipate that ensuring that data driven approaches are capable of capturing these emergent, and often highly non-linear, phenomena will be a key challenge. While there have been multiple successful examples of modeling buckling problems using data driven methods, these previous works do not explicitly incorporate the geometry of the structure into the ML model [47,87]. Critically, many recent advances in graph-based machine learning applied to solid mechanics problems have come from the computer science literature [61,68]. In these initial examples, domain geometry has been relatively simple, and the problems have not necessarily exhibited compelling non-linearities. In this work, our goal is to not only investigate the efficacy of graph-based machine learning approaches applied to problems with emergent behavior, but also to define a test problem that other researchers can use to evaluate their own alternative frameworks.

To accomplish this goal, we first introduce the Asymmetric Buckling Columns (ABC) dataset that consists of columns with complex geometry generated from different probabilistic distributions. For every given column, the goal is to classify its buckling direction under compression as either ‘‘left” or ‘‘right.” Then, we investigate the efficacy of neural message-passing [21], a GNN-based spatial convolution framework, for predicting this straightforward mechanically-driven global emergent behavior from local structure geometry. Specifically, we adapt the spatially-based graph convolution PointNet++ architecture originally developed to classify point cloud data structures to our dataset [64]. With the ABC dataset, we address the current lack of benchmark datasets for classification problems in mechanics. With the accompanying metamodeling pipeline, we introduce an approach to representing complex domain geometries as graph networks, and demonstrate the efficacy of spatial graph convolutions for making predictions on the resulting graph representations. Looking forward, we view this work as an important methodological step towards advancing computational design and analysis of architected materials where geometry plays an essential role in the behavior of the designed structure.

The remainder of the paper is organized as follows. In Section 2, we first discuss the generation of our finite element analysis based ABC dataset. Then, we introduce our metamodeling pipeline that includes converting each domain geometry into a spatial graph and training a message-passing graph convolution based ML model. We end Section 2 by briefly introducing simple ensemble methods to boost the performance of ML model prediction, as well as introducing methods to evaluate the efficacy of ensemble model confidence and uncertainty predictions. Next, in Section 3, we present the prediction accuracy as well as model confidence of our framework, and highlight the most effective approaches to implementing our pipeline. Finally, we present concluding remarks and potential future directions in Section 4. We note briefly that information for accessing all associated data and code is given in Section 5.

2. Methods

In this Section, we begin by introducing the open source Finite Element Analysis dataset that we generated and curated for this

project – the Asymmetric Buckling Columns (ABC) dataset. Then, we provide details of our graph neural network based metamodeling pipeline for approximating the results of these simulations. We note briefly that details for accessing the dataset, the associated simulation software, and the metamodel implementation are given in Section 5.

2.1. Defining and Generating the ABC Dataset

To date, most applications of ML methods to mechanics research have focused on problems in regression. In general, there has been less work within the mechanics community on classification problems. In addition, we note that many problems that have been treated as classification problems in mechanics could also be conveniently recast as regression problem with a threshold, e.g. tough vs. not tough [25], stable vs. unstable [42]. Critically, there are limited examples of open source benchmark datasets for testing the efficacy of ML pipelines in this area. To address this, we first introduce the Asymmetric Buckling Columns (ABC) dataset as a toy mechanics-specific classification dataset that is amenable to treatment as a graph structure and involves a global property that emerges from complex structure geometry. Specifically, the ABC dataset contains matched input and output pairs where each input is a column domain geometry and each output is the direction of symmetry breaking of the column under compression (either "left" or "right"). The dataset consists of three sub-datasets with different algorithms for input domain generation. Details of domain geometry generation for all three sub-datasets are given in Section 2.1.1. Details of the Finite Element Analysis (FEA) simulations used to determine the direction of symmetry breaking for each geometry are given in Section 2.1.2. Finally, we describe the data curation strategy and format in Section 2.1.3.

2.1.1. Domain Geometry Generation

In Fig. 1, we schematically illustrate all three sub-datasets of the ABC dataset. We chose to generate three sub-datasets to test if our metamodeling pipeline, described in Section 2.2, would function across multiple different input geometry distributions. For all three sub-datasets, the input domain generation begins with a rectangle with width (w) to length (L) ratio of 1:8. Then, all internal geometric features are algorithmically generated within the rectangular domain following procedures with varying geometric complexity. As illustrated in Fig. 1, sub-dataset 1 has the simplest underlying features while sub-dataset 3 has the most complex underlying features. Sub-dataset 1 is generated by stacking 38 blocks of height $0.025L$ with varying widths uniformly sampled from range $[0.4w, 0.9w)$ (see Fig. 1 left). Note that the top-most and bottom-most blocks cover the whole width for all geometries for consistent boundary conditions. Sub-dataset 2 consists of 200 to 300 overlapping rings with outer radius of $0.25w$ and inner radius of $0.15w$ (see Fig. 1 center). The centers of the rings are uniformly sampled from range $[0.25w, 0.75w)$. Again, the top and bottom of the column have rectangular blocks of height $0.05L$ to enable the application of consistent boundary conditions across all structures. Sub-dataset 3 consists of 1000 rings of different sizes that are overlapped and trimmed (see Fig. 1 right) The radii of the outer rings R are uniformly sampled from range $R = [0.1w, 0.25w)$, and each inner ring radius r is uniformly sampled from a range based on the corresponding outer ring radius defined as $r = [0.35R, 0.75R)$. Again, the top and bottom of the column have rectangular blocks of height $0.05L$ to ensure consistent boundary conditions.

2.1.2. FEA Simulations

We employ Finite Element Analysis (FEA) simulations to obtain the buckling direction of each column. We use PyGmsh [20,71] to mesh each domain, and open source software FEniCS [3,49] to per-

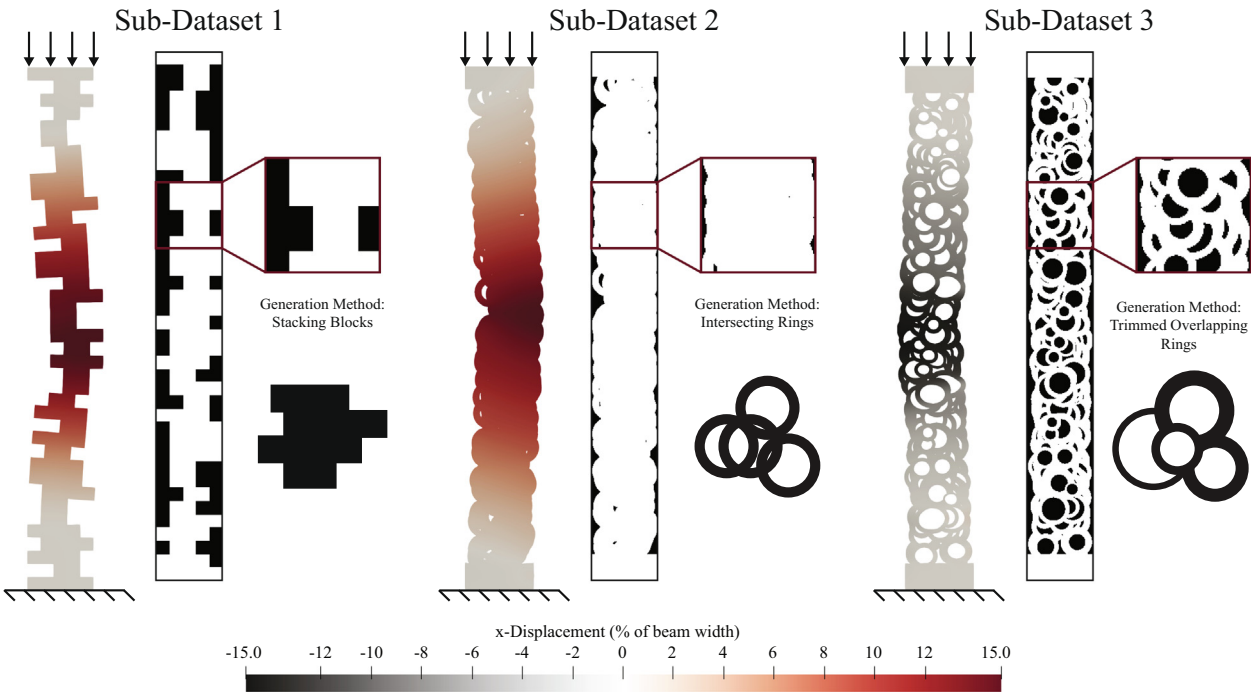

Fig. 1. Schematic illustration of our Asymmetric Buckling Columns "ABC" dataset: The dataset consists of compressed inhomogeneous columns with fixed–fixed boundary conditions. The columns are compressed until the onset of buckling instability and the buckling direction is classified as either "left" (0) or "right" (1). The dataset is split into three sub-datasets, each with varying complexity of geometric features. The geometry of the first sub-dataset (sub-dataset 1) is generated by stacking rectangular blocks. The second sub-dataset is generated by intersecting rings of uniform size. The third sub-dataset is generated by overlapping and trimming rings of varying inner and outer radii. In this figure, deformed columns are visualized with ParaView [2,4].

form the FEA simulations. Specifically, we compressed the column under a displacement control loading protocol with the column fixed at the bottom and incrementally apply y displacement at the top, schematically illustrated in Fig. 1. The left and right side of each column is traction-free. We stop the FEA simulation once the magnitude of maximum x displacement u_x is greater than $0.15w$ ($\max|u_x| \geqslant 0.15w$). To determine the buckling direction, we compare the magnitude of the maximum and minimum x displacements u_x. If $|\max u_x| < |\min u_x|$, then the column is determined to buckle left and vice versa. We note briefly that we validated this approach by visualizing a random subset of FEA results with Paraview [2,4]. For the purpose of ML model training, the columns that buckle left are assigned the label of "0" and the columns that buckle right are assigned the label of "1."

For all simulations, we used a compressible Neo-Hookean material model with the following strain energy density function:

$$\psi = \frac{\mu}{2}[\mathbf{F}:\mathbf{F} - 3 - 2\ln J] + \frac{\lambda}{2}\left[\frac{1}{2}(J^2 - 1) - \ln J\right] \quad (1)$$

where ψ is the strain energy density, $\mathbf{F}$ is the deformation gradient, $J = \det \mathbf{F}$, λ and μ are the first and second Lamé parameters, respectively. The Lamé parameters are functions of Young's Modulus E and Poisson's Ratio ν expressed as $\lambda = E\nu/[(1+\nu)(1-2\nu)]$, and $\mu = E/[2(1+\nu)]$. For these simulations, we arbitrarily set the material parameters to $E = 1$ and $\nu = 0.3$. Prior to finalizing our simulation framework, we performed convergence studies to ensure that the direction of symmetry breaking was not sensitive to either the magnitude of incrementally applied loading or level of mesh refinement. We found that an incremental displacement of $(2.5 \times 10^{-4})L$ was sufficient for solution convergence with respect to loading [33,43,44]. We also found that a mesh of quadratic triangular elements with $\approx 1,300$ elements for sub-dataset 1, $\approx 27,300$ elements for sub-dataset 2, and $\approx 40,700$ elements for sub-dataset 3 was sufficient for solution convergence with respect to mesh size. We note briefly that in some cases the generated columns did not exhibit left–right symmetry breaking and instead either skipped to higher buckling modes or were dominated by local buckling behavior in a non-straightforward manner. This occurred in under 0.6% of simulations across all sub-datasets. For simplicity, and to keep the focus on this manuscript on our ML modeling approach, we discarded these designs and generated new designs to obtain a total of $25,000$ structures with a clear direction of symmetry breaking per sub-dataset. For all sub-datasets, the direction of symmetry breaking is evenly split between the "left" and "right" classes.

2.1.3. Note on data format

As stated in Section 1, we designed the ABC dataset to address three needs: (1) the need for true classification mechanical benchmark datasets, (2) the need for mechanical benchmark datasets with non-grid like geometries, and (3) the need for additional mechanical benchmark datasets where global mechanical behavior emerges from local geometry. Releasing the ABC dataset as an open source mechanics-based classification dataset is a major contribution of this work. Here, we note briefly that we have made the ABC dataset inputs (i.e., the structure of each column) available through two independent approaches. First, in order to give other researchers maximum flexibility in adapting the components of our proposed pipeline, we provide the details necessary to reconstruct the exact input geometry. To accompany this information, we provide the code needed to re-generate the exact structures from these input parameters. Second, we directly provide graphs consisting of "spare," "medium," and "dense" nodal densities generated with the pipeline proposed in Section 2.2.1 for each input geometry. This second approach is designed to make it as easy as possible for other researchers to begin working with the ABC dataset in PyTorch Geometric [17], a GNN library built on top of the popular ML framework Pytorch [60]. Looking forward, we hope that other researchers will be able to use the ABC dataset to both design improved GNN architectures and devise new strategies to effectively represent the complex geometry of these structures. Information for accessing the ABC dataset, and the accompanying code for both geometry reconstruction and importing graph structures into PyTorch Geometric, is given in Section 5.

2.2. Metamodeling Pipeline

Our goal is to design a metamodeling pipeline that trains a ML model to effectively classify the direction of buckling (left vs. right) for the structures in our ABC dataset introduced in Section 2.1. We begin in Section 2.2.1 by outlining how we converted our heterogeneous columns into spatial graph network. The overall ML model architecture is introduced in Section 2.2.2, and further details of our message-passing convolution layer implementation are given in Section 2.2.3. We note that we employ the same set of ML model architecture and hyperparameters to separately train the metamodels for each sub-dataset.

2.2.1. Representation of Structures

As stated in Section 1, the main focus of this work is predicting the direction of buckling deformation for complex geometric structures. Specifically, we are interested in structures where representing them coarsely as image-like arrays would be inefficient and/or potentially distort their geometry and lead to loss of critical information. In this work, we will represent these complex input data with minimal loss of information by converting each geometric structure into an undirected graph [79]. As a brief background, graphs are ordered pairs $G = (V, E)$ consisting of n nodes (also referred to as vertices) $V = \{v_1, v_2, \ldots, v_n\}$ and edges $E \subseteq V \times V$ (edges are constructed by connecting nodes). For undirected graphs, the edges are bidirectional. In general, graphs can contain both nodal features $\mathbf{f}_i$ that contains information associated with nodes, and edge features $\mathbf{e}_{ij}$ that define the relationship between two nodes.

In Fig. 2a, we show our methodology for converting geometric structures into undirected graphs. We note that every step in our pipeline is modular, and other methods to generate nodes and determine their connectivity, such as methods in point cloud processing [78], could also be applied in the context of our broader analysis pipeline. The first step of our approach is to create a high resolution image representation (100×800) of the domain. Then, we segmented the high resolution image array with simple linear iterative clustering (SLIC) [1] using the package sci-kit image (skimage) [75]. With this approach, each segmented "superpixel" becomes a node in a spatial graph with nodal features of centroid position, area, and eccentricity. Since the SLIC algorithm can vary the number of superpixels in each image, it is possible to get a different density of nodes for the same structure by changing this parameter. The nodal density of the graph is a parameter that we will investigate in Section 3.2. After segmentation, the Region Adjacency Graph (RAG) of the structure is formed by discarding superpixels not associated with the structure (i.e., the superpixels associated with the dark areas in Fig. 1), and adding edges between adjacent remaining superpixels. The RAG is one method of data representation that we will explore in Section 3.3. In addition, we also investigate a "Ball Query" based method of data representation where the structure-associated superpixel nodes are connected to form an undirected graph via a ball query algorithm [64]. Specifically, the ball query algorithm constructs edges by searching for nodes within a prescribed and tunable radius from the central node ($E_{ij} = \{\{i,j\} \mid \|\mathbf{x}_j - \mathbf{x}_i\| \leqslant r \text{ and } i,j \in V\}$) where $\mathbf{x}_i$

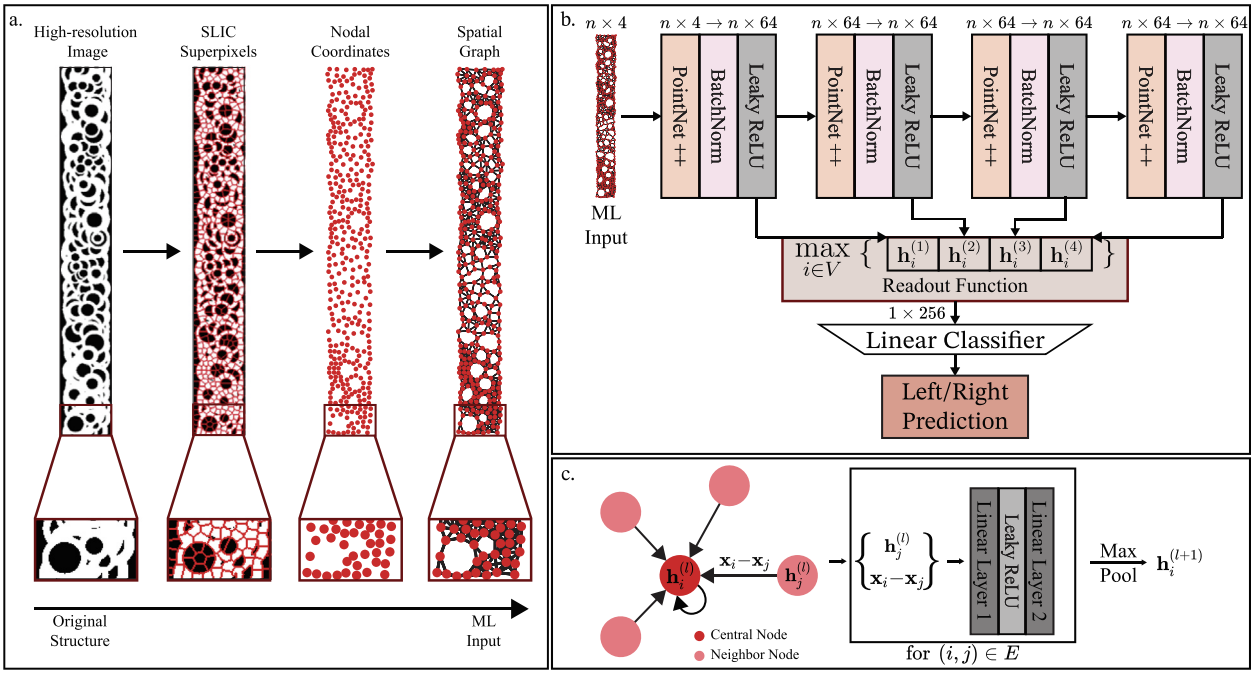

Fig. 2. An overview of our metamodeling pipeline: (a) Steps to convert the high-resolution image representation of the structure to an undirected graph. (b) Architecture used to predict left/right symmetry breaking. We use 4 PointNet++ Layers to generate embeddings $\mathbf{h}_i^{(l)}$ (nodal features of node i in layer l) [64] with skip-connections [29] after each batch normalization layer to obtain the final node embedding, and global max-pooling as the readout function before passing the final embedded vector through a linear classifier for "left" vs. "right" prediction. The corresponding labels $n \times 4$ and $n \times 64$ denote the number of nodes n and the length of the current embedding vector. c) A schematic of our implementation of a Pointnet++ Layer with spatial graph convolutions. We construct our message functions for message-passing by concatenating node features $\mathbf{h}_j^{(l)}$, which are the embedded feature vector of the neighboring node, and edge features, which are formed from the difference of positions $\mathbf{x}_j - \mathbf{x}_i$. The message function $\mathbf{h}_i^{(l)}$ is updated using a Multi-Layer Perceptron (MLP) with one hidden layer aggregated with max pooling. The resulting vector $\mathbf{h}_i^{(l+1)}$ is the new embedding that is passed in the next step of t.he architecture.

and $\mathbf{x}_j$ are the positions of the central and neighboring node respectively and r is the radius of ball query). For our spatial graphs, the edges for both the RAG and ball query approaches carry feature vectors $\mathbf{x}_j - \mathbf{x}_i$ that describe the distances between the two nodes. We note that all the node feature vectors are normalized after ball query to ensure faster training convergence.

Alternatively, we note that structures from sub-dataset 1 and sub-dataset 2 can also be simply represented as spatial graphs without loss of information. In Section 3.3, we investigate the effectiveness of using these "exact structures" as ML model inputs. In the case of sub-dataset 1, we obtain "exact structures" by first converting each pixel of the structure into a node with a nodal feature vector containing only nodal position. Then, we connect neighboring pixels with edges that contain the relative difference in positions of connected nodes $(\mathbf{x}_j - \mathbf{x}_i)$. Since each pixel is the same size and shape, additional information regarding the geometry of each pixel would be redundant. For sub-dataset 2, we represent the "exact structure" by treating the center of each ring as a node where graph edges denote rings that overlap each other. Similarly, the edges contain the relative difference in positions of the connected nodes $(\mathbf{x}_j - \mathbf{x}_i)$. Again, since the shape and size of each ring in sub-dataset 2 is identical, additional information regarding shape and size would be redundant. With this approach, no information is lost in describing the geometry – thus we refer to it as "perfect representation." We note briefly that for sub-dataset 3, the trimmed rings (see Fig. 1) preclude a perfect representation approach.

2.2.2. Machine Learning Model Architecture

The goal of our ML task is to predict the direction of buckling (left or right) from the geometry of the structure. Since buckling behavior is a global emergent behavior that arises from local geometric features [7,30,59], we chose to use a convolution based approach which inherently enforces locality through inductive bias. However, traditional Convolutional Neural Networks (CNNs) [40] are not designed for data that is not grid-like (e.g., graph structure data [79], point cloud data [12], mesh data [61]). Instead of CNNs, we draw from recent interest within the computer vision research community in classifying point clouds, unordered and non-gridlike sets of points in space, with Graph Neural Networks (GNNs) and spatial graph convolutions [54,78]. These geometry-encoding point clouds are related to our geometric graph structures because they are both required to encode *spatial* information to drive classification. As such, our ML model starting point is the previously developed PointNet++ architecture of spatial graph convolution layers [64]. While one of the perceived drawback of PointNet++ is the lack of inherent rotation invariance in the convolution layers [64], not having inherent rotation invariance is advantageous for our dataset where we classify left–right symmetry breaking since the direction of buckling (and the corresponding labels) change depending on the orientation of the column. In Fig. 2b, we schematically illustrate our proposed machine learning model architecture. Specifically, we applied 4 PointNet++ layers (See Section 2.2.3) with skip connections [29] to help smooth the gradient flow during back propagation. In the context of graph convolutions, skip connections have also been shown to help prevent over smoothing by preserving local information [86]. We implement these skip connections by concatenating each embedded nodal feature vector $(\mathbf{h}_i^l)$ after every convolution layer into a final nodal embedding vector (see Fig. 2b). The global graph embedding is then obtained by using global max-pooling on the concatenated nodal features as the readout function (highlighted $\max_{i \in V}$ in Fig. 2b). The resulting global graph embedding vector is subsequently used to predict the direction of symmetry breaking via a

linear classifier. To train this model, we use the cross-entropy loss function, and the ML model is trained for 50 epochs with an Adam optimizer [37] before convergence. In short, the ML algorithm takes in nodal features and computes the global graph embedding through spatial graph convolution layers. The global graph embedding is then passed through a linear classifier to predict the buckling direction of the structure. To ensure reproducibility, all ML models are trained with 10 different weight initialization seeds. For each sub-dataset, the ML model is trained on $20,000$ data points. We also initially deployed and tuned the hyperparameters and architecture of our ML models on validation data ($2,500$ data points) before finally evaluating our model on held out test data ($2,500$ data points).

2.2.3. Spatial Graph Convolution Layer Implementation

We implemented the PointNet++ spatial graph convolutions using the Pytorch Geometric library [17]. For tasks involving prediction of whole graphs, GNNs that utilize spatial graph convolutions, also known as message passing neural networks [21], can be summarized in two phases: the message passing phase, and the readout phase. During the message passing phase, the nodal embeddings are recurrently updated through "messages." These "messages" are constructed from current central nodal embeddings $\mathbf{h}_i^{(l)}$ (or nodal feature vectors $\mathbf{f}_i$) and neighboring embeddings $\mathbf{h}_j^{(l)}$. Note that neighboring nodes in terms of graphs refer to nodes that are directly connected by an edge (see Fig. 2c). New nodal embeddings $\mathbf{h}_i^{(l+1)}$ are obtained by putting the current embeddings through an update function and combining them by an aggregation function. While the update function can be any differentiable function [5,21], in the case of GNNs, the most common update function is a Multilayer Perceptron (MLP) [31]. The aggregation function is also a differentiable function with the additional constraints of being invariant to input node order and able to take in a variable number of inputs. Examples of commonly used aggregation function include element-wise sum, mean pooling, and maximum/minimum pooling [5]. Each update-aggregation pairing is also referred to as a spatial graph convolution layer. Note that in general, while our implementation passes the message through the update function before the aggregation function, the order of update and aggregation can be interchanged [5]. During the message passing phase, the nodal embeddings will be updated a prescribed number of times prior to the readout phase. During the readout phase, we then compute a global graph feature vector through readout functions that are also invariant to node ordering. Common readout functions, similar to aggregation functions, are mean-pooling, min/max-pooling, and sum-pooling applied to all of the node embeddings in the graph [5]. For ease of implementation, we employ EdgeConv [78] (a generalized framework for performing spatial convolutions on point clouds) to perform spatial graph convolutions during the message passing phase on our spatial graph. In general, EdgeConv also uses the edge features alongside nodal features to construct messages for message passing. We note again briefly that since our pipleline is modular, variants of EdgeConv or alternative spatial graph convolutions layers could be applied here instead in future work.

For our ML model implementation of message passing, each node feature $\mathbf{f}_i$ consists of nodal position $\mathbf{x}_i$, superpixel area, and superpixel eccentricity (eccentrity is defined as $e = \frac{c}{a}$ where c and a are the focal distance and major axis length respectively of an ellipse with the same second moment of inertia as the superpixel). Each edge feature contains the difference in position between the central node and the connected neighboring node ($\mathbf{x}_j - \mathbf{x}_i$ with $\mathbf{x}_i$ and $\mathbf{x}_j$ denoting the position of the central and neighboring node respectively). As seen in Fig. 2c, each Pointnet++ layer constructs its message by concatenating the current nodal features and the edge features. The message is passed through a MLP [31] with 1 hidden layer (our update function), and aggregated with max-pooling to update the nodal embedding. Note that self-loops (edges that connect the central node to itself) are present in our graphs. As such, the new embedded feature $\mathbf{h}_i^{l+1}$ vector is expressed as:

$$\mathbf{h}_i^{(l+1)} = \max_{(i,j)\in E} \text{MLP}\{\mathbf{h}_j^{(l)}, \mathbf{x}_j - \mathbf{x}_i\} \tag{2}$$

where $\mathbf{h}$ denotes the nodal embedding vectors. Subscripts i and j denote the central node and neighboring node respectively, and superscript l denotes the convolution layer number. We initialize $\mathbf{h}_i^{(0)}$ as $\mathbf{f}_i$ (i.e., the first input to the first PointNet++ Layer is the nodal feature vector). After the first layer, all the embedding vectors have a length of 64. We note that all features except for position are min–max scaled. After every PointNet++ Layer, we apply batch normalization (BatchNorm) [32] to accelerate training and to provide some additional regularization. We apply the Leaky ReLu activation function [50] after the BatchNorm layer, and the resulting embedding nodal vectors are passed onto the next layer. After the last convolutional layer, our readout function is max-pooling, as described in Section 2.2.2. In summary, during the message passing phase, our spatial convolutional layers construct messages from initialized nodal features and edge features that recurrently update with each convolutional layer. For the readout phase, we use max-pooling to generate an embedding of each graph and then use it to classify the buckling direction of each corresponding geometric structure.

2.2.4. Ensemble Learning

In this work, we train our ML model 10 times for each dataset with 10 different random weight initializations (i.e., different seeds). For each ML model initialization, we obtain a slightly different set of predictions. Thus, each ML model acts as a different classifier. With these different classifiers on hand, we are then able to use ensemble methods to increase the overall prediction accuracy of our ML model framework. Broadly speaking, there are many different ensemble methods such as boosting [70], stacking [80], and bagging [10], that have been shown to increase overall prediction accuracy by leveraging multiple ML models [13,19,62]. In this work, we employ a simple voting approach that requires no additional training beyond creating the initial ML models [13]. Specifically, we investigate if a voting based approach will enhance our GNN ML model framework. We investigate predictions obtained by combining 10 trained ML models through both hard and soft voting. For hard voting, also referred to as majority voting, the predicted labels for each input are counted and the label with the highest count becomes the final prediction. For soft voting, also referred to as unweighted model averaging, the predicted probability of classes for each label is averaged over all ML models. The final predicted label is then the class with the highest average probability. In Section 3.4, we show the performance of both hard and soft voting. Additionally, since each individual ML model is trained with a cross entropy loss, the computed probabilities via soft voting are also the ensemble's prediction confidence [39]. Broadly speaking, ML models are most useful for engineering applications when they are able to produce a meaningful confidence level for each prediction. In this context, a ML model is considered "well-calibrated" when model confidence reflects the true probability that the model is correct [26,56,57]. By way of example, if a ML model is perfectly calibrated and is 80% confident for 100 predictions, 80 of those predictions will be correct. Reliability diagrams, also referred to as calibration curves, are a common visual approach to assessing the quality of model confidence. Briefly, reliability diagrams are constructed by first separating the prediction confidence that a given sample is in a chosen class into sequential

bins, and then comparing the bin average to the true fraction of samples with the chosen class in each bin. In the context of the ABC dataset, we can construct a reliability curve by choosing the "right" buckling class and placing "model confidence" on the x-axis and "fraction buckled right" on the y-axis. A perfectly-calibrated model will exhibit a reliability curve equal to the identity function [26,56,57]. In addition to visualizing these curves, we also report quantitative metrics for model calibration. Specifically, we compute the Expected Calibration Error (*ECE*) and the Maximum Calibration Error (*MCE*) from each reliability diagram [56]. The *ECE* is computed by taking a weighted average the of gaps between perfect calibration and model confidence within each bin, written as:

$$ECE = \sum_{i=1}^{B} \frac{n_i}{N} |F_i - C_i| \tag{3}$$

where B is the number of bins, n_i is the number of samples in each bin, N is the total number of samples, F_i is the frequency of class 1 in the bin, and C_i is average confidence that the sample is in class 1 in the bin). The *MCE* is defined as the maximum gap in each reliability diagram, written as:

$$MCE = \max_{i \in \{1,\dots B\}} |F_i - C_i|. \tag{4}$$

In Section 3.5, we investigate the model calibration of our best-performing ensembles in each sub-dataset.

3. Results and Discussion

In this Section, we investigate multiple approaches towards designing a metamodeling pipeline and subsequently increasing its prediction accuracy. Section 3.1 describes our data augmentation strategy that takes advantage of the symmetries in our training dataset. Next, in Section 3.2, we explore the effect of graph nodal and edge density on the prediction accuracy. Then, in Section 3.3, we perform a brief study on different strategies to represent structures as graphs. In Section 3.4 we leverage ensemble learning to obtain higher prediction accuracies. We end by examining model calibration in Section 3.5 We note that all results reported from here onward are obtained from held out test data.

3.1. Data Augmentation Enhances Prediction Accuracy

Here, we present the performance of our PointNet++ architecture on both un-augmented and augmented versions of each sub-dataset. We perform data augmentation by reflecting each column in the input domain about the x axis, y axis and both axes. When the column is reflected about the y axis and both axes, the associated output label is changed due to the symmetry of the problem. This method of data augmentation is only feasible because of the lack of rotational invariance in the PointNet++ Layers. We note briefly that in these studies, we begin data augmentation with the ground truth geometries of the columns before segmentation and conversion into point clouds (see Fig. 2a). The augmentations are then combined with the original training dataset. By reflecting the geometries rather than the graphs themselves, the graphs of the reflected structure will be slightly different than if the graphs in the original training dataset had simply been rotated due to the randomness associated with the segmentation process. These variable graph representations also enforce the notion that slightly different graphs can represent the same structure.

In Fig. 3 and Table 1, we demonstrate the effectiveness of data augmentation in increasing model predictive performance. Note that each data point contains the mean $\pm 95\%$ confidence interval (CI) of 10 separately trained models with the same architecture but with different weight initializations. In Fig. 3, we show test prediction accuracy with respect to the number of unaugmented training datapoints for both unaugmented and augmented training datasets. The results shown in Table 1 (values obtained from Fig. 3) are the highest prediction accuracy for each sub-dataset. Based on these results, it is clear that our approach to data augmentation is highly effective at increasing prediction accuracy. Specifically, it increases prediction accuracy by 3% to 8% in the case of $20,000$ training points. And, from the results presented in Table 1, we can see that the accuracy between training the model without data augmentation with $20,000$ data points (Table 1 column 1) is nearly equivalent to training the model with $5,000$ data points (Table 1 column 2) but with data augmentation. This demonstrates that our data augmentation method is comparable to increasing the training set size by a factor of 4. In addition, as shown in Fig. 3, even with data augmentation, the trend of the accuracy is still increasing at $20,000$ datapoints. This indicates that more datapoints (i.e., a larger training dataset) would likely improve the model prediction accuracy. We also note that as anticipated, the highest prediction accuracy is in sub-dataset 1, which has the simplest features, while sub-dataset 3 has the lowest accuracy corresponding to the most complicated features.

3.2. Node Density and Query Radius Influence Prediction Accuracy

In this Section, we examine the influence of node and edge density of the graph representations on model prediction accuracy. We controlled graph nodal density by prescribing "sparse," "medium," and "dense" levels of segmentations to the high-resolution images (see Fig. 2a). Details on the distribution of node densities for each sub-dataset as well as visualizations of graphs from each node density are presented in Appendix A. We controlled edge density by defining different query radii of $r = 0.2w, 0.3w, 0.4w$, where w is the width of the column. Note that for a fixed ball query radius, simply increasing the nodal density will increase the edge density of the graph. In Fig. 4, we illustrate the effect of graph edge density on prediction accuracy. Again, we note that each data point contains the mean $\pm 95\%$ confidence interval (CI) of 10 separately trained models with the same architecture but different initializations. In all cases, the ML model is trained with an augmented dataset representing $20,000$ simulation results. As demonstrated in Fig. 4, regardless of neighborhood size, the prediction accuracy increases when edge density increases up to a maximum prediction accuracy where adding in additional edges does not improve performance. This result may be due to the fact that after each message passing layer, the subsequent message passing layers will contain information from the next hop (neighbors of neighbors), thus higher edge density is redundant [86] since no additional information about the geometry will be added. With our chosen model architecture of 4 Pointnet++ layers, each node will have information about its 4-hop neighbors. The average number of edges for which the highest prediction accuracies are achieved is 2046 edges for sub-dataset 1 ("medium" node density with query radius of $r = 0.4w$), 3815 edges for sub-dataset 2 ("dense" node density with query radius of $r = 0.3w$), and 7479 edges for sub-dataset 3 ("dense" node density with query radius of $r = 0.3w$). Again, the maximum prediction accuracy is shown in Table 1 column 4.

3.3. Ball Query Graph Representation Out-Performs the Region Adjacency Graph and Perfect Structure Representation

Here, we investigate three approaches to column geometry representation and observe the effect of each approach on pipeline prediction accuracy. We note that aside from the different input

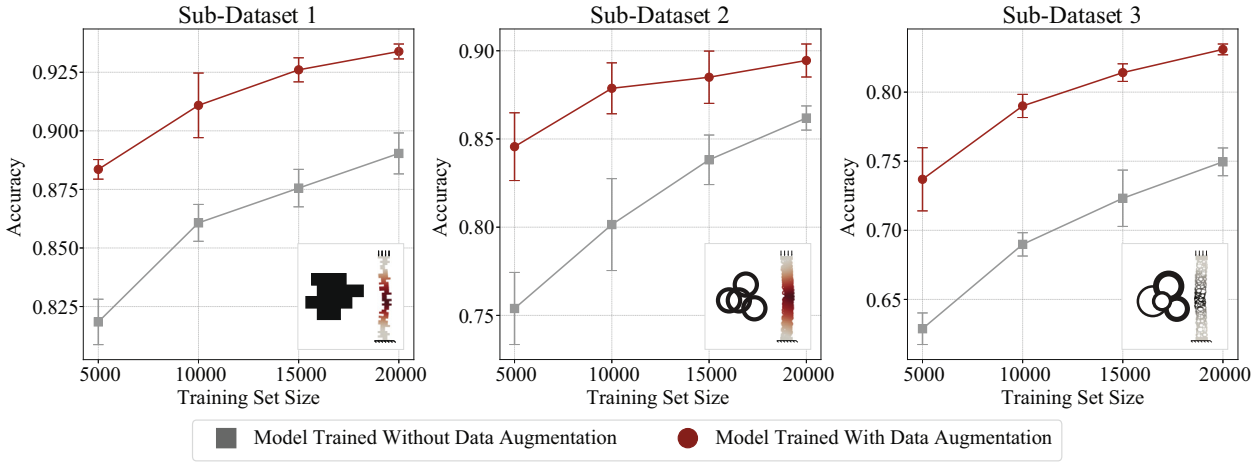

Fig. 3. Effect of data augmentation on test accuracy. The markers show the mean of 10 trained models with different weight initialization and the error bars represent a 95% confidence interval. Note that the scales for the y-axis are different for each plot. The dataset is augmented by reflecting the columns about the *x* axis, *y* axis and both axes. Note that the output labels is changed when the column is reflected about the *y* axis and both axes.

Table 1
Comparison of the model test accuracy on the original training dataset and the augmented training dataset. Tabulated values are the mean and 95% CI of the model with 10 different initializations. The values shown in this table are the best prediction accuracies we obtained using this metamodeling pipeline. Prediction accuracies are obtained from Fig. 3.

	Original Dataset		**Augmented Dataset**	
	5 K training points	20 K training points	5 K training datapoints	20 K training datapoints
Sub-Dataset 1	0.818 ± 0.010	0.890 ± 0.009	0.884 ± 0.004	0.934 ± 0.003
Sub-Dataset 2	0.754 ± 0.020	0.861 ± 0.007	0.846 ± 0.019	0.894 ± 0.009
Sub-Dataset 3	0.628 ± 0.011	0.749 ± 0.010	0.737 ± 0.023	0.831 ± 0.004

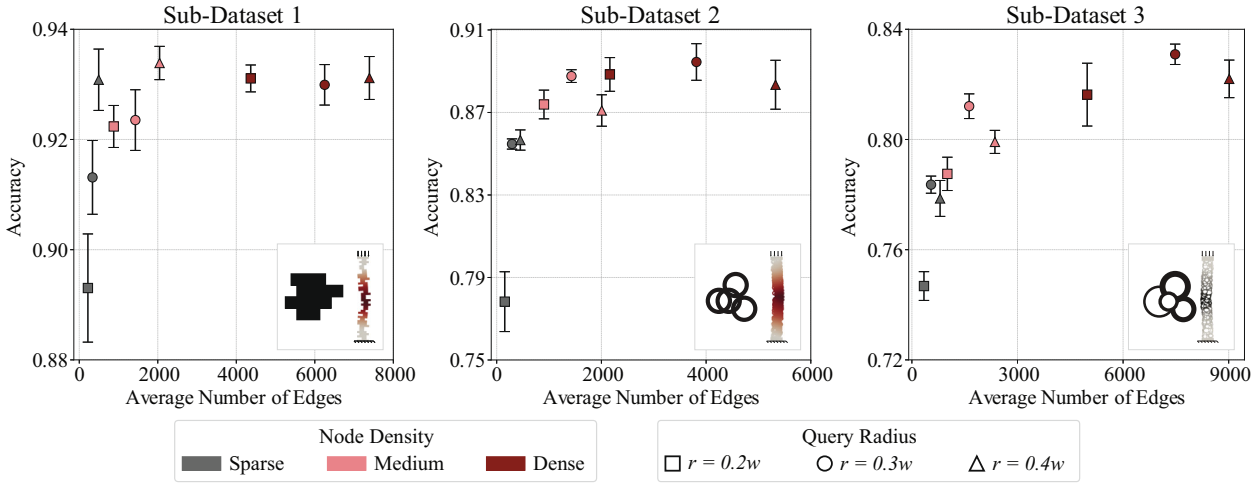

Fig. 4. These plots show the importance of graph structure representation. Each marker represents the mean of 10 trained models with different initializations and the error bars represent the 95% confidence interval. The parameters varied are the number of superpixels during segmentation ("sparse," "medium," and "dense"), and the ball query radius. The plots show that increasing the edges density of the graphs increases test prediction accuracy up to a point. Note that the scale for the y-axis is different for each plot.

graphs for training, validating, and testing phases, everything else in the pipeline (e.g. the ML architecture and the 10 different weight initializations) remains the same. As stated in Section 2.2.1, we investigate ball query representation as the primary approach, and region adjacency graph (RAG) representation and "exact structure" representation as two potential alternatives. Fig. 5 illustrates the test prediction accuracies of each input representation type. For sub-dataset 1 and sub-dataset 3, RAG representations perform slightly worse than the ball query approach. For sub-dataset 2, RAG performs comparably to ball query, but is still marginally worse. We note that RAG representations can be seen as a special case of the ball query algorithm where only neighboring superpixels form edges (i.e., query radius r matches the superpixel size), leading it to perform similarly to graphs constructed using the ball query approach. Critically, we note that despite no loss of geometric information, the "exact" representation approach performs

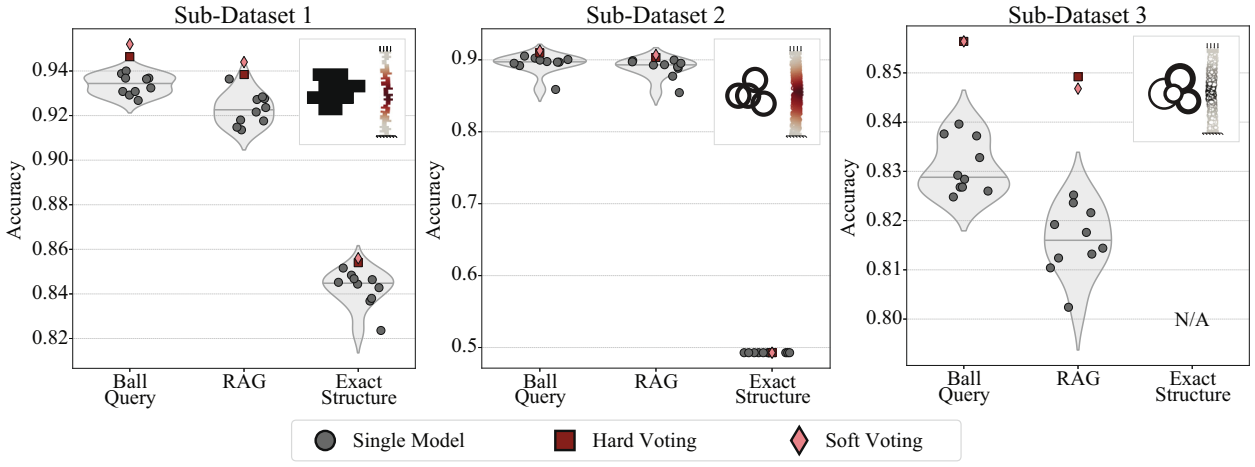

Fig. 5. Test accuracy with respect to graph structure representation method for an augmented training dataset of 20,000 simulations with node density and ball radius informed by the parameter sweep in Section 2.2.1. In each plot, the black circle marker shows the accuracy of an individual ML model. The red square and pink diamond show aggregated accuracy from hard voting and soft voting respectively. For each dataset, we show results from input graphs created through ball query, RAG, and exact representation. Note that the scale for the y-axis are different for each plot, and that for sub-dataset 3 there is no "exact structure". representation.

worse than both ball query and RAG for both sub-dataset 1 and sub-dataset 2 (for sub-dataset 3 there is no "exact" structure formulation available due to the trimming step of geometry generation). The difference in performance is particularly striking for sub-dataset 2, where the "exact" representation prediction accuracy is $\sim 50\%$ for all model initializations, indicating that the ML model does not perform better than random guessing. We believe that this was because PointNet++ was initially developed to deal with point cloud datasets such as ShapeNet [12,64], so point cloud analogous geometric representations work best with this pipeline.

Table 2
Comparison of test prediction accuracies for individual model mean, best individual model, hard voting, and soft voting. This table reflects the 10 ball query based models for each sub-dataset shown in Fig. 5.

	Sub-Dataset 1	Sub-Dataset 2	Sub-Dataset 3
Mean of 10 Models	0.934	0.894	0.831
Best Individual Model	0.937	0.905	0.839
Hard Voting	0.946	0.910	0.856
Soft Voting	0.952	0.913	0.856

We note that there could potentially be other ML model architectures where exact geometry representations would perform better, but exploration of these architectures is outside the scope of this paper.

3.4. Ensemble Learning Enhances Prediction Accuracy

For each sub-dataset, we independently train 10 ML models with the same hyperparameters but different weight initializations. Therefore, we can investigate not only the average and range of model accuracy, but also the efficacy of applying ensemble learning methods to leverage all independently trained models to boost performance. Specifically, we employ both hard voting and soft voting to aggregate our predictions from the available ML models (see Section 2.2.4 for details on hard and soft voting). The plots in Fig. 5 demonstrate the effectiveness of both voting methods for increasing the prediction accuracy for column representations produced from our proposed metamodeling pipeline outlined in Section 2.2.1, particularly for sub-dataset 3. In all cases, the final predictions obtained via voting lead to slightly better prediction accuracy than the best individual accuracy from the 10 dif-

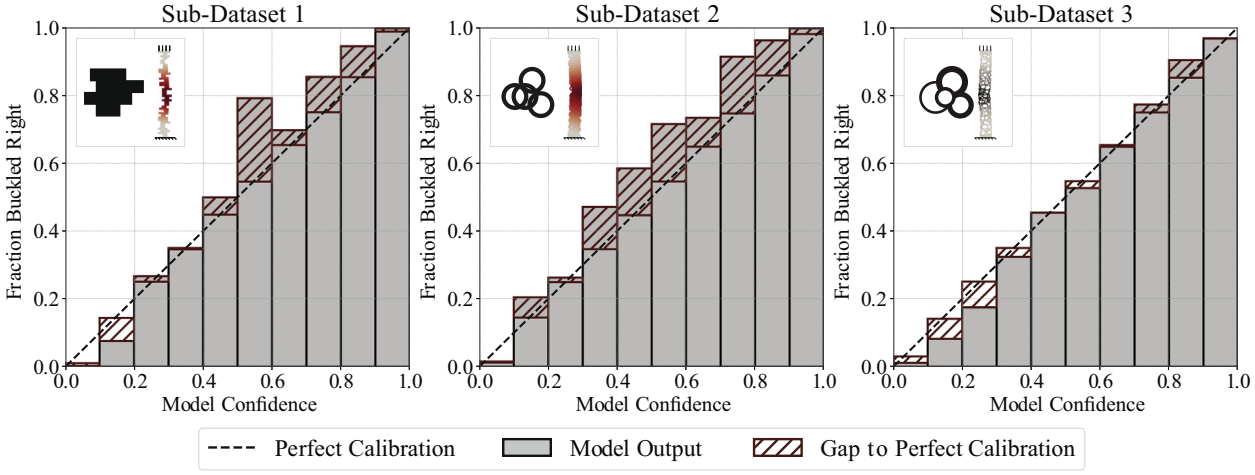

Fig. 6. Reliability Curves for the best performing ensemble of 10 models. Models are trained on $r = 0.4w$ with "medium" node density for sub-dataset 1, and "dense" node density with $r = 0.3w$ for sub-dataset 2 and sub-dataset 3. A perfectly calibrated model would be a perfect diagonal. In this plot, the gap between observed and perfect calibration is noted with the hatched pattern.

Table 3
Comparison of Expected Calibrated Error (*ECE*, see Eqn. 3) and Maximum Calibrated Error (*MCE*, see Eqn. 4) for ensemble of ML models. A perfectly calibrated model would have *ECE* and *MCE* of zero.

	Sub-Dataset 1	Sub-Dataset 2	Sub-Dataset 3
ECE	0.024	0.041	0.023
MCE	0.246	0.169	0.076

ferent model initializations (see Table 2). For sub-dataset 1, soft voting performed better than hard voting and lead to an accuracy increase of $\sim 0.15\%$. For sub-dataset 2, soft voting performed better than hard voting and lead of an accuracy increase of $\sim 0.076\%$. For sub-dataset 3, soft voting and hard voting performed similarly and lead to an accuracy increase of $\sim 1.7\%$. The efficacy of ensemble voting with our classifier suggests that differently initialized models potentially learn different geometric features and combining these trained models leads to overall variance reduction [10,19,62,70]. The visualizations of the important edges with GNNExplainer [83] for each sub-dataset in Appendix B further support the notion that different geometric features are learned in different initializations. However, due to the different geometrical features that are highlighted depending on model initialization, it is not clear if important insights on the geometry can be inferred from GNNExplainer.

3.5. Ensemble Learning Displays Well-Calibrated Confidence

In addition to evaluating the accuracy of our ML models, we are also interested in evaluating the confidence of our ensemble of predictions. As referred to in Section 2.2.4, reliability diagrams are a useful visual tool to evaluate model calibration. To construct our reliability diagrams, we first separate the model predictions into bins according to the model confidence that the input column is in class 1 (i.e. buckles right), and then count the actual number of class 1 labels in the bin. The number of class 1 labels is then normalized with the number of samples in each bin. Fig. 6 shows the reliability diagrams for the ensemble of 10 best-performing models defined in Section 3.2 that are trained on "medium" node density with $r = 0.4w$ for sub-dataset 1, and "dense" node density with $r = 0.3w$ for both sub-dataset 2 and sub-dataset 3. In Table 3, we report the *ECE* and *MCE* for each ensemble. Recall that for model confidences to be well-calibrated, the reliability diagram must be close to the diagonal (i.e., the black dotted line in Fig. 6). As such, perfectly-calibrated models will also have *ECE* and *MCE* values that are close to zero. Visually, the reliability diagrams in Fig. 6 shows that our ML models are slightly underconfident (lower confidence compared to fraction of class 1), but overall still exhibit reliable behavior. Quantitatively, the *ECE* and *MCE* for each ML models is relatively low, especially in the case of sub-dataset 3. In Appendix C, we further investigate potential differences in the deformation profile between columns from different model confidence levels.

4. Conclusion

In this paper, we introduce the Asymmetric Buckling Columns (ABC) dataset, a mechanics specific classification dataset of spatially heterogenous columns under compression where local geometric structure influences the global emergent buckling direction. The dataset consists of 3 sub-datasets, each constructed from basic geometric features with varying degrees of complexity. Our goal in defining these complex geometries was to create a dataset where treating each column as an image-like array would not be the obvious choice for data representation. Then, to explore alternatives to "image-like" data representations while keeping the benefits of inductive biases from the convolution operation, we proposed a metamodeling pipleline that is broadly applicable to any structure with complex geometric features, and tested its performance on the ABC dataset. The proposed pipeline first converts the columns into spatial graphs via image segmentation, then employs a ML architecture comprised of PointNet++ spatial graph convolution layers [64] to classify each column based on its buckling direction. Within the scope of our proposed pipeline, we also investigated strategies to increase prediction accuracy through data augmentation, alternative spatial graph formulations, and ensemble learning. We showed that our data augmentation scheme improved the prediction accuracy comparably to increasing the training set size by a factor of 4. We also demonstrated that choosing a suitable edge density to represent our structure is key to obtaining the best prediction accuracy. Additionally, we improved the overall prediction accuracy by applying voting methods to our 10 separately trained models. Finally, we showed that our proposed ensemble of metamodels is well-calibrated.

Looking forward, we hope that this work will serve as a platform for other researchers to explore alternative data representations for ML pipelines in mechanics, especially in problems where geometry plays an important role in the emergent behavior of the system. To facilitate this, we have released the ABC dataset, the code to generate the dataset, and the code to implement our proposed ML pipeline all under open source licenses. The availability of these resources will allow others to either apply our ML pipeline to alternative problems, or benchmark alternative methods on the same dataset. Beyond the scope of the classification problem presented in this paper, it would also be interesting to see how different structure representation strategies and GNN architectures perform for predicting full-field QoIs for various problems in mechanics. Specifically, while there are reported successes in predicting full-field QoIs in a variety of mechanics problems [53,82,85], and examples of applying GNNs to predict full-field QoIs from physics-based simulations [61,68], there are limited examples in the literature that predict full-field QoIs of truly emergent mechanical behavior from unstructured and non-gridlike data. Similarly, substantial future work is needed in curating and disseminating benchmark datasets of complex structures that are experimentally tested [24,47] rather than simulated. Due to the time and cost for experiments, methods to combine experimental data and computational data through methods such as transfer learning should also be explored [23,45]. Another interesting direction for future work is in the interpretability of GNNs especially when applied to problems concerning solid mechanics. While GNN interpretibility for problems in computer vision typically focuses on qualitative visualizations [83], there have also been important recent advances in interpreting learned physical laws as simple symbolic equations with GNNs on discrete n-body problems [14,15]. It would be interesting to see if similar methods could be applied to problems in solid mechanics to potentially interpret the complex relations between structural behavior and domain geometry. To this end, we view the creation of the ABC dataset and our initial pipeline as a starting point for substantial future work at the intersection of machine learning and mechanics.

5. Additional Information

The ABC dataset is available through the OpenBU Institutional Repository at https://open.bu.edu/handle/2144/43730 [63]. All code used to generate the domain geometry and labels is available at https://github.com/pprachas/ABC_dataset. All code for implementation of the metamodeling pipeline from spatial graph gen-

eration to PointNet++ layers is also available at https://github.com/pprachas/ABC_dataset.

Declaration of Competing Interest

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

Acknowledgements

We would like to thank the staff of the Boston University Research Computing Services and the OpenBU Institutional Repository (in particular Eleni Castro) for their invaluable assistance with generating and disseminating the Asymmetric Buckling Columns (ABC) Dataset. This work was made possible through start up funds from the Boston University Department of Mechanical Engineering, the David R. Dalton Career Development Professorship, the Hariri Institute Junior Faculty Fellowship, the Haythornthwaite Research Initiation Grant, and the National Science Foundation grant CMMI-2127864.

Appendix A. Node density

Fig. 7 illustrates the histograms of graph node densities from using the metamodeling pipeline proposed in Section 2.2.1. We note that the node densities are not fixed for each dataset since

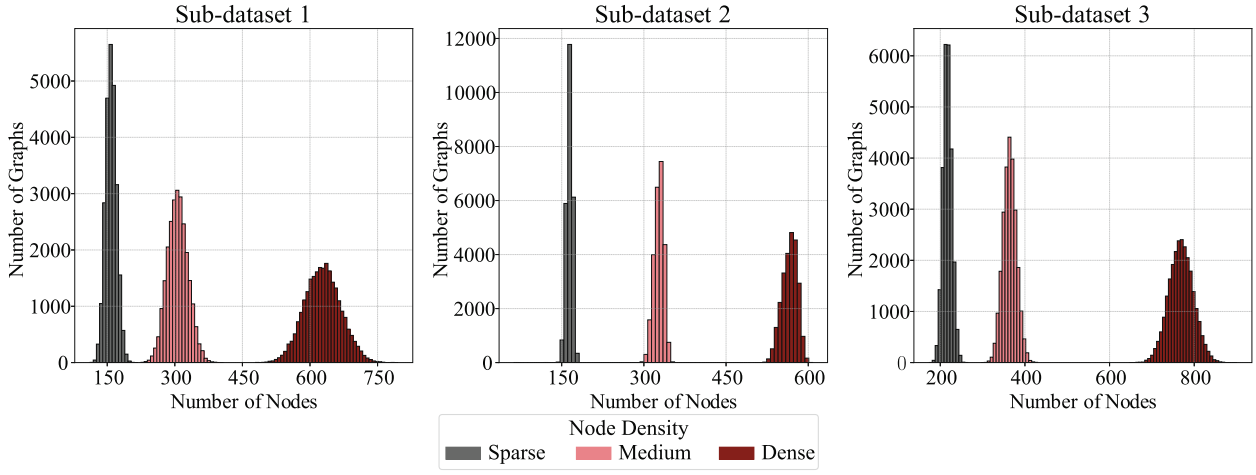

Fig. 7. Histogram of graph node densities from our dataset. Each plot contains histograms of sparse, medium, and dense node densities for each sub-dataset.

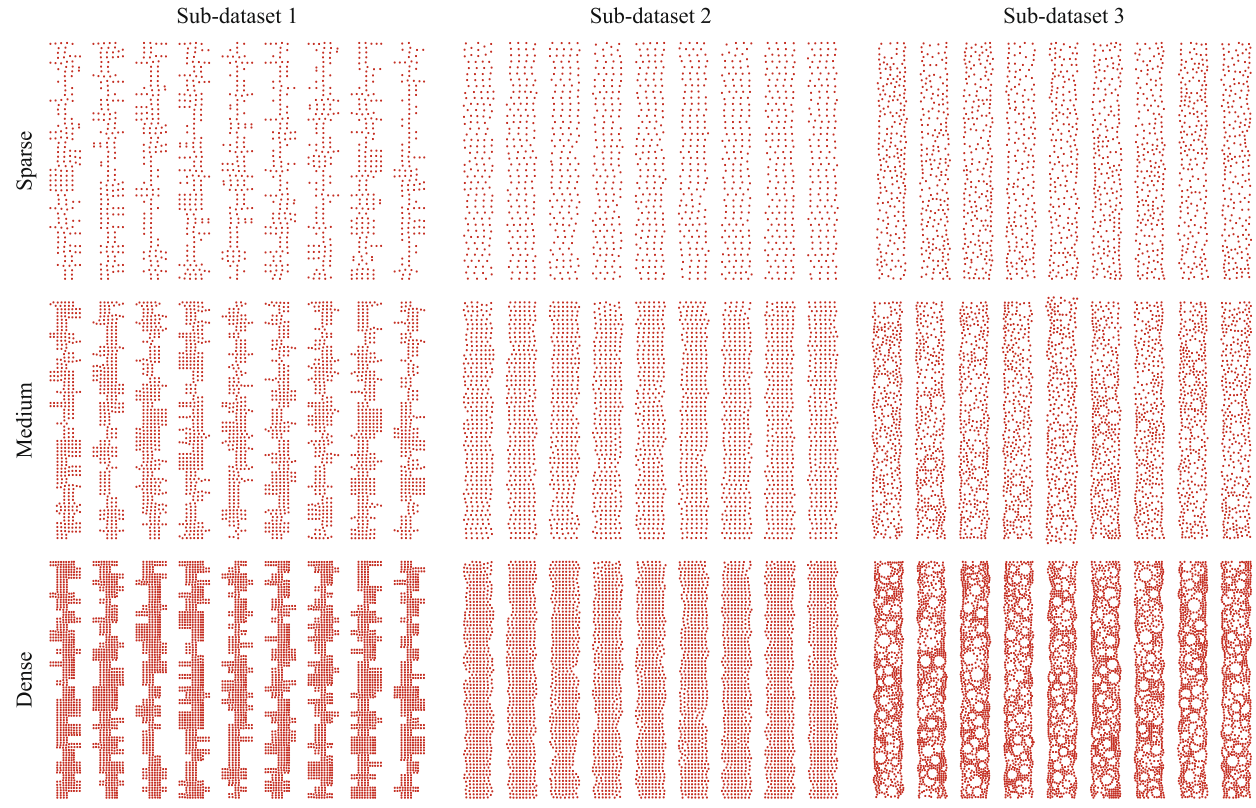

Fig. 8. Comparisons of node densities of the ABC columns obtained from our metamodeling pipeline for each sub-dataset. The structures in each row are ordered from sparse (top) to dense (bottom).

we controlled the number of superpixels during SLIC segmentation, and superpixels not associated with the structure (the black regions in Fig. 2) are discarded. Fig. 8 gives a visual comparison between the different nodal densities for each sub-dataset.

Appendix B. Visualization of ML architecture with different initializations

In this Appendix, we visualize our trained ML models with GNNexplainer [83]. In brief, GNNExplainer interprets the decisions of a GNN model by identifying the sub-graph G_S of the input graph that is the most important to the prediction of the output label. We note that in our case of graph classification, G_S does not have to be connected. For large graphs, brute-force testing of all possible sub-graphs is computationally intractable. The GNNExplainer algorithm works around this constraint by determining the probability that each edge is part of sub-graph G_S [83]. As such, the output of GNNExplainer is a weighted edge-mask of a given graph structure that determines the probability that each edge is a part of G_S for a given trained GNN model. In Fig. 9–11, we visualize the edge-

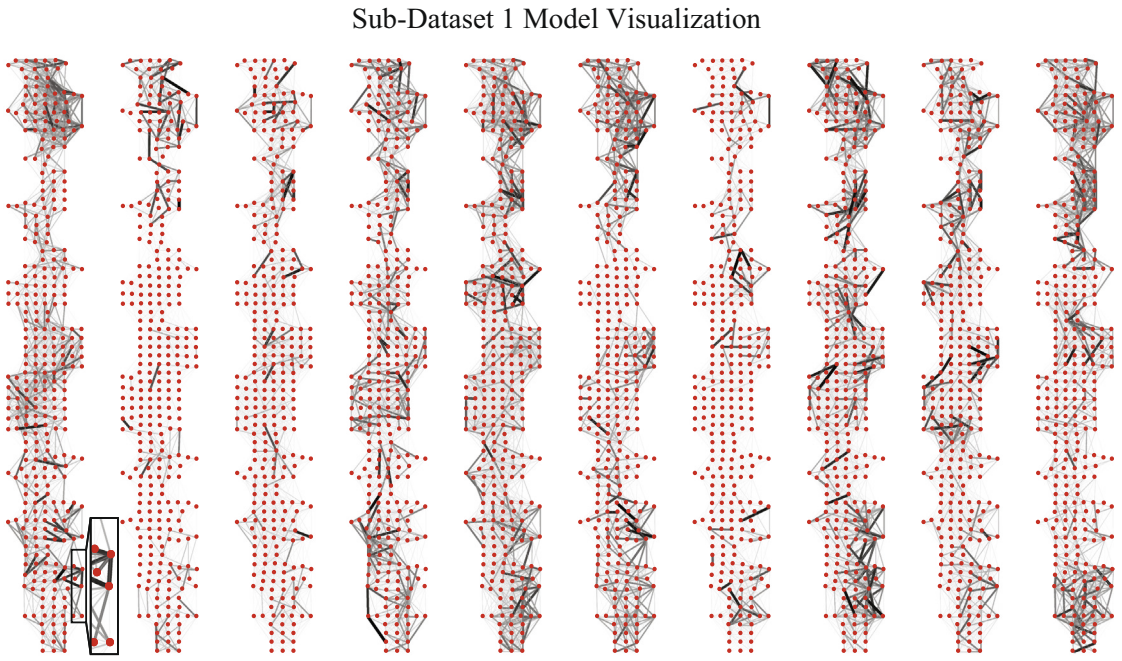

Fig. 9. Visualization of the 10 different initializations of the trained GNN model for sub-dataset 1 with medium node density. In this visualization, created with GNNExplainer [83], nodes are represented as red markers, all nodes within ball radius 40 are connected by edges, and edges that are likely to be influential to label predictions are darker and thicker (see zoom of the first structure).

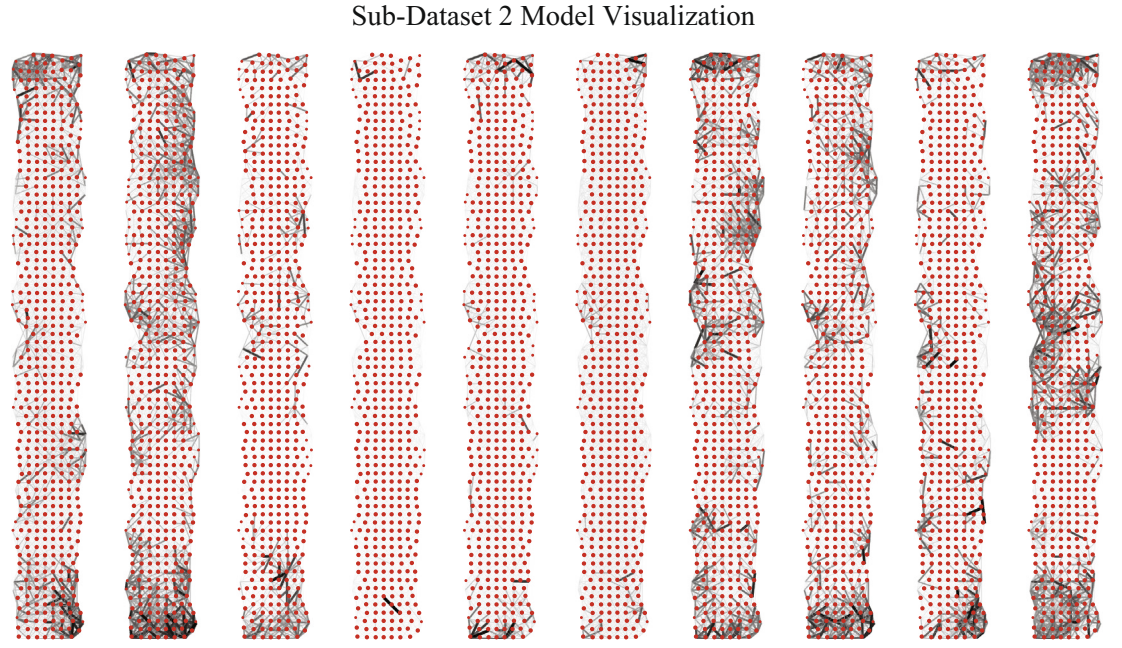

Fig. 10. Visualization of the 10 different initializations of the trained GNN model for sub-dataset 2 with dense nodal density. In this visualization, created with GNNExplainer [83], nodes are represented as red markers, all nodes within ball radius 30 are connected by edges, and edges that are likely to be influential to label predictions are dar.ker and thicker.

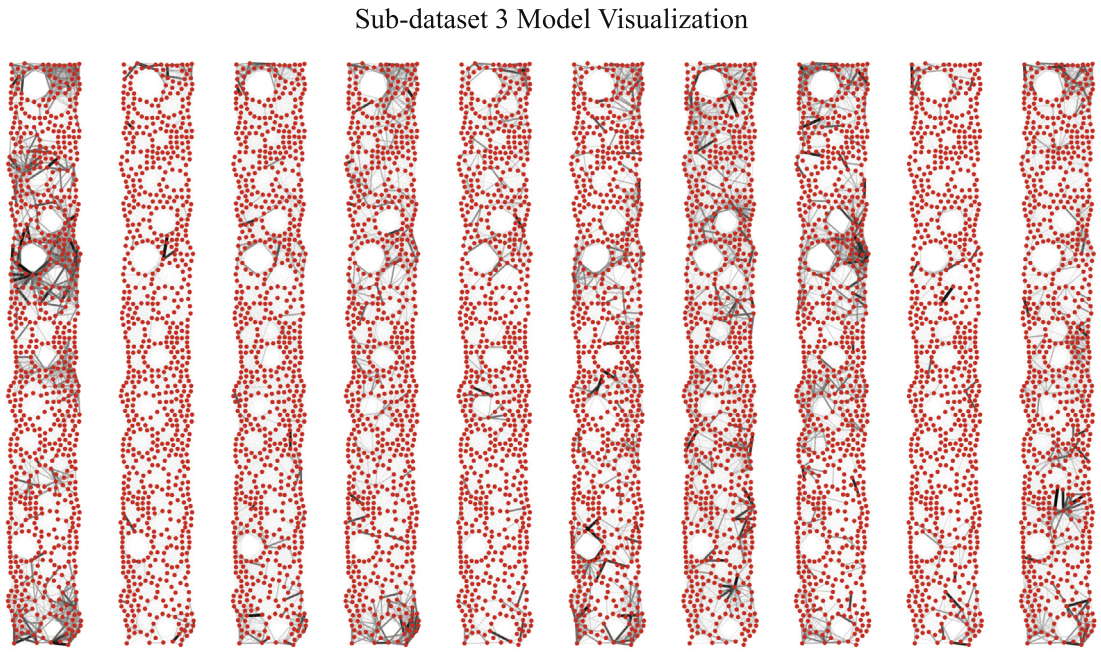

Fig. 11. Visualization of the 10 different initializations of the trained GNN model for sub-dataset 3 with dense nodal density. In this visualization, created with GNNExplainer [83], nodes are represented as red markers, all nodes within ball radius 30 are connected by edges, and edges that are likely to be influential to label predictions are darker and thicker.

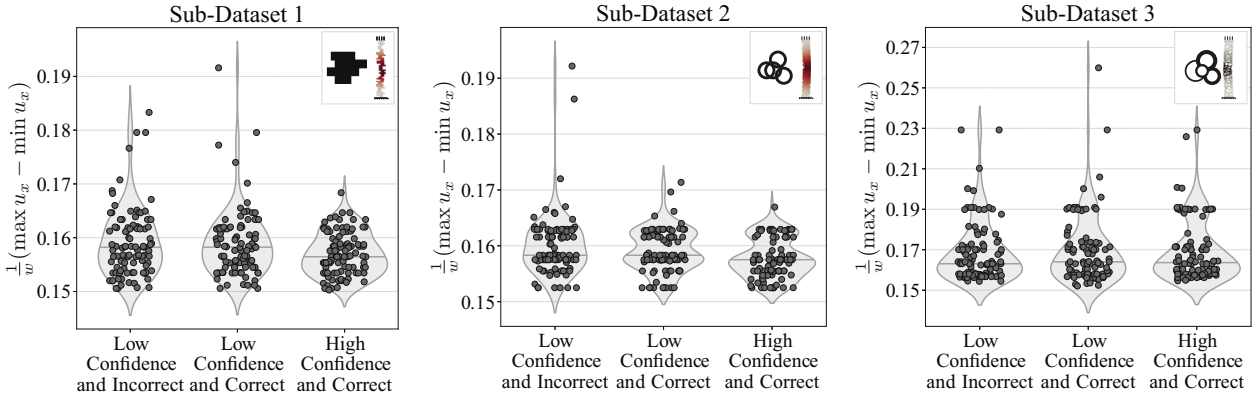

Fig. 12. Difference of maximum normalized x-displacements for each category, with 100 columns in each category. The proposed metric roughly measures the potential presence of higher order buckling modes, which is a source of uncertainty. We note that our stopping criterion for our simulations is defined as $\max|u_x| > 0.15w$.

masks for each of our trained ML models applied to identical samples. For each example, the underlying graph structure is identical (created from our metamodeling pipeline), and GNNExplainer is applied to the 10 different ML model initializations. Edges with a higher probability of being part of G_S are darker and thicker (see in particular the Fig. 9 zoom of the first structure). We note that while the model visualization highlights interesting features, in all cases, the features highlighted by GNNExplainer are not consistent across models. This observation leads us to believe that each ML model is learning different features, and therefore prompted us to investigate ensemble methods as a means to increase prediction accuracy (see Section 2.2.4). We also note that GNNExplainer outputs a feature mask that shows the importance of each node. Similar to the edge masks, we observed that each initialization led to a different importance for each nodal feature, further supporting the notion that the models from each initialization learn different features. On the other hand, since G_S depends on model initializations, it is not clear if any insights on mechanistically important features can be obtained through GNNExplainer directly.

Appendix C. Comparison between Deformed Profiles of High and Low Confidence Predictions

After confirming that our metamodel ensembles are well-calibrated, we investigate potential patterns across three cases: (1) low confidence with incorrect predictions ("Low Confidence and Incorrect"), (2) low confidence with correct predictions ("Low Confidence and Correct"), and (3) high confidence with correct predictions ("High Confidence and Correct"). To this end, we visualize randomly selected examples from each case in Fig. 13, we visualize overlaid column centerline displacements from each case in Fig. 14, and we visualize the range of x-displacement present in each column in Fig. 12. Across all examples, there is little

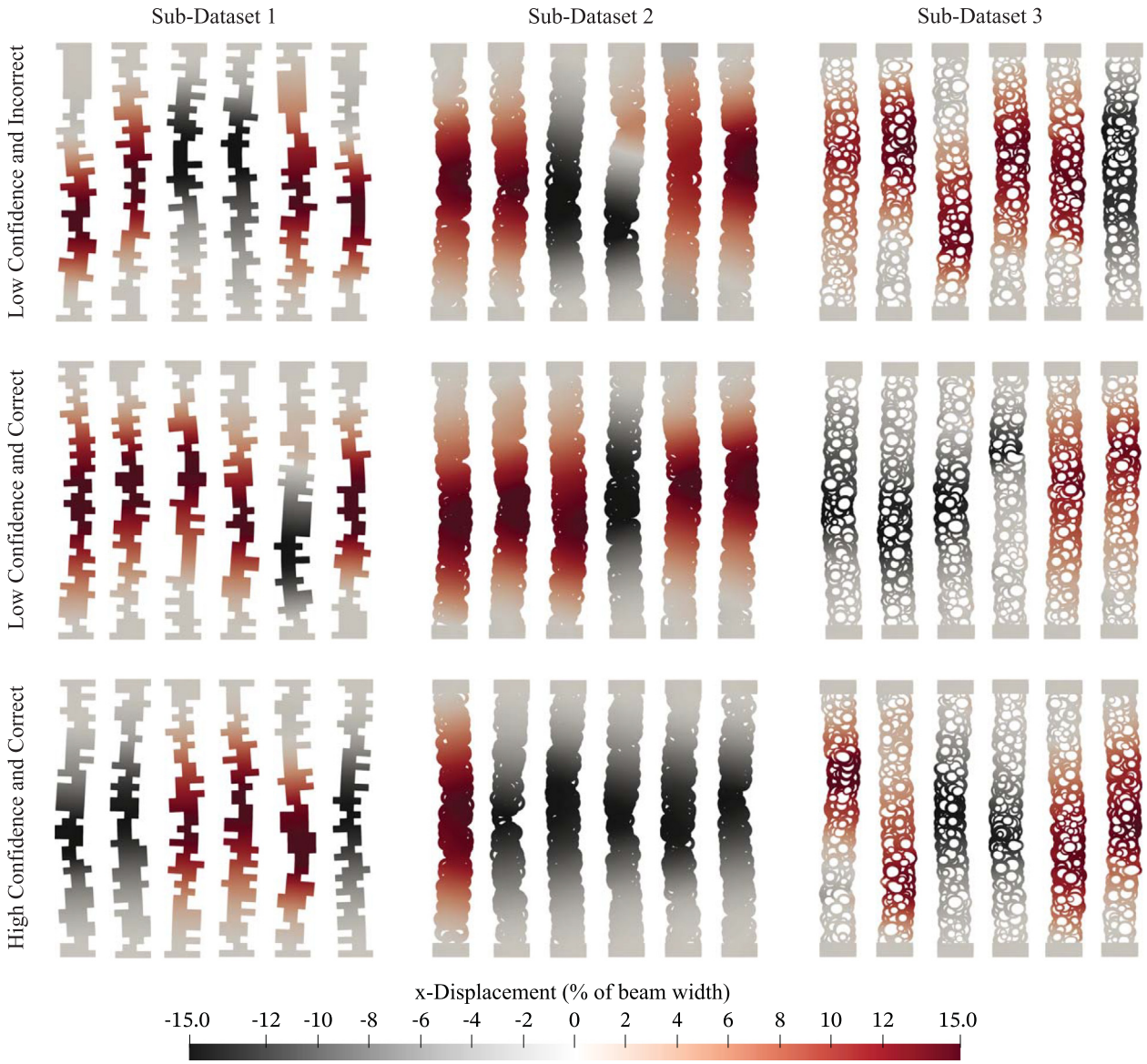

Fig. 13. Examples of deformed columns for high and low confidence predictions. Structures are arranged according to confidence levels (with low being 50% probability of being in either class and high being 100% in the correct class) and accuracy of prediction.

overt visible difference between cases. However, in sub-dataset 1 and sub-dataset 2, some columns that correspond to low confidence predictions exhibits signs of higher modes of buckling (e.g., centerline x-deflections cross the $y = 0$). However, this qualitative difference between cases is not observed in subdataset-3, perhaps due to the fact that local buckling of trimmed rings in subdataset-3 lead to more complex deformation profiles as shown in Fig. 13 and Fig. 14.

In Fig. 12, we plot the range of x-displacements for each column for 100 columns per group to further attempt to visualize a trend. Specifically, Fig. 12 shows the distribution of the difference in maximum and minimum x-displacement that is normalized by the column width $((1/w)(\max u_x - \min u_x))$. Note that the stopping criterion for our simulations is a maximum absolute x-displacement greater than 15% of the column width ($\max u_x > 0.15w$). Again, while there is some distinctions in the differences in deflection between high confidence and low confidence predictions in the case of sub-dataset 1 and sub-dataset 2, there is no real separation in the case of sub-dataset 3. This lack of clear distinction between the deformation profiles for each case might be due to the fact that uncertainty in model predictions comes from a combination of sources. First, the model performance plots shown in Fig. 3 clearly indicate that imperfect model performance is in part due to small training set size. Essentially, lack of coverage of the massive input parameter space is a source of model uncertainty. Second, as shown in Fig. 2a, each structure geometry is captured approximately rather than exactly, which could potentially contribute to uncertainty in prediction. Finally, and perhaps most interestingly, some of the qualitative results shown in Fig. 14 indicate that a subset of the columns in this dataset could be behaving a way that is "mechanically distinct" where the buckled column deforms in a manner that is more dissimilar from typical first mode buckling of a symmetrically loaded fixed–fixed column without geometric heterogeneity. We consider this third potential source of uncertainty a particularly interesting avenue for further study.

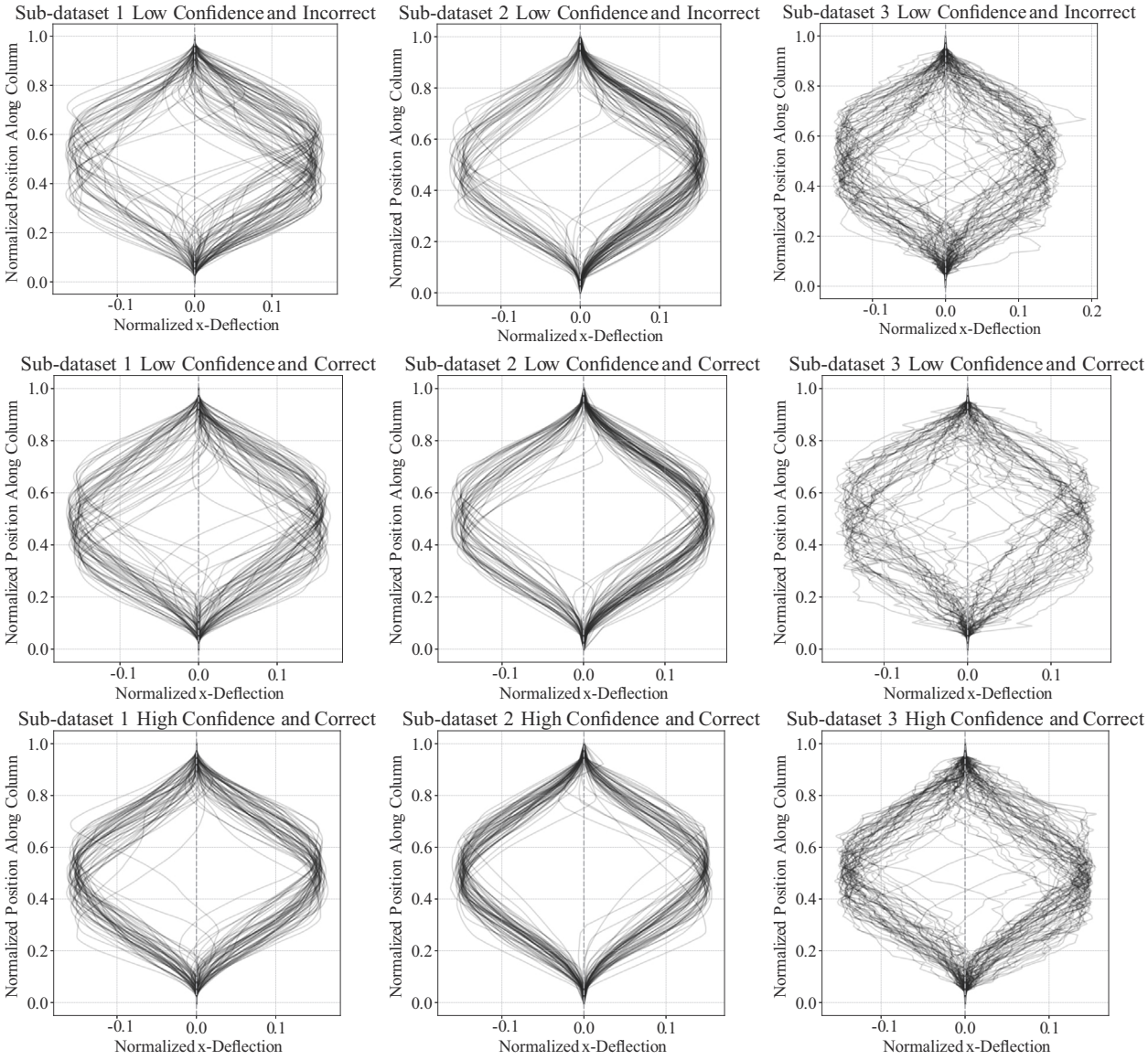

Fig. 14. Centerline displacements of 100 columns in each category of confidence and accuracy. Plots are arranged according to confidence levels (with low being 50% probability of being in either class and high being 100% in the correct class) and accuracy of prediction. Note that x-axis and y-axis are scaled differently for ease of visualization, and thus the illustrated deflections are not to scale.

References

[1] Achanta Radhakrishna, Shaji Appu, Smith Kevin, Lucchi Aurelien, Fua Pascal, Süsstrunk Sabine. Slic superpixels compared to state-of-the-art superpixel methods. IEEE Trans Pattern Anal Mach Intell 2012;34(11):2274–82.

[2] Ahrens James, Geveci Berk, Law Charles. Paraview: An end-user tool for large data visualization. Visual Handbook 2005;717(8).

[3] Alnæs Martin, Blechta Jan, Hake Johan, Johansson August, Kehlet Benjamin, Logg Anders, et al. The fenics project version 1.5. Arch Numer Software, 2015;3 (100)..

[4] Ayachit Utkarsh. The paraview guide: a parallel visualization application. Kitware Inc; 2015..

[5] Battaglia Peter W, Hamrick Jessica B, Bapst Victor, Sanchez-Gonzalez Alvaro, Zambaldi Vinicius, Malinowski Mateusz, et al. Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261; 2018..

[6] Bertoldi Katia, Reis Pedro M, Willshaw Stephen, Mullin Tom. Negative poisson's ratio behavior induced by an elastic instability. Adv Mater 2010;22 (3):361–6.

[7] Bertoldi Katia, Vitelli Vincenzo, Christensen Johan, Van Hecke Martin. Flexible mechanical metamaterials. Nat Rev Mater 2017;2(11):1–11.

[8] Bessa Miguel A, Glowacki Piotr, Houlder Michael. Bayesian machine learning in metamaterial design: Fragile becomes supercompressible. Adv Mater 2019;31 (48):1904845.

[9] Bogo Federica, Romero Javier, Loper Matthew, Black Michael J. Faust: Dataset and evaluation for 3d mesh registration. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2014; p. 3794–3801..

[10] Breiman Leo. Bagging predictors. Mach Learn 1996;24(2):123–40.

[11] Bronstein Michael M, Bruna Joan, LeCun Yann, Szlam Arthur, Vandergheynst Pierre. Geometric deep learning: going beyond euclidean data. IEEE Signal Process Magaz 2017;34(4):18–42.

[12] Chang Angel X., Funkhouser Thomas, Guibas Leonidas, Hanrahan Pat, Huang Qixing, Li Zimo, et al. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012; 2015..

[13] Ciregan Dan, Meier Ueli, Schmidhuber Jürgen. Multi-column deep neural networks for image classification. In: 2012 IEEE conference on computer vision and pattern recognition 2012; p. 3642–9. IEEE..

[14] Cranmer Miles, Sanchez-Gonzalez Alvaro, Battaglia Peter, Xu Rui, Cranmer Kyle, Spergel David, Ho Shirley. Discovering symbolic models from deep learning with inductive biases. arXiv preprint arXiv:2006.11287; 2020..

[15] Cranmer Miles D, Xu Rui, Battaglia Peter, Ho Shirley. Learning symbolic physics with graph networks. arXiv preprint arXiv:1909.05862; 2019..

[16] Duvenaud David, Maclaurin Dougal, Aguilera-Iparraguirre Jorge, Gómez-Bombarelli Rafael, Hirzel Timothy, Aspuru-Guzik Alán, Adams Ryan P. Convolutional networks on graphs for learning molecular fingerprints. arXiv preprint arXiv:1509.09292; 2015..

[17] Fey Matthias, Lenssen Jan E. Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds; 2019..

[18] Elia Forte Antonio, Hanakata Paul Z, Jin Lishuai, Zari Emilia, Zareei Ahmad, Fernandes Matheus C, et al. Inverse design of inflatable soft membranes through machine learning. Adv Funct Mater 2022:2111610.

[19] Ganaie MA, Hu Minghui, et al. Ensemble deep learning: A review. arXiv preprint arXiv:2104.02395; 2021..

[20] Geuzaine Christophe, Remacle Jean-François. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. Int J Numer Methods Eng 2009;79(11):1309–31.

[21] Gilmer Justin, Schoenholz Samuel S, Riley Patrick F, Vinyals Oriol, Dahl George E. Neural message passing for quantum chemistry. In: International conference on machine learning 2017; p. 1263–72. PMLR..

[22] Gómez-Bombarelli Rafael, Wei Jennifer N, Duvenaud David, Miguel Hernández-Lobato José, Sánchez-Lengeling Benjamín, Sheberla Dennis, et al.. Automatic chemical design using a data-driven continuous representation of molecules. ACS Cent Sci; 2018..

[23] Gongora Aldair E, Snapp Kelsey L, Whiting Emily, Riley Patrick, Reyes Kristofer G, Morgan Elise F, et al. Using simulation to accelerate autonomous experimentation: A case study using mechanics. Iscience 2021;24(4):102262.

[24] Gongora Aldair E, Xu Bowen, Perry Wyatt, Okoye Chika, Riley Patrick, Reyes Kristofer G, et al. A bayesian experimental autonomous researcher for mechanical design. Sci Adv 2020;6(15):eaaz1708.

[25] Gu Grace X, Chen Chun-Teh, Buehler Markus J. De novo composite design based on machine learning algorithm. Extreme Mech Lett 2018;18:19–28.

[26] Guo Chuan, Pleiss Geoff, Sun Yu, Weinberger Kilian Q. On calibration of modern neural networks. In: International Conference on Machine Learning, pages 1321–1330. PMLR; 2017..

[27] Guo Kai, Buehler Markus J. A semi-supervised approach to architected materials design using graph neural networks. Extreme Mech Lett 2020;41:101029.

[28] Hanakata Paul Z, Cubuk Ekin D, Campbell David K, Park Harold S. Forward and inverse design of kirigami via supervised autoencoder. Phys Rev Res 2020;2 (4):042006.

[29] He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2016;p. 770–8..

[30] Holmes Douglas P. Elasticity and stability of shape-shifting structures. Curr Opin Colloid Interface Sci 2019;40:118–37.

[31] Hornik Kurt, Stinchcombe Maxwell, White Halbert. Multilayer feedforward networks are universal approximators. Neural Networks 1989;2(5):359–66.

[32] Ioffe Sergey, Szegedy Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. PMLR; 2015. p. 448–456..

[33] Javili Ali, Dortdivanlioglu Berkin, Kuhl E, Linder Christian. Computational aspects of growth-induced instabilities through eigenvalue analysis. Comput Mech 2015;56(3):405–20.

[34] Khan Asifullah, Sohail Anabia, Zahoora Umme, Qureshi Aqsa Saeed. A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 2020;53(8):5455–516.

[35] Samir Khatir D, Boutchicha C Le, Thanh H Tran-Ngoc, Nguyen TN, Abdel-Wahab Magd. Improved ann technique combined with jaya algorithm for crack identification in plates using xiga and experimental analysis. Theoret Appl Fract Mech 2020;107:102554.

[36] Khatir Samir, Tiachacht Samir, Thanh Cuong Le, Ghandourah Emad, Mirjalili Seyedali, Abdel Wahab Magd. An improved artificial neural network using arithmetic optimization algorithm for damage assessment in fgm composite plates. Compos Struct 2021;273:114287.

[37] Kingma Diederik P, Ba Jimmy. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980; 2014..

[38] Kumar Siddhant, Tan Stephanie, Zheng Li, Kochmann Dennis M. Inverse-designed spinodoid metamaterials. Npj Comp Mat 2020.

[39] Lakshminarayanan Balaji, Pritzel Alexander, Blundell Charles. Simple and scalable predictive uncertainty estimation using deep ensembles. Adv Neural Inform Process Syst 2017;30.

[40] LeCun Yann, Bengio Yoshua, et al. Convolutional networks for images, speech, and time series. Handbook Brain Theory Neural Networks 1995;3361 (10):1995.

[41] Lejeune Emma. Mechanical mnist: A benchmark dataset for mechanical metamodels. Extreme Mech Lett 2020;36:100659.

[42] Lejeune Emma. Geometric stability classification: datasets, metamodels, and adversarial attacks. Comput Aided Des 2021;131:102948.

[43] Lejeune Emma, Javili Ali, Linder Christian. An algorithmic approach to multi-layer wrinkling. Extreme Mech Lett 2016;7:10–7.

[44] Lejeune Emma, Javili Ali, Linder Christian. Understanding geometric instabilities in thin films via a multi-layer model. Soft Matter 2016;12 (3):806–16.

[45] Lejeune Emma, Zhao Bill. Exploring the potential of transfer learning for metamodels of heterogeneous material deformation. J Mech Behav Biomed Mater 2021;117:104276.

[46] Leng Yue, Tac Vahidullah, Calve Sarah, Tepole Adrian B. Predicting the mechanical properties of biopolymer gels using neural networks trained on discrete fiber network data. Comput Methods Appl Mech Eng 2021;387:114160.

[47] Lew Andrew J, Buehler Markus J. Deepbuckle: Extracting physical behavior directly from empirical observation for a material agnostic approach to analyze and predict buckling. J Mech Phys Solids 2022:104909.

[48] Libonati Flavia, Gu Grace X, Qin Zhao, Vergani Laura, Buehler Markus J. Bone-inspired materials by design: toughness amplification observed using 3d printing and testing. Adv Eng Mater 2016;18(8):1354–63.

[49] Logg Anders, Mardal Kent-Andre, Wells Garth. Automated solution of differential equations by the finite element method: The FEniCS book,, volume 84. Springer Science & Business Media; 2012.

[50] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml, volume 30, page 3. Citeseer, 2013..

[51] Mao Yunwei, He Qi, Zhao Xuanhe. Designing complex architectured materials with generative adversarial networks. Sci Adv 2020;6(17):eaaz4169.

[52] Meza Lucas R, Zelhofer Alex J, Clarke Nigel, Mateos Arturo J, Kochmann Dennis M, et al. Resilient 3d hierarchical architected metamaterials. Proc Natl Acad Sci 2015;112(37):11502–7.

[53] Mohammadzadeh Saeed, Lejeune Emma. Predicting mechanically driven full-field quantities of interest with deep learning-based metamodels. Extreme Mech Lett 2021:101566.

[54] Monti Federico, Boscaini Davide, Masci Jonathan, Rodola Emanuele, Svoboda Jan, Bronstein Michael M. Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2017. p. 5115–24..

[55] Monti Federico, Bronstein Michael M., Bresson Xavier. Geometric matrix completion with recurrent multi-graph neural networks. arXiv preprint arXiv:1704.06803; 2017..

[56] Pakdaman Naeini Mahdi, Cooper Gregory, Hauskrecht Milos. Obtaining well calibrated probabilities using bayesian binning. In: Twenty-Ninth AAAI conference on artificial intelligence; 2015..

[57] Niculescu-Mizil Alexandru, Caruana Rich. Predicting good probabilities with supervised learning. In: Proceedings of the 22nd international conference on Machine learning 2005; p. 625–32..

[58] Overvelde Johannes TB, Shan Sicong, Bertoldi Katia. Compaction through buckling in 2d periodic, soft and porous structures: effect of pore shape. Adv Mater 2012;24(17):2337–42.

[59] Overvelde Johannes TB, Shan Sicong, Bertoldi Katia. Compaction through buckling in 2d periodic, soft and porous structures: effect of pore shape. Adv Mater 2012;24(17):2337–42.

[60] Paszke Adam, Gross Sam, Massa Francisco, Lerer Adam, Bradbury James, Chanan Gregory, et al. Pytorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, dAlché-Buc F, Fox E, Garnett R, editors, Advances in Neural Information Processing Systems 32, Curran Associates Inc, 2019; p. 8024–35..

[61] Pfaff Tobias, Fortunato Meire, Sanchez-Gonzalez Alvaro, Battaglia Peter W. Learning mesh-based simulation with graph networks. arXiv preprint arXiv:2010.03409; 2020..

[62] Polikar Robi. Ensemble learning. In: Ensemble machine learning. Springer; 2012. p. 1–34.

[63] Prachaseree Peerasait, Lejeune Emma. Asymmetric buckling columns (abc); 2022..

[64] Qi Charles R, Yi Li, Su Hao, Guibas Leonidas J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413; 2017..

[65] Raissi Maziar, Perdikaris Paris, Karniadakis George E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.

[66] Raney Jordan R, Lewis Jennifer A. Printing mesoscale architectures. Mrs Bull 2015;40(11):943–50.

[67] Reid Daniel R, Pashine Nidhi, Wozniak Justin M, Jaeger Heinrich M, Liu Andrea J, Nagel Sidney R, et al. Auxetic metamaterials from disordered networks. Proc Natl Acad Sci 2018;115(7):E1384–90.

[68] Sanchez-Gonzalez Alvaro, Godwin Jonathan, Pfaff Tobias, Ying Rex, Leskovec Jure, Battaglia Peter. Learning to simulate complex physics with graph networks. In: International conference on machine learning, PMLR 2020; p. 8459–68..

[69] Sanders Emily D, Aguiló Miguel A, Paulino Glaucio H. Multi-material continuum topology optimization with arbitrary volume and mass constraints. Comput Methods Appl Mech Eng 2018;340:798–823.

[70] Schapire Robert E. The strength of weak learnability. Mach Learn 1990;5 (2):197–227.

[71] Schlömer Nico, Cervone Antonio, McBain GD, Gokstorp Filip, Van Staden Ruben, Dokken Jørgen Schartum, et al. nschloe/pygmsh v7. 1.5. Zenodo; 2020..

[72] Schumacher Christian, Bickel Bernd, Rys Jan, Marschner Steve, Daraio Chiara, Gross Markus. Microstructures to control elasticity in 3d printing. ACM Trans Graph (TOG) 2015;34(4):1–13.

[73] Shin Dongil, Cupertino Andrea, de Jong Matthijs HJ, Steeneken Peter G, Bessa Miguel A, Norte Richard A. Spiderweb nanomechanical resonators via bayesian optimization: inspired by nature and guided by machine learning. Adv Mater, page 2106248..

[74] Tran-Ngoc H, Samir Khatir T, Le-Xuan G De, Roeck T Bui-Tien, Abdel Wahab M. A novel machine-learning based on the global search techniques using vectorized data for damage detection in structures. Int J Eng Sci 2020;157:103376.

[75] van der Walt Stéfan, Schönberger Johannes L, Nunez-Iglesias Juan, Boulogne François, Warner Joshua D, Yager Neil, et al. scikit-image: image processing in Python. PeerJ 2014;2:e453. 6.

[76] Vlassis Nikolaos N, Ma Ran, Sun WaiChing. Geometric deep learning for computational mechanics part i: Anisotropic hyperelasticity. Comput Methods Appl Mech Eng 2020;371:113299.
[77] Vyatskikh Andrey, Delalande Stéphane, Kudo Akira, Zhang Xuan, Portela Carlos M, Greer Julia R. Additive manufacturing of 3d nano-architected metals. Nat Commun 2018;9(1):1–8.
[78] Wang Yue, Sun Yongbin, Liu Ziwei, Sarma Sanjay E, Bronstein Michael M, Solomon Justin M. Dynamic graph cnn for learning on point clouds. Acm Trans Graph (tog) 2019;38(5):1–12.
[79] Douglas Brent West et al. Introduction to graph theory, volume 2. Prentice hall Upper Saddle River; 2001..
[80] Wolpert David H. Stacked generalization. Neural networks 1992;5(2):241–59.
[81] Yang Charles, Kim Youngsoo, Ryu Seunghwa, Gu Grace X. Prediction of composite microstructure stress-strain curves using convolutional neural networks. Mater Des 2020;189:108509.
[82] Yang Zhenze, Chi-Hua Yu, Buehler Markus J. Deep learning model to predict complex stress and strain fields in hierarchical composites. Sci Adv 2021;7 (15):eabd7416.
[83] Ying Rex, Bourgeois Dylan, You Jiaxuan, Zitnik Marinka, Leskovec Jure. Gnnexplainer: Generating explanations for graph neural networks. Adv Neural Inform Process Syst 2019;32:9240.
[84] You Jiaxuan, Liu Bowen, Ying Rex, Pande Vijay, Leskovec Jure. Graph convolutional policy network for goal-directed molecular graph generation. NeurIPS; 2018..
[85] Zhang Xiaoxuan, Garikipati Krishna. Bayesian neural networks for weak solution of pdes with uncertainty quantification. arXiv preprint arXiv:2101.04879, 2021..
[86] Zhou Jie, Cui Ganqu, Shengding Hu, Zhang Zhengyan, Yang Cheng, Liu Zhiyuan, Wang Lifeng, Li Changcheng, Sun Maosong. Graph neural networks: A review of methods and applications. AI Open 2020;1:57–81.
[87] Zhuang Xiaoying, Guo Hongwei, Alajlan Naif, Zhu Hehua, Rabczuk Timon. Deep autoencoder based energy method for the bending, vibration, and buckling analysis of kirchhoff plates with transfer learning. Eur J Mech-A/Solids 2021;87:104225.