A Graph-based probabilistic geometric deep learning framework
with online physics-based corrections to predict the criticality of
defects in porous materials

Vasilis Krokos1,*, Stéphane P. A. Bordas2, and Pierre Kerfriden3,1,*

1Cardiff School of Engineering, Cardiff University, UK
2Institute for Computational Engineering, Faculty of Science, Technology and
Communication, University of Luxembourg, Luxembourg
3Centre des Matériaux, Mines ParisTech, Evry, France
*Corresponding authors: Vasilis Krokos, KrokosV@cardiff.ac.uk; Pierre Kerfriden,
pierre.kerfriden@mines-paristech.fr

May 13, 2022

Abstract

Stress prediction in porous materials and structures is challenging due to the high computational cost associated with direct numerical simulations. Convolutional Neural Network (CNN) based architectures have recently been proposed as surrogates to approximate and extrapolate the solution of such multiscale simulations. These methodologies are usually limited to 2D problems due to the high computational cost of 3D voxel based CNNs. We propose a novel geometric learning approach based on a Graph Neural Network (GNN) that efficiently deals with three-dimensional problems by performing convolutions over 2D surfaces only. Following our previous developments using pixel-based CNN, we train the GNN to automatically add local fine-scale stress corrections to an inexpensively computed coarse stress prediction in the porous structure of interest. Our method is Bayesian and generates densities of stress fields, from which credible intervals may be extracted. As a second scientific contribution, we propose to improve the extrapolation ability of our network by deploying a strategy of online physics-based corrections. Specifically, we condition the posterior predictions of our probabilistic predictions to satisfy partial equilibrium at the microscale, at the inference stage. This is done using an Ensemble Kalman algorithm, to ensure tractability of the Bayesian conditioning operation. We show that this innovative methodology allows us to alleviate the effect of undesirable biases observed in the outputs of the uncorrected GNN, and improves the accuracy of the predictions in general.

Keywords: Multiscale stress analysis, Graph Neural Network, Surrogate modelling, Bayesian machine learning, Physics-corrected neural network

1 Introduction

Multiscale computational analysis is very common in various engineering areas such as geo-engineering (heterogeneous soil layers and subterranean cavities), bio-mechanical engineering (bones and soft tissues) and mechanical engineering (composite and architectured materials). Finite Element Analysis (FEA) in these structures is notoriously challenging since the FE mesh required to fully resolve the finer scale features has to be very dense resulting into intractable computational problems.
Common approaches to tackle multiscale problems usually fall either in the homogenisation or concurrent scale coupling approaches. In both cases the multiscale problem is split into a macroscale and a microscale problem. The macroscale problem diffuse the overall stress field in the entire structure, and the microscale computations are performed to correct the macroscale fields. In homogenisation it is assumed that the gradients of the macroscale displacement are homogeneous over the entire material sample (i.e. the representative volume element, RVE) [Évariste Sanchez-Palencia, 1987; Zohdi and Wriggers, 2005]. If that assumption does not hold, for instance due to fast macroscale gradients induced by sharp macroscale features, then concurrent scale coupling methods are employed that are usually more expensive and intrusive compared to RVE based methods. In these techniques the results of homogenisation are applied to the boundary of regions of interest for concurrent microscale corrections to be performed [Raghavan and Ghosh, 2004; Oden et al., 2006; Kerfriden et al., 2009; Hesthaven et al., 2015; Paladim et al., 2016; Krokos et al., 2021] which may be performed in a purely downward approach [Oden et al., 1999], or with 2-way coupling [Gendre et al., 2009].

In this work we propose a Neural Network (NN) based concurrent scale coupling method to tackle 3D multiscale problems and specifically we focus on porous media with no separation of scales. Neural Networks (NNs) have been successfully employed in solving engineering problems by reproducing the output of simulation codes in a variety of fields like fluid dynamics [Lye et al., 2020; Raissi et al., 2020], solid mechanics [Nie et al., 2019; Jiang et al., 2020; Deshpande et al., 2021] and computational biology [Senior et al., 2019, 2020]. Most of the publications that focus on learning the response of FE models employ Convolutional Neural Networks (CNNs), that are very efficient in working with images [Nie et al., 2019; Mendizabal et al., 2020; Sun et al., 2020].

Unfortunately, most of the aforementioned works deal with 2D problems. One of the reasons, in our opinion, is that 3D CNNs are very expensive and thus the literature on the subject is limited [Mendizabal et al., 2020; Rao and Liu, 2020] and mostly focused on segmentation [Buda et al., 2019; Lu et al., 2019] and classification tasks [Khosla et al., 2018; Vu et al., 2020]. An important reason for this is that representing arbitrary geometries with voxels is highly inefficient because the resolution of the image has to be constant while the geometry of the object can be better described with a variable resolution. Specifically, for multiscale problems, this means that the voxel size needs to be very small to capture the microscale features. This results in a very computationally wasteful data representation that uses a large number of voxels to represent areas with no microscale features that could be represented with only a few voxels. This is also true for CNNs applied to 2D problems, but the computational cost is disproportionately larger in 3D problems. We will follow a different approach, that of geometric learning.

Another problem with current deep learning models is their poor generalisation in cases where the training dataset is small or different from the cases we want to predict. The problem is more prominent when the computational model is very expensive and cannot be evaluated multiple times or when no computational model is in place and experiments need to be performed. Consequently, it is difficult to create a representative and large enough training dataset to get reasonable predictions.

Our paper addresses these two points by extending our previous work [Krokos et al., 2021], while adding two novel algorithmic elements. First, we manage to tackle 3D problems by using geometric learning where instead of voxelised images we learn directly from the mesh that is used to perform the FE simulations. Second, we use physics based online corrections to improve the generalisation of the NN without the need of creating more data.

Geometric learning can be seen as an extension of previously established NN techniques from Euclidean, grid structures to non-Euclidean, graph and manifold structures. The NNs that perform geometric learning are called Graph Neural Networks (GNNs). GNNs can overcome the constant voxel resolution problem by operating directly on the FE mesh used to solve the FE problem. The FE mesh can have a large number of triangles or tetrahedra in areas where the geometry is complex or large gradients are expected during the FEA and a small number of triangles or tetrahedra in the rest of the cases. This adaptive resolution strategy allows for a very efficient use of computational resources. Additionally, because we are primarily interested in predicting the maximum stress, GNNs allow us to further reduce the computational cost. Specifically, we show that the maximum stress, in the cases we
examine, is present on the surface and not the volume of the mesh [3.2] and thus we choose to extract and work only on the surface mesh which greatly reduces the training time.

We use Ensemble Kalman method [Evensen, 1994] as the online correction technique. The difference with respect to the original Kalman method is that the covariance matrix is replaced by the sample covariance. This reduces the computational cost and makes this variant suitable for problems with a large number of variables such as in our case where the number of variables is equal to the number of nodes on the FE mesh used to solve the PDE. In order to use the Ensemble Kalman method we need to convert the GNN into a Bayesian GNN (BGNN). In this work we use the Bayes by Backprop (BBB) method [Blundell et al., 2015] to quantify the uncertainty of the GNN and convert it into a BGNN. The online correction step takes as input the BGNN output, prior distribution, and performs an update step using Bayes’ Theorem, which produces the updated distribution, posterior, that satisfies the desirable physical law. This technique allows us to improve the BGNN prediction by taking advantage of the physics in place without creating new data or retraining the BGNN.

BGNNs are a very useful tool even outside the online correction framework, since they can provide credible intervals (CIs) along with a mean prediction. When the BGNN makes a prediction for a case that is different from the cases in the training dataset the CIs will be broad reflecting the fact that the BGNN is uncertain for the prediction. In case where the CIs are very tight this implies that the BGNN is certain for the prediction. The BGNN not only warns the user for the potential uncertainty in the prediction but it also attempts to squeeze the correct value between an upper and a lower value.

Our paper is organised as follows. In section [2] we present the reference multiscale mechanical model that we aim to solve online, and we introduce useful definitions for our methodology. In section [3] we discuss the GNN architecture and the input and output features. Moreover, in section [4] we explain the strategy that we follow to convert the deterministic GNN to a BGNN. Furthermore, in section [4.2] we introduce the online correction method we use. Finally, in section [5] we apply our multiscale GNN method to two different problems, we demonstrate the effectiveness of the GNN on accurately predicting the maximum stress values and the stress distributions and also we demonstrate the online stress correction technique that we use to improve the GNN prediction in common cases where NNs fail namely small training dataset and extrapolation.

2 Deep-learning-based meta-modelling for multiscale finite element analysis

2.1 Problem statement: linear elasticity

We consider a 3D body occupying domain $\Omega \in \mathbb{R}^3$ with a boundary $\partial \Omega = S$. The body is subjected to prescribed displacements $U_D$ on its boundary $\partial \Omega_u$ and prescribed tractions $T_D$ on the complementary boundary $\partial \Omega_T = \partial \Omega \setminus \partial \Omega_u$. We consider a body force $f$ and a displacement $u : \Omega \to \mathbb{R}^3$. For an initial configuration $x$, the boundary value problem of linear elasticity consists in finding $u(x) = \arg \min_{u^*} E_p(u^*)$ where the potential energy is defined as

$$E_p(u) = \int_{\Omega} W(\epsilon) \, d\Omega - \int_{\Omega} f \cdot u \, d\Omega + \int_{\partial \Omega_T} T_d \cdot u \, dS$$ (1)

We consider a linear Saint-Venant–Kirchhoff material model defined by its strain energy density

$$W(\epsilon) = \frac{1}{2} \lambda |tr(\epsilon)|^2 + \mu tr(\epsilon^2)$$ (2)

In the equations above, $\lambda$ and $\mu$ are the Lamé elasticity parameters and $\epsilon$ the linearised strain tensor defined as

$$\epsilon = \frac{1}{2} (\nabla u + (\nabla u)^\top)$$ (3)
The Cauchy stress tensor $\sigma$ is calculated as follows:

$$\sigma = \frac{\partial W(\epsilon)}{\partial \epsilon}$$  \hspace{1cm} (4)

2.2 Case study: mechanical specimen made of a porous material

For the purpose of training the GNN we need to create multiple realisations of a multiscale geometry with both macroscale and microscale features. In this work we consider a structure that we will refer to from now on as “dogbone” [Fig 1]. The dogbone specimen has a cylindrical hole as a macroscale feature. The material is porous, made of randomly distributed spherical pores as microscale features.

![Figure 1](image)

Figure 1: In the top subfigure, we can see a realisation of the dogbone structure. The dogbone is porous and it has a cylindrical hole in the middle. The porous phase is geometrically defined as the union of randomly distributed spheres. The spheres are allowed to intersect with each other, the boundaries of the dogbone and the cylindrical hole. In the bottom subfigure, we see the mesh of the dogbone. We observe that the mesh is denser in the middle of the structure, where the porous phase is present, and coarser everywhere else.

2.3 Mesh and Finite element solver

The FE meshes are created using the open source 3D finite element mesh generator gmsh [Geuzaine and Remacle, 2009], using pygmsh [Schlömer, 2021] as a python interface. Unstructured tetrahedral meshes will be used.

The FE problems, comprising the macroscopic surrogates models and the full fine-scale models (as defined in the next subsection [2.4]), will be solved using the FEniCS software which is a python based popular open-source computing platform for solving partial differential equations (PDEs) [Alnaes et al., 2015; Logg et al., 2012]. We use standard linear tetrahedral elements to solve the elasticity problem.
Meshes are manipulated using the python package pyvista [Sullivan and Kaszynski, 2019] that allows for easy interaction with VTK objects through Numpy arrays.

2.4 Global-local framework

The multiscale framework that we use here is a direct extension of our previous work [Krokos et al., 2021]. As mentioned in section [2.2], the examples that we show in this work are dedicated to linear elasticity for porous media made of a homogeneous matrix with a random distribution of microscale features that are modelled as spherical voids.

Firstly, we use a coarse finite element mesh that ignores the existence of the microscale features, to obtain the macroscale solution, with a small computational cost, over the entire domain Ω. Then, we assume there exists a function $F$ that takes as input the geometry of the microscale features and the local macroscale solution in a window $B \subset \Omega$, which we call patch, and outputs the microscale stress field in a sub-region of $B$ namely $\hat{B} \subset B$ that we call Region of Interest (ROI) [Fig 2].

The function $F$ will be approximated through a GNN. The existence of such function is not proved mathematically but it is supported by the Saint-Venant’s principle, our previous work [Krokos et al., 2021] and will be shown to be valid via numerical experiments in the context of the present paper, which extends previous findings to the three-dimensional setting.

2.5 Quantity of interest and accuracy measure

We are interested in predicting a mechanical quantity that indicates potential crack initiation sites. In this work we choose the Von Mises stress (VM) for this purpose, which can be calculated from the following formula

$$\sigma_{VM} = \sqrt{\frac{1}{2}[(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{xx} - \sigma_{zz})^2 + (\sigma_{yy} - \sigma_{zz})^2] + 3(\sigma^2_{xy} + \sigma^2_{xz} + \sigma^2_{yz})}$$

(5)

Additionally, we will define accuracy in a way that serves the purpose of accurately predicting the maximum Von Mises stress in the patches. We consider that the maximum Von Mises stress is accurately calculated in the ROI of the patch if it is predicted with a relative error less than the threshold of 10%, unless stated otherwise. This procedure is summarised in [Appendix A].

3 Graph-based geometric deep learning

In this section we briefly review geometric learning methods reported in literature. Also, we explain the geometric learning framework that we choose to use and finally we discuss the structure of the GNN along with the input and output features.

3.1 Brief review of earlier work on geometric deep learning

GNNs have been vastly developed over the last years with various works on segmentation [Hanocka et al., 2019; Schult et al., 2020; Lei et al., 2021] and shape correspondence or retrieval tasks [Masci et al., 2018; Gong et al., 2020]. One early attempt of deep learning in Non-Euclidean spaces is introduced by [Qi et al., 2017] where a method for geometric learning using point clouds is described. Even though point clouds is a very simple way to describe 3D objects, a more common and informative way is by using a mesh. One common approach for geometric learning with meshes is described in [Hanocka et al., 2019] where a generic framework for processing 3D models for classification and segmentation tasks is described. MeshCNN operates directly on the mesh and not a voxel based representation of the object, which leads to increased computational cost, or a point cloud, which results in loss of valuable topology information.

GNNs have also been used for various engineering applications. [Vlassis et al., 2020] used geometric deep learning to model anisotropic hyperelastic materials. Specifically they modelled polycrystals using
Figure 2: Porous material, $\Omega$, with patches, $B$. We can observe a dogbone structure where we have extracted 2 patches. With green we can see the ROI of the patches, $\tilde{B}$.
a graph where the nodes represent the monocrystals and the edges their connectivity. This utilization of non-Euclidean data allowed them to take advantage of the rich micro structure information not described by classical Euclidean descriptors like density and porosity. Additionally, [Sanchez-Gonzalez et al., 2020] used GNNs to simulate particle based dynamic systems where 1 or more types of particles, ranging from water to sand, are involved. The physical state of the system is described by a graph of interacting particles and the physics is predicted by message passing through the nodes (particles) of this graph. Later, [Pfaff et al., 2021] used GNNs for mesh based dynamic simulations. In contrast to particle based problems, in this work the authors took advantage of the connectivity of the mesh to define the edges of the graph. They demonstrated their method in a variety of applications such as deformable bodies, fluid flow around obstacle and air flow around an aircraft wing. Specifically for the air flow around an aircraft wing they show that GNNs have superior performance compared to CNNs. Recently, [Mylonas et al., 2022] used a Bayesian GNN to infer the position and shape of an unknown crack via patterns of dynamic strain field measurements at discrete locations. Lastly, [Lino et al., 2021] developed a multiscale GNN that efficiently diffuses information across different scales making it ideal for tackling strongly non local problems such as advection and incompressible fluid dynamics.

Most of these methods are using the formulation described in [Battaglia et al., 2018] which provides a general framework to work with GNNs. We use the same formulation since it allows for a very natural encoding of information on the mesh where absolute information, for instance material properties, can be encoded on the vertices of the mesh while relative information, for example distance between vertices, is encoded on the edges of the mesh.

### 3.2 Geometric learning for multiscale stress analysis: assumptions and justification

As can be seen from the literature review, researchers can take advantage of the flexibility that GNNs offer, to reduce the computational cost by modelling their system in more meaningful ways that stem from the physical understanding of the underlying problem, as for example [Vlassis et al., 2020] who modelled the behaviour of a polycrystal by considering every crystal as a node of the graph instead of operating on the volume mesh. Following this paradigm, for reasons we explain in the next paragraphs, we choose to work only on the surface mesh of the porous medium and not the volume mesh.

First of all, we claim that the maximum stress in the cases we examine is located at the surface of the structure, not in the bulk of it. In order to test this hypothesis we conduct an experiment where we perform 100 FE simulations with 100 realisations of the dogbone structure. For the boundary conditions we apply a displacement along the $x$ and $-x$ direction of the same magnitude but with opposite direction. Afterwards, we extract from the volume mesh the surface mesh with the stress tensors encoded on the nodes of the surface mesh [Appendix B]. Lastly, we calculate the maximum Von Mises stress both on the volume and surface mesh and we compare the two. From [Fig 3] we can conclude that indeed the maximum volume stress is the same as the maximum surface stress.

Additionally, we hypothesise that the surface mesh is a comprehensive representation of the geometry of the specimen. Lastly, we assume that the trace of the, smooth, macroscale stress over the skin mesh is sufficient to inform the GNN of the macroscale stress state over the patch. We prove these 2 hypotheses numerically in the context of the current paper.

Consequently, we choose to work only on the surface mesh since we are primarily interested in predicting the maximum stress. This greatly reduces the memory requirements and the training time.

### 3.3 Graph construction

In our attempt to perform geometric learning on the surface mesh of a dogbone structure we consider a graph $G$ that can be described by a set of nodes $V : \{V_1, V_2, ... V_N\}$ where $N$ is the number of nodes of $G$ and a set of edges $E : \{E_1, E_2, ... E_M\}$ where $M$ is the number of edges of $G$.

The nodes of the graph $V$ coincide with the nodes of the surface mesh, since their position is optimised by the mesh generation software to best describe the geometry of the structure. The edges
Figure 3: The x axis corresponds to the maximum Von Mises stress on the volume mesh and the y axis to the maximum Von Mises stress on the surface mesh for each of the 100 performed simulations. The red line is the $y = x$ line. We can observe that all the points lay on this line and thus the maximum volume and surface stress values are the same.

of the graph are used to pass messages between the nodes and they define the connectivity of the graph. Consequently, the edges of the graph cannot coincide with the edges of the surface mesh since the surface mesh has disconnected areas. This means that it is possible for areas of the surface mesh that are close to each other to not share an edge, for example two interacting but non intersecting defects. This does not allow passing of information between the two and thus their interaction cannot be modelled. To overcome this problem we define 2 different neighbourhoods [Fig 4].

- Geodesic: The Geodesic, or 1-ring neighbourhood, includes the nodes that share an edge with the central node. This is equivalent to the neighbourhood dictated by the surface mesh connectivity.

- Euclidean: The Euclidean neighbourhood in this work is defined using the radius neighbours. Every node $V_i$, is connected with every node that falls into a sphere with a predefined radius $r$ and with centre $V_i$. The predefined radius $r$ that is used for this work is 4 times the radius of the spherical voids as suggested in [Krokos et al., 2021]. In practice that results into a very high number of neighbours and thus into a very expensive computational problem. To tackle this problem we need to restrict the number of neighbours for each node by setting a threshold and randomly choosing neighbours so that the total number of neighbours is below the threshold. In section [D.5] we discuss the most appropriate value for this threshold.

In this work we use either Euclidean or a combination of Euclidean and Geodesic neighbours, as proposed by [Schult et al., 2020], both of which allow the modeling of interactions between non-intersecting parts of the mesh.

The edges of the graph are bidirectional so that the information can be exchanged both ways between two connected nodes. This means that for every edge $E_{k,l}$ passing a message from node $V_k$ to $V_l$ there exists the edge $E_{l,k}$ passing a message from node $V_l$ to $V_k$.

### 3.4 Input and Output features

The framework we use requires the input and output graphs to be of the same structure. Consequently, we choose to interpolate the 3D FE solution from the macroscale mesh, i.e. mesh without spherical
voids, to the microscale mesh, i.e. mesh with the spherical voids.

For the input of the GNN we encode node features, $v_i$ where $i = 1, N$, on the nodes of the graph and edge features, $e_{i,j}$ where $j = 1, M$, on the edges of the graph. The input node features are the independent components of the macroscale stress tensor along with the microscale feature indicator, a single integer indicating if the node corresponds to a microscale feature or not. The input edge features are the relative position between the 2 nodes that the edge is connecting along with the distance between the 2 nodes. This is summarised in [Fig 5].

The output of the graph contains information only on the nodes of the mesh. Specifically, the output node features are the components of the microscale stress tensor.

We sum up in [Fig 6] the input and output of the GNN. We have only included the node features since the edge features are calculated as a pre-processing step from the connectivity of the mesh.

### 3.5 GN block

The GN block is the basic building block of the GNN and it is used to map an input graph to an output graph with the same nodes and edges but with updated node and edge features. The node and edge features are jointly passed through an edge update Multilayer Perceptron (MLP) and a node
update MLP in a 2-step procedure described in [Battaglia et al., 2018] that sequentially updates the edge features first and the node features later. A sketch of this procedure can be found in [Fig 7].

For the edge update step, if we consider an edge $E_{k,l}$ with edge features $e_{kl}$ that sends a message from the sender node $V_k$ with node features $v_k$ to the receiver node $V_l$ with node features $v_l$ then the updated edge feature, $e_{kl}^*$, is calculated by passing through the edge MLP the concatenation of these features, $e_{kl}^* = \text{MLP}(\text{concatenate}(e_{kl}, v_k, v_l))$. If $M$ is the number of edges of the graph, $NNF$ is the number of node features and $NEF$ is the number of edge features then the input of the edge update MLP is of size $[M \times (2NNF + NEF)]$.

For the node update step, the updated edge features, $e^*$, are aggregated per node. In this work we use mean aggregation although elementwise summation, maximum or any other symmetric operation can be used. After the aggregation step, we have a single edge feature, $e^{**}$, corresponding to each node feature $v$. The updated node features, $v^*$, are calculated by passing through the node MLP the concatenation of the original node features with the updated and aggregated edge features corresponding to this node, $v^* = \text{MLP}(\text{concatenate}(v, e^{**}))$. If $N$ is the number of nodes of the graph, $NNF$ is the number of node features and $F$ is the dimensionality of the updated edge features then the input of the node update MLP is of size $[N \times (NNF + F)]$.

3.6 Model

In this work we deploy an Encode-Process-Decode GNN as described by [Battaglia et al., 2018]. The encoder, encodes the node and edge features in a latent space of higher dimensions. The processor updates these node and edge features and finally the decoder, decodes these features from the latent space to the output space.

- **Encoder**: The node and edge features are independently encoded into a latent space of 128 dimensions using 2 distinct MLPs with the same structure. The MLP has 3 hidden layers followed by a Dropout layer [Hinton et al., 2012] and ReLU activation function each and a Layer Normalization layer after the output layer, that improves training stability and decreases training time in recurrent NNs [Ba et al., 2016].

- **Processor**: The processor is composed of 5 residual GN Blocks. By using residual blocks the NN itself can choose its depth by skipping the training of a few layers using skip connections. This is currently the most common way to train deep NNs [He et al., 2015; Kim et al., 2016; Zagoruyko
Figure 6: In this figure we can see the input and output of the GNN. The input consists of the 6 stress components of the macroscale stress tensor along with the micro indicator, a single integer per node that determines if the node belongs to a microscale or a macroscale geometrical feature. For the patch corresponding to the micro indicator we the FE mesh is visible. The output of the GNN is the 6 stress components of the microscale stress tensor.

and Komodakis, 2017; Lim et al., 2017] and it also applies to GNNs [Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Mylonas et al., 2022]. The structure of the MLPs of the Processor is the same as the structure of the encoder MLPs. The processor combines information from edges and nodes and passes messages between nodes that share an edge.

- **Decoder**: The decoder operates only on the node features as the output of the GNN is the microscale stress field on the nodes. A single MLP is used to decode the node features from the latent space to the output space. The decoder MLP is similar to the MLPs used in the encoder and the processor but there is no Layer Normalization layer applied after the output layer.

There is a skip connection that connects the macroscale stress tensor, from the input, with the output of the Decoder. That way the network can learn how the microscale stress field diverges from the macroscale stress field. In section [D.1] we demonstrate that this leads to smoother and faster convergence. A sketch of the GNN can be found below [Fig 8].

11
4 Stress correction using homogeneous Neuman conditions: an online Kalman-based approach

In this section we introduce an online hierarchical Bayesian method to constrain the prediction of the GNN. This method allows us to improve the network prediction without creating more data and retraining the network, which is computationally expensive and requires access to the function that creates the training data. We impose homogeneous Neumann conditions, that cannot be explicitly imposed while training the network, even though the training examples satisfy them up to the FE accuracy.

To apply this method we need a distribution of outputs from the GNN. To achieve that, we convert the deterministic GNN to a Bayesian GNN (BGNN). The posterior estimation of the BGNN for the stress field over the surface of the structure becomes a prior for the online assimilation step. Physical constraints are applied statistically by introducing them as partial information of the stress field, yielding a Bayesian posterior update. We chose to resort to a Gaussian approximation of the posterior in the form of the ensemble Kalman method. Alternative methods include Langevin MCMC and Variational Inference, however the online cost may prove to be prohibitive.
4.1 Bayesian GNN

Several techniques have been proposed to quantify the uncertainty in NNs resulting in Bayesian NNs [Hinton and van Camp, 1993; Graves, 2011; Kingma and Welling, 2014; Blundell et al., 2015]. In this work we use the Bayes by Backprop method (BBB) to convert the deterministic GNN to a BGNN [Blundell et al., 2015].

In order to define the optimisation objective, we consider a probabilistic NN where we replace the constant weights of a deterministic NN with a distribution over each weight. Before observing the data, $D$, the weight distribution is called prior, $P(w)$, and after observing the data the weight distribution is updated, so that the BNN will fit the data, and called posterior $P(w|D)$. For an input $x \in \mathbb{R}^n$ the output of the probabilistic NN will be a probability distribution over all possible outputs $y \in \mathbb{R}^k$. Minimising the loss function results in determining the posterior distribution of the weights given the training data, $P(w|D)$. The posterior cannot be computed analytically through the Bayes’ rule, $P(w|D) = \frac{P(D|w)P(w)}{P(D)}$, because of the existence of intractable integrals. A common approach is to perform Variational Inference (VI) to approximate the posterior with a variational distribution $q(w|\theta)$ [Hinton and van Camp, 1993; Graves, 2011], parameterised by $\theta$. The new objective is to find the parameters $\theta_{\text{opt}}$ [Eq. 6] that minimise the Kullback-Leibler (KL) divergence between the approximate posterior and the true Bayesian posterior.

$$
\theta_{\text{opt}} = \arg \min_{\theta} \text{KL}[q(w|\theta)||P(w|D)]
= \arg \min_{\theta} \int q(w|\theta) \log \frac{q(w|\theta)}{P(w)P(D|w)} \, dw
= \arg \min_{\theta} \text{KL}[q(w|\theta)||P(w)] - \mathbb{E}_{q(w|\theta)}[\log P(D|w)]
$$

(6)

We consider the approximate posterior to be a fully factorised Gaussian [Graves, 2011], corresponding to a squared loss [Blundell et al., 2015]. The prior is also a Gaussian distribution corresponding to L2 regularization [Blundell et al., 2015]. Each layer has a single prior and thus a single prior mean, $\mu$, and a single prior standard deviation, $\sigma$. In a forward pass of the BGNN the weights are sampled from the posterior distribution. This creates a problem during backpropagation since we cannot define a gradient for this operation with respect to the weights. To tackle this problem we couple the BBB method with the reparametrisation trick [Kingma and Welling, 2014], as described in [Blundell et al., 2015], which replaces the sampling from the posterior with sampling from a unit Gaussian scaled by the posterior standard deviation, $\sigma^*$, and shifted by the posterior mean, $\mu^*$. The standard deviation needs to be positive, thus it is parameterised using a parameter, $\rho^*$, as $\sigma^* = \log(1 + \exp(\rho^*))$. Consequently, the variational parameters to be optimised are $\theta = (\hat{\mu}, \hat{\rho})$, where $\hat{\mu} = (\mu, \mu^*)$ and $\hat{\rho} = (\rho, \rho^*)$. 

![Figure 8: Architecture of the GNN](image-url)
4.2 Ensemble Kalman method for stress correction

The ensemble Kalman method is used to update the prior state vector of a system, $\mathbf{X}$. In this work the state vector corresponds to the microscale stress components predicted by the BGNN on the nodes of the mesh. In order to generate the prior ensemble we generate a number of samples from the BGNN. For node $j$ of sample $i$ we have

$$\tilde{\mathbf{x}}_{i,j} = [S_{ij}^{xx}, S_{ij}^{xy}, S_{ij}^{xz}, S_{ij}^{yy}, S_{ij}^{yz}, S_{ij}^{zz}]$$ (7)

Assuming that the mesh has $J$ nodes and that we draw $N$ samples from the BGNN the prior ensemble matrix can be expressed as

$$\mathbf{X} = [\tilde{\mathbf{x}}_{1}^{T}, ..., \tilde{\mathbf{x}}_{N}^{T}]$$ (8)

where

$$\tilde{\mathbf{x}}_{i} = [\tilde{\mathbf{x}}_{i,1}, ..., \tilde{\mathbf{x}}_{i,J}]$$ (9)

The updated stress distribution (posterior), $\mathbf{X}^*$, after observing the data $\mathbf{D}$, can be obtained as

$$\mathbf{X}^* = \mathbf{X} + \tilde{\mathbf{G}}(\mathbf{D} - \mathbf{H}\mathbf{X})$$ (10)

where $\mathbf{H}$ is the observation matrix that is used to provide the noise free value of the data, $\mathbf{HX}$, and $\tilde{\mathbf{G}}$ is the Kalman gain matrix, calculated as

$$\tilde{\mathbf{G}} = \tilde{\mathbf{\Sigma}}\mathbf{H}^{T}(\mathbf{H}\tilde{\mathbf{\Sigma}}\mathbf{H}^{T} + \mathbf{\Sigma}_{\epsilon})^{-1}$$ (11)

where $\mathbf{\Sigma}_{\epsilon}$ is the error of the data and $\tilde{\mathbf{\Sigma}}$ is the ensemble covariance calculated as

$$\tilde{\mathbf{\Sigma}} \approx \frac{1}{N-1} \sum_{i=1}^{N} \mathbf{X}^{(i)}\mathbf{X}^{(i)^{T}}$$ (12)

The data, $\mathbf{D}$, reflect the homogeneous Neumann conditions that we impose on every node

$$\mathbf{D}^{j} = \mathcal{N}(0, \mathbf{\Sigma}_{\epsilon})$$ (13)

Instead of explicitly defining the observation matrix we use an observation function, $h(\mathbf{x})$, (as described in appendix E.3) and we calculate the $\mathbf{HX}$ term on every node as

$$\mathbf{H}\mathbf{X} = h(\mathbf{X}) = \mathbf{S}^{j} \cdot \mathbf{n}^{j}$$ (14)

where $\mathbf{S}^{j}$ is the $[3 \times 3]$ symmetric microscale stress tensor and $\mathbf{n}^{j}$ is the $[3 \times 1]$ unit normal vector

$$\mathbf{S}^{j} = \begin{bmatrix} S_{xx}^{j} & S_{xy}^{j} & S_{xz}^{j} \\ S_{yx}^{j} & S_{yy}^{j} & S_{yz}^{j} \\ S_{zx}^{j} & S_{zy}^{j} & S_{zz}^{j} \end{bmatrix}, \quad \mathbf{n}^{j} = \begin{bmatrix} n_{xx}^{j} \\ n_{yy}^{j} \\ n_{zz}^{j} \end{bmatrix}$$ (15)

For more details the reader can refer to [appendix E]
5 Numerical Examples

5.1 Cubical Heterogeneous material

In this section we want to test our methodology using a dataset containing samples from a heterogeneous material. We will demonstrate how we can use our methodology to

- predict the microscale stress distribution
- predict the maximum microscale Von Mises stress over the patches
- quantify the uncertainty of the prediction
- improve the BGNN prediction using the online correction technique introduced in section 4.2

5.1.1 FEA set-up and generation of the training dataset

For the purpose of training our model we will work with a synthetic heterogeneous material from which we extract and examine cubical specimens. The size of each specimen is 2 units along each spatial dimension. The specimen has two elliptical pores, with random position, size and orientation and a distribution of 50 to 100 spherical pores with the same radius, $R = 0.08$ units. The spherical and elliptical pores are generated without intersecting. For the macroscale simulations, input of the BGNN, only the elliptical pores are taken into account and the spheres are ignored. Two realisations of the cubical specimen are shown in [Fig 9].

Figure 9: Two realisations of a cubical specimen from an heterogeneous materials with a bimodal distribution of random pores: large elliptical pores and smaller equally-sized spherical pores.

As already explained in [2.4] the input to the GNN is a patch of the geometry, not the entire structure, and the prediction of the GNN happens in a sub region of the patch that we called ROI. The patch and ROI size depend on the radius of interaction of the spherical pores. Based on the work of [Pilkey and Pilkey, 2008] on stress intensity factors, we assume that the effect of spherical voids on the global stress field fades out for a distance larger than 4 radii from the centre of the spherical voids. We combine this information with our previous work [Krokos et al., 2021] and we choose a patch size of $[18R \times 18R]$ and a ROI size of $[8R \times 8R]$, where $R$ is the radius of the spherical voids, as can be seen in [Fig 10].
The patches that are extracted never intersect with the exterior boundaries of the FE mesh. Additionally, we apply homogeneous Dirichlet boundary conditions away from the material specimen, as represented in [Fig 11], [Eq. 16].

\[
\begin{bmatrix}
E_{xx} & E_{xy} & E_{xz} \\
E_{xy} & E_{yy} & E_{yz} \\
E_{xz} & E_{yz} & E_{zz}
\end{bmatrix}
(X - X_0)^T
\] (16)

where \(E_{xx}, E_{xy}, E_{xz}, E_{yy}, E_{yz}, E_{zz}\) are far-field load parameters, \(X\) is the position of a point in \(\mathbb{R}^3\) and \(X_0\) is the initial position of the centre of the body in \(\mathbb{R}^3\).

For data pre-processing we use a simple linear rescaling to map the input data to a range from -1 to 1. Further information can be found in [appendix C].

The dataset we use consists of 200 FE simulations completed in 8.7 hours on 5 cores on an Intel® Xeon® Gold 6148 CPU @ 2.40GHz CPU. From every FE simulation we extract 5 patches, at random locations in the cubic specimen, resulting in 1,000 data points, 900 for training and 100 for testing.

5.1.2 Training parameters

For the training of the network we used a batch size of 2 and the Adam optimiser with an initial learning rate of \(1 \cdot 10^{-4}\) decreasing by 5% every 50 epochs. Additionally, we identified some key parameters for the training of the GNN and performed several tests to identify their optimum values. We concluded that for achieving optimum behaviour we will use a GNN with 5 GN blocks, residual connection, independent encoder, 128 filters and a maximum of 10 neighbours per node. More details can be found in [Appendix D.1 - D.5].

Training on 900 patches using the parameters identified above requires 6 hours on an Nvidia V100 GPU with 16GB of RAM with a maximum memory requirement of 4.3GB.
Figure 11: On the left (a) we can see a volume mesh of the multiscale structure. The buffer area where the mesh is coarse is visible. In the centre we can see the dense mesh area from where we extract the patches. On the right (b) we can see a zoom in the dense mesh region.

5.1.3 BGNN MAP prediction

In this section we will examine the mean prediction provided by the BGNN, without referring to the uncertainty estimation.

In [Fig 12a] we observe that the maximum stress components predicted by the BGNN are in a good agreement with the ones calculated by the FEA. Specifically, the coefficient of determination, $R^2$, for all the 6 stress components is larger that 0.98. In [Fig 12b] we observe similar results for the maximum Von Mises stress. Although the points are more scattered and the coefficient of determination dropped to 0.84, the accuracy is 96%.

Additionally, we examine the stress distribution on the FE mesh. In [Fig 13] we compare the microscale Von Mises stress in the ROI of a patch as predicted by the GNN with the one calculated by FE simulations. We observe that the predicted stress distribution is very close to the ground-truth.

5.1.4 BGNN uncertainty estimation

After studying the quality of the predictions provided by the mode of the BGNN, we propose to evaluate the ability of the BGNN to quantify the uncertainty of the prediction and provide credible intervals (CIs) that reflect the error in the prediction. From [Fig 14] we can see that the percentage of points inside the 95% CIs is 92%, which is satisfactory.
Figure 12: In both subfigures the x-axis corresponds to the FE prediction in the ROI of each patch and the y-axis to the BGNN prediction in the ROI of each patch. In the top subfigure, (a), we observe 6 diagrams corresponding to each of the maximum absolute microscale stress components. In the bottom subfigure, (b), we observe the maximum Von Mises stress.
Figure 13: Comparison between the Von Mises stress distribution as calculated by FEA (right) and the Von Mises stress distribution as predicted by the GNN (left).

Figure 14: In this figure we see the upper and lower 95% CIs for the maximum Von Mises stress prediction, blue and orange points respectively. The BGNN is able to include the real value in it's 95% CIs in 92% of the cases.
5.2 Online bias correction

In this section we will demonstrate the use of the Ensemble Kalman method, section [4.2], to improve the BGNN prediction. We will examine 2 realistic cases, first case is to improve the prediction of an under trained network and second case is to improve the prediction of a network in an extrapolation regime.

5.2.1 Under trained GNN

With the phrase under trained we refer to a network for which we use a small dataset for training. In [Fig 15] we show the accuracy plots for the mean BGNN prediction as a function of the epochs for 2 GNNs trained with the same parameters but the one on the left uses data from 32 FE simulations for training and the one on the right uses data from 100 FE simulations for training. The BGNN trained with more data performs better than the undertrained network.

![Accuracy plots for mean BGNN prediction](image)

Figure 15: In both diagrams we see accuracy curves defined using the mean BGNN prediction for the maximum Von Mises stress. Specifically, we see the accuracy as function of the training epochs for the training set (blue) and the test set (red). In the diagram on the left (a) the GNN is trained with 32 FE simulations, 160 patches, and in the diagram on the right (b) the GNN is trained with 100 FE simulations, 500 patches.

Using the Ensemble Kalman method we want to update the output distribution of the BGNN, prior, and obtain the posterior. In [Fig 16] we can see the posterior and the prior for the prediction of the maximum Von Mises stress. The coefficient of determination between the posterior and the FE results has a value of 0.3074 which is larger than the coefficient of determination between the prior and the FE results that has a value of 0.0176. Also we can see that best line that fits the posterior data is closer to the $y = x$ line (ideal result) compared to the best line that fits the prior data, i.e. the output of the BGNN, without the online correction. Therefore, the posterior mean better fits the data compared to the prior mean and thus the online stress correction resulted in an improvement of the mean BGNN prediction.

Furthermore, we want to examine the posterior distribution and not only the posterior mean value. In [Fig 17] we see the comparison between the prior 95% CIs (left) and the posterior 95% CIs (right). Firstly, we can observe that even in the prior case, where we have only used 32 FE simulations, the 95% CIs give a good estimation of where the real value is. We can see that the percentage of points that are inside the 95% CIs is 79%. Additionally, we see that this value climbs to 88% in the posterior case. This clearly means that by using the online stress update we were able to improve the BGNN.
prediction without adding more data to the training dataset. We note that we only update the mean value and the posterior standard deviation is taken equal to the prior one.

Lastly, in [Fig 18] we compare the mean Von Mises stress distribution on the mesh before and after the stress update with the one calculated through FE simulations. We can see that the posterior, middle image, is closer to the FE results, image on the left, compared to the prior, image on the right.

5.2.2 Prediction for out-of-training microstructural inputs

In this section, with the phrase “out-of-training” we refer to a network trained with spherical defects but evaluated in cases that have elliptical defects. The BGNN is extrapolating which is a particularly undesirable but common scenario for real applications. In [Fig 19] we show a realisation of the training dataset on the left and a realisation of the test dataset on the right.

In [Fig 20] we see the comparison between the prior 95% CIs (left) and the posterior 95% CIs (right). Firstly, we can observe that even in the prior case, where the BGNN extrapolates, the 95% CIs give a good estimation of where the real value is. We can see that the percentage of points that are inside the 95% CIs is 91%. Additionally, we see that this value climbs to 95% in the posterior case. Most importantly we see that using the ensemble Kalman method we managed to avoid underpredicting the maximum values. This clearly means that by using the Ensemble Kalman method we were able to improve the BGNN prediction without adding more data to the training dataset. We note that we only update the mean value and the posterior standard deviation is taken equal to the prior one.
Figure 17: Comparison between the prior and posterior 95% CIs for the maximum Von Mises stress in the ROI of the patches for a BGNN trained with only 32 FE simulation data. The diagram on the left, (a), is showing the upper and lower 95% CIs, blue and orange points respectively, before the stress update. While the diagram on the right, (b), is after the stress update. Two clusters, A and B, are marked before and after the stress correction highlighting that blue points, +95% CIs, that were below the $y = x$ line for the prior case have moved towards the $y = x$ line and in a lot of cases surpassed it, in the posterior case.

Figure 18: Comparison between the Von Mises stress before and after the online stress update with the one calculated through FE simulations. On the left we see the FE results, in the middle the posterior and on the right the prior. For the posterior and prior we show the mean prediction.
(a) realisation of the training dataset

(b) realisation of the test dataset

Figure 19: On the left (a) we see a realisation of the training dataset and on the right (b) of the test dataset. For the training dataset the porous phase is composed of spheres while for the test set of ellipsoids.

Figure 20: Comparison between the prior and posterior 95% CIs for the maximum Von Mises stress in the ROI of the patches for a BGNN trained on a dataset with spheres as microscale features and tested on a dataset with ellipses as microscale features. The diagram on the left, (a), is showing the upper and lower 95% CIs, blue and orange points respectively, before the stress update. While the diagram on the right, (b), is after the stress update. A cluster of points is marked before and after the stress correction highlighting that blue points, +95% CIs, that were below the $y = x$ line for the prior case have moved towards the $y = x$ line and in a lot of cases surpassed it, for the posterior case.
5.3 Dogbone

After demonstrating the applicability of our GNN and studying the online correction technique, in this subsection we apply the proposed deep learning methodology to a more challenging case. We also provide multiple examples of the Von Mises stress distribution on the surface of the mesh.

5.3.1 Training dataset

The geometry that we choose to model is a dogbone with a single hole in the middle that does not change from one realisation to another, and with a random distribution of 50 to 100 spherical pores. The size of the dogbone is 2 units along the \(x\) axis (length), 0.6 units along the \(y\) axis (height) and 0.08 units along the \(z\) axis (width). The radius of the hole is 0.032 units and the radius of the spherical pores is 0.016 units. The spherical pores may intersect with each other, with the boundaries of the geometry and the hole. For the macroscale simulations, input of the BGNN, only the hole is taken into account and the spherical pores are ignored. In [Fig 21] we show 2 examples of the geometry. For the boundary conditions we apply a displacement along the \(x\) direction of the same magnitude but of opposite sign.

![Figure 21: Two realisations of the dogbone used to train the GNN. Each dogbone has a cylinder-shaped hole and the porous material is defined via a random distribution of spherical voids.](image)

We performed 500 FE simulations completed in 5.7 hours on 10 cores on an Intel® Xeon® Gold 6148 CPU @ 2.40GHz CPU. From every FE simulation we extract 10 patches resulting in 5,000 data points, 4,000 for training and 1,000 for testing. As explained in section [5.1.1] the patch is of size \([18R \times 18R]\) and the ROI is of size \([8R \times 8R]\), where \(R\) is the radius of the spherical pores.
5.3.2 GNN parameters

For the GNN training we used the Adam optimiser with an initial learning rate of $1 \cdot 10^{-4}$ decreasing by 5% every 50 epochs. Additionally, we started with the parameters that we identified in [5.1.2], namely 5 GN blocks, residual connection, independent encoder, 128 filters and a maximum of 10 neighbours per node. After experiments we concluded that in this case the GNN can benefit from a maximum of 20 neighbours per node but all the other hyperparameters remained the unchanged. Training of the GNN was performed on an Nvidia V100 GPU with 16GB of RAM.

In this more challenging problem we investigate the idea of using dual convolutions. Instead of only considering the Euclidean neighbourhood we also consider the Geodesic neighbourhood of every node in order to perform joint Geodesic and Euclidean convolutions as explained in [3.3]. After experiments, we concluded that in our case the optimum value for the ratio between Geodesic and total convolutions is 75%. For more details the reader can refer to [Appendix D.6].

In addition, the reader can find experiments that support our choice to predict the full stress tensor instead of the Von Mises stress in [Appendix D.7] and our choice to work on patches of the structure instead of the entire structure in [Appendix D.8].

5.3.3 Numerical examples with Deterministic GNN

As can be seen in [Fig 22a] when training with 4000 patches the accuracy for the maximum Von Mises stress is 71% which is considerably lower than the accuracy reported for the cubical heterogeneous material case, 96%. In [Fig 22b] we can also evaluate the performance of the GNN on the entire dogbone and not only at patch level. The accuracy does not change considerably in this case but the data are more spread around the $y = x$ line which is indicated by the lower coefficient of determination $R^2$.

![Graph](image)

(a) evaluate on patches  
(b) evaluate on entire structure

Figure 22: In both subfigures the x-axis corresponds to the maximum Von Mises stress as calculated by FEA and the y-axis as calculated by the GNN. For the left subfigure, (a), we see results for the ROI of the patches while for the right subfigure, (b), for the entire dogbone.

Furthermore, we demonstrate the quality of the stress distribution. In [Fig 23] and [Fig 24] we show that the stress field predicted by the GNN is in good agreement with that provided by direct FEA.

Lastly, we are interested in studying the predicted stress distribution in the highly stressed regions of the specimen. In [Fig 25] we have zoomed in a region where a spherical void interacts strongly with the cylinder-shaped hole, and in [Fig 25] we have zoomed in at a strong interaction between 2 spherical
voids. We observe that in both cases the GNN successfully located the area where the maximum Von Mises occurs, predicted its value and the stress distribution around it.
Figure 24: Comparison between the Von Mises stress distribution, extracted from a cross section of the structure (top), as calculated by FEA (middle) and as predicted by the GNN (bottom). We can observe that the two distributions are qualitatively very similar. The GNN result is reconstructed from the union of multiple patch predictions where only the ROI is extracted.
Figure 25: In both subfigures we compare the maximum Von Mises stress as calculated by FEA and as predicted by the GNN. In the top subfigure (a), the maximum is due to an interaction between spherical voids and the cylinder-shaped hole while in the bottom, (b), due to an interaction between 2 spherical voids.
5.4 Variable Dimension Dogbone

In this section we demonstrate the ability of our method to be used for exploration of responses in a space of shapes described by a few geometrical parameters. To this end, we perform geometric learning in a family of dogbone shapes where not only the spherical void distribution will be different from realisation to realisation but also the cylindrical-hole and the dimensions of the dogbone. We focus on the dependence of the quality of the results with respect to the size of the dataset both in terms of of maximum Von Mises stress and stress distribution.

5.4.1 Training dataset

To create this dataset we follow the same process as in the “Dogbone” section, 5.3. The difference lays on the fact that here the dimension of the specimen is not constant. The length of the dogbone is equal to 2, the width varies from 0.06 to 0.12 and the height to varies from 0.24 to 0.7. Additionally, the number of cylindrical holes varies from 1 to 4 in every realisation, and their radii from 0.024 to 0.06. Two realisations of the variable dogbone geometry can be found in [Fig 26].

Figure 26: Two realisations of the variable dogbone structure. In the top subfigure we observe a dogbone with large height, small width and 4 cylindrical holes. In the bottom subfigure we observe a dogbone with small height, large width and 2 cylindrical holes.

5.4.2 Numerical Examples with variable dimension dogbone

We investigate the performance of the GNN with respect to the size of the training dataset. We use patches extracted from 100 FE simulations for the test dataset and 4 training datasets consisting of patches extracted from 50, 100, 200, and 400 FE simulations. Results can be found in [Fig 27]. From [Fig 27a] we can observe that the accuracy increases as we increase the size of the training dataset, from 50.21% for the GNN trained with 50 FE simulations to 64.30% for the GNN trained with 400
FE simulations. We see the opposite trend in [Fig 27b] where the loss decreases in fewer epochs and reaches a lower value as the size of the training dataset increases. Compared to the previous family of dogbones, where the dimensions of the dogbone and the cylindrical hole did not change from one realisation to another we can observe that the accuracy that can be achieved using 400 FE simulations decreases from 71% to 64% for the variable dogbone case. This is an expectable result since the space of shapes we try to learn now has more parameters than before. In order to achieve similar accuracy, we create additional 600 FE simulations and we train the GNN for 100 epochs using as initial weights the ones determined by the GNN trained with 400 FE simulations. The accuracy achieved for this case is 70%.

Figure 27: In the diagram on the left (a) we see accuracy curves defined using the maximum Von Mises stress and a 15% threshold for the relative error. Specifically, we see the accuracy as function of the training epochs. In the diagram on the right (b) we see the loss function as a function of the epochs. For both diagrams, coloured lines correspond to GNNs trained with datasets of different size, namely patches extracted from 50, 100, 200 and 400 FE simulations.

Additionally, in order to examine the stress distribution, in the following figure we have extracted the upper part of the structure so that we can observe the stress distribution inside. This can be found in [Fig 28], where we can see that the GNN successfully reconstructed the Von Mises distribution.

Lastly, want to examine the quality of the stress distribution with respect to the size of the training dataset. In [Fig 29] we see the Von Mises prediction for a case where not many strong interactions are present. We can observe that all the GNNs even the one trained with 100 FE simulations successfully predicts the correct Von Mises stress distribution. On the other hand, in [Fig 30] we can see a case where two intersecting defects, located at the corner of the domain, strongly interact and we can observe that the GNNs trained with less than 200 FE results fail to predict this interaction.

6 Conclusions

In this work we have developed a Neural Network (NN) based concurrent scale coupling method to deal with 3D multiscale stress predictions on porous media to overcome the need for computationally expensive direct numerical multiscale simulations. Our approach does not assume separation of scales nor does it require prior geometrical parametrisation of the multiscale problem.

Our approach is based on geometric learning which is more suitable for multiscale problems as it allows us to operate on a variable resolution mesh of the geometry rather than a fixed size voxel-based image. Additionally, we made the choice to analyse and predict only the surface stress of the geometry. This choice was based on 3 assumptions, 2 for the input and 1 for the output. For the input of the
Figure 28: Comparison between the Von Mises stress distribution, extracted from the top of the structure (top), as calculated by FEA (middle) and as predicted by the GNN (bottom). We can observe that the two distributions are qualitatively very similar. The GNN result is reconstructed from the union of multiple patch predictions where only the ROI is extracted.

GNN we assumed that the surface representation is a comprehensive representation of the geometry and that the trace of the smooth macro stress over the surface mesh is sufficient to inform the GNN of the macroscopic stress state. For the output, we assumed and proved, that in our dataset the maximum microscale stress field can be found on the surface of the structure.

This work extended our previous work on 2D image based CNNs [Krokos et al., 2021], to 3D graph-based Graph Neural Networks (GNNs). We found the same efficiency we had in the 2D case, to predict stress fields, local maxima and quantify uncertainty.

Moreover, we introduced a way to enforce physics contraints at evaluation time, using a Bayesian framework whereby the Bayesian GNN (BGNN) predictions were considered as prior, and posterior predictions that conform to homogeneous Neumann conditions on free surfaces of the porous materials were computed via an ensemble Kalman update. We have used this method to tackle problems com-
monly reported with data-driven methodologies, namely lack of training data and poor extrapolation ability. We have shown that without this online correction, downward bias is observed for maximum equivalent stresses, which is problematic. This effect is alleviated thanks to the online physics-based correction.

Lastly, we show that for Monte-Carlo simulations of porous microstructure, relatively few simulations are required to train the GNN, 50 simulations already give 50% accuracy. The trained GNN can be used in an inverse problem setting for instance for the optimisation of a CAD model under uncertainty stemming from the microstructure.
Figure 30: On the left the Von Mises distribution on the surface of a dogbone structure as calculated by FE simulations. On the right, we can see a zoom of the maximum Von Mises stress area. From top to bottom we see the FE simulation results, the GNN prediction for a GNN that was trained with patches extracted from 400, 200 and 100 FE simulations. We can observe that only the GNN that was trained using 400 FE simulations was able to correctly predict the maximum Von Mises stress that is created from the interaction between two spherical voids.
Acknowledgments

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764644.

This paper only contains the author’s views and the Research Executive Agency and the Commission are not responsible for any use that may be made of the information it contains.

Stéphane P.A. BORDAS received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 811099 TWINNING Project DRIVEN for the University of Luxembourg and from the Fonds National de la Recherche Luxembourg FNR under grant O17-QCCAAS-11758809.

We acknowledge the support of the Supercomputing Wales project, which is part-funded by the European Regional Development Fund (ERDF) via Welsh Government.

We thank Synopsys Northern Europe Ltd for its support and specifically Dr Viet Bui Xuan for his constructive suggestions during the planning and development of this research work.

References

Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M., and Wells, G. (2015). The fenics project version 1.5. arXiv preprint arXiv: 2001.08042.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv: 1607.06450.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv: 1806.01261.

Bishop, C. M. (1995). Neural Networks for Pattern Recognition. Oxford University Press, Inc., USA.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. arXiv preprint arXiv: 1505.05424.

Buda, M., Saha, A., and Mazurowski, M. A. (2019). Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm. Computers in Biology and Medicine, 109:218–225.

Deshpande, S., Lengiewicz, J., and Bordas, S. P. A. (2021). Fem-based real-time simulations of large deformations with probabilistic deep learning. arXiv preprint arXiv: 2111.01867.
Evensen, G. (1994). Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162.

Gendre, L., Allix, O., Gosselet, P., and Comte, F. (2009). Non-intrusive and exact global/local techniques for structural problems with local plasticity. *Computational Mechanics*, 44(2):233–245.

Geuzaine, C. and Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79:1309–1331.

Gong, S., Bahri, M., Bronstein, M. M., and Zafeiriou, S. (2020). Geometrically principled connections in graph neural networks. arXiv preprint arXiv: 2004.02658.

Graves, A. (2011). Practical variational inference for neural networks. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS’11, page 2348–2356, Red Hook, NY, USA. Curran Associates Inc.

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., and Cohen-Or, D. (2019). Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38:1–12.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. arXiv: 1512.03385.

Hesthaven, J., Zhang, S., and Zhu, X. (2015). Reduced basis multiscale finite element methods for elliptic problems. *SIAM Journal on Multiscale Modeling and Simulation*, 13:316–337.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv: 1207.0580.

Hinton, G. E. and van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT ’93, page 5–13, New York, NY, USA. Association for Computing Machinery.

Jiang, H., Nie, Z., Yeo, R., Farimani, A. B., and Kara, L. B. (2020). Stressgan: A generative deep learning model for 2d stress distribution prediction. arXiv: 2006.11376.

Kerfriden, P., Allix, O., and Gosselet, P. (2009). A three-scale domain decomposition method for the 3d analysis of debonding in laminates. *Computational Mechanics*, 44:343–362.

Khosla, M., Jamison, K., Kuceyeski, A., and Sabuncu, M. (2018). *3D Convolutional Neural Networks for Classification of Functional Connectomes: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings*, pages 137–145.

Kim, J., Lee, J. K., and Lee, K. M. (2016). Deeply-recursive convolutional network for image super-resolution. arXiv: 1511.04491.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. arXiv preprint arXiv: 1312.6114.

Krokos, V., Bui Xuan, V., Bordas, S., Young, P., and Kerfriden, P. (2021). A bayesian multiscale cnn framework to predict local stress fields in structures with microscale features. *Computational Mechanics*, pages 1–34.

Lei, H., Akhtar, N., and Mian, A. (2021). Picasso: A cuda-based library for deep learning over 3d meshes. arXiv preprint arXiv: 2103.15076.
Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M. (2017). Enhanced deep residual networks for single image super-resolution. arXiv preprint arXiv: 1707.02921.

Lino, M., Cantwell, C., Bharath, A. A., and Fotiadis, S. (2021). Simulating continuum mechanics with multi-scale graph neural networks. arXiv preprint arXiv: 2106.04900.

Logg, A., Mardal, K.-A., Wells, G. N., et al. (2012). Automated Solution of Differential Equations by the Finite Element Method. Springer.

Lu, H., Wang, H., Zhang, Q., Yoon, S. W., and Won, D. (2019). A 3d convolutional neural network for volumetric image semantic segmentation. Procedia Manufacturing, 39:422–428. 25th International Conference on Production Research Manufacturing Innovation: Cyber Physical Manufacturing August 9-14, 2019 — Chicago, Illinois (USA).

Lye, K. O., Mishra, S., and Ray, D. (2020). Deep learning observables in computational fluid dynamics. Journal of Computational Physics, 410:109339.

Masci, J., Boscaini, D., Bronstein, M. M., and Vandergheynst, P. (2018). Geodesic convolutional neural networks on riemannian manifolds. arXiv preprint arXiv: 1501.06297.

Mendizabal, A., Márquez-Neila, P., and Cotin, S. (2020). Simulation of hyperelastic materials in real-time using deep learning. Medical Image Analysis, 59:101569.

Mylonas, C., Tsialiamanis, G., Worden, K., and Chatzi, E. N. (2022). Bayesian graph neural networks for strain-based crack localization. In Madarshahian, R. and Hemez, F., editors, Data Science in Engineering, Volume 9, pages 253–261, Cham. Springer International Publishing.

Nie, Z., Jiang, H., and Kara, L. B. (2019). Stress field prediction in cantilevered structures using convolutional neural networks. Journal of Computing and Information Science in Engineering, 20(1).

Oden, J., Prudhomme, S., Romkes, A., and Bauman, P. (2006). Multiscale modeling of physical phenomena: Adaptive control of models. SIAM Journal on Scientific Computing, 28(6):2359–2389.

Oden, J. T., Vemaganti, K., and Moës, N. (1999). Hierarchical modeling of heterogeneous solids. Computer Methods in Applied Mechanics and Engineering, 172(1):3–25.

Paladim, D., Almeida, J., Bordas, S., and Kerfriden, P. (2016). Guaranteed error bounds in homogenisation: an optimum stochastic approach to preserve the numerical separation of scales. International Journal for Numerical Methods in Engineering, 110.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. (2021). Learning mesh-based simulation with graph networks. arXiv preprint arXiv: 2010.03409.

Pilkey, W. and Pilkey, D. (2008). Peterson’s stress concentration factors, third edition. Peterson’s Stress Concentration Factors, Third Edition, pages 1–522.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv: 1612.00593.

Raghavan, P. and Ghosh, S. (2004). Concurrent multi-scale analysis of elastic composites by a multi-level computational model. Computer Methods in Applied Mechanics and Engineering, 193(6):497–538.

Raissi, M., Yazdani, A., and Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science, 367(6481):1026–1030.
Rao, C. and Liu, Y. (2020). Three-dimensional convolutional neural network (3d-cnn) for heterogeneous material homogenization. *Computational Materials Science*, 184:109850.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. W. (2020). Learning to simulate complex physics with graph networks. arXiv preprint arXiv: 2002.09405.

Schlömer, N. (2021). pygmsh: A python frontend for gmsh.

Schult, J., Engelmann, F., Kontogianni, T., and Leibe, B. (2020). Dualconvmesh-net: Joint geodesic and euclidean convolutions on 3d meshes. arXiv preprint arXiv: 2004.01002.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W. R., Bridgland, A., Penedones, H., Petersen, S., Simonyan, K., Crossan, S., Kohli, P., Jones, D. T., Silver, D., Kavukcuoglu, K., and Hassabis, D. (2019). Protein structure prediction using multiple deep neural networks in the 13th critical assessment of protein structure prediction (casp13). *Proteins: Structure, Function, and Bioinformatics*, 87(12):1141–1148.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W. R., Bridgland, A., Penedones, H., Petersen, S., Simonyan, K., Crossan, S., Kohli, P., Jones, D. T., Silver, D., Kavukcuoglu, K., and Hassabis, D. (2020). Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710.

Sullivan, C. B. and Kaszynski, A. (2019). PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK). *Journal of Open Source Software*, 4(37):1450.

Sun, Y., Hanhan, I., Sangid, M. D., and Lin, G. (2020). Predicting mechanical properties from microstructure images in fiber-reinforced polymers using convolutional neural networks. arXiv: 2010.03675.

Vlassis, N. N., Ma, R., and Sun, W. (2020). Geometric deep learning for computational mechanics part i: anisotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering*, 371:113299.

Vu, H., Kim, H.-C., Jung, M., and Lee, J.-H. (2020). fMRI volume classification using a 3d convolutional neural network robust to shifted and scaled neuronal activations. *NeuroImage*, 223:117328.

Zagoruyko, S. and Komodakis, N. (2017). Wide residual networks. arXiv: 1605.07146.

Zohdi, T. and Wriggers, P. (2005). *An Introduction to Computational Micromechanics*, volume 20.

Évariste Sanchez-Palencia (1987). *General introduction to asymptotic methods*, volume 272.

### Appendix A  Accuracy

Below we can see the algorithm that we use to calculate the accuracy of the GNN prediction [Algorithm 1].

### Appendix B  Volume and surface mesh

As discussed in section [3.2] the input and output is the surface and not the volume stress. In [Fig. 31] we see an example of a surface stress extracted from the volume mesh. We can observe that the maximum value of the Von Mises stress is the same in both cases and the location where the maximum Von Mises stress occurs is the same as well.
Algorithm 1 Compute accuracy

1: function \texttt{acc}(\texttt{datapoints}, \texttt{threshold} = 0.1)
2: \begin{align*}
3: N &= \texttt{length}(\texttt{datapoints}) \\
4: \texttt{accepted} &= \texttt{zeros}(N) \\
5: \end{align*}
6: \begin{align*}
7: \textbf{for} \texttt{patch} \textbf{in} \texttt{datapoints} \textbf{do} \\
8: \end{align*}
9: \begin{align*}
10: S_{\text{NN}} &= \texttt{patch}.\texttt{prediction} & \triangleright \text{get the predicted stress tensor for the current patch} \\
11: S_{\text{FE}} &= \texttt{patch}.\texttt{ground\_truth} & \triangleright \text{get the real stress tensor for the current patch} \\
12: S_{\text{NN\_ROI}} &= S_{\text{NN}}[\texttt{patch}.\texttt{ROI}] & \triangleright \text{extract the predicted stress values present in the ROI} \\
13: S_{\text{FE\_ROI}} &= S_{\text{FE}}[\texttt{patch}.\texttt{ROI}] & \triangleright \text{extract the real stress values present in the ROI} \\
14: y_{\text{NN}} &= \texttt{VM}(S_{\text{NN\_ROI}}) & \triangleright \text{calculate the Von Mises stress for the predicted stress tensor} \\
15: y_{\text{FE}} &= \texttt{VM}(S_{\text{FE\_ROI}}) & \triangleright \text{calculate the Von Mises stress for the real stress tensor} \\
16: y_{\text{NN\_max}} &= \max(y_{\text{NN}}) & \triangleright \text{get the maximum Von Mises stress for the predicted stress tensor} \\
17: y_{\text{FE\_max}} &= \max(y_{\text{FE}}) & \triangleright \text{get the maximum Von Mises stress for the real stress tensor} \\
18: \text{error} &= \frac{|y_{\text{NN\_max}} - y_{\text{FE\_max}}|}{y_{\text{FE\_max}}} & \triangleright \text{calculate the relative error} \\
19: \textbf{if} \texttt{error} \leq \texttt{threshold} \textbf{then} \\
20: \hspace{1cm} \texttt{accepted}[\texttt{patch}] = 1 & \triangleright \text{decide if the error is acceptable} \\
21: \end{align*}
22: \begin{align*}
23: \texttt{accuracy} &= \texttt{sum}(\texttt{accepted})/N \\
24: \end{align*}
25: \begin{align*}
26: \textbf{return} \texttt{accuracy} \\
27: \end{align*}
Figure 31: Two structures corresponding to one quarter of a random realisation of the dogbone geometry. In the diagram on the left (a) we see the Von Mises stress distribution on the volume mesh. In the diagram on the right (b) we see the Von Mises stress distribution on the surface mesh. We can see, in the red circles, that the maximum values in both structures are in the same location.

Appendix C  Data pre-processing

Data pre-processing is a standard technique used in data driven methodologies. A common practice is to pre-process the input data usually with a simple linear rescaling [Bishop, 1995]. In this work we perform rescaling in order to both have smoother convergence of the optimiser and to restrict the space of possible inputs. We divide the input, and the output for consistence reasons, of the NN with the maximum absolute component of the stress tensor. This rescales the input data to a range $[-1, 1]$. This rescaling alleviates the differences in scales across the data points which helps the optimiser to avoid bad local minima and converge to a stable solution. Additionally, scaling the input in the case of linear elasticity caries a deeper physical meaning. Scaling the macroscale stress field with a scaling factor, $k$, results in scaling the microscale stress tensor with the same scaling factor, $k$. By taking advantage of this linear relationship we can effectively train our NN in a range of values from -1 to 1 and then evaluate it to an input of any range. This is achieved by dividing the new input with the scaling factor $k$, to map it to a range from -1 to 1. To retrieve the real output, microscale tensor, the output of the NN needs to be multiplied with the scaling factor $k$.
Appendix D  GNN parameters

We identified some key parameters for the training of the GNN and performed several tests to identify their optimum values. The parameters that we examined is the number of filters in the MLPs, the number of GN Blocks, the number of maximum neighbours per node, the existence of a skip connection between the input and the output of the network and finally the type of encoder. For these tests we used 500 patches as the training set and 100 patches for the test set.

D.1 Skip Connection

We suggest that using a skip connection to add the input macroscale stress tensor to the output of the network will improve GNN’s performance. The reason behind this is that the GNN will learn how the microscale stress field deviates from the macroscale stress field instead of learning the microscale stress field from zero which we believe that it is easier and it should also improve the generalization ability of the network. In order to validate our assumption we train 2 identical GNNs with the exact same parameters and training set but one of them has a skip connection and the other does not. Both networks have MLPs with 128 filters, 10 GN blocks and each node has at most 10 neighbours. The results can be found in [Fig 32] where we can observe that the GNN trained with the skip connection has smoother convergence compared to the GNN trained without the skip connection. Also, the skip connection improved accuracy from 65% to 80%. We conclude that indeed the skip connection helped not only to have more stable training but also to improve the accuracy and thus we decide to include it in our architecture.

![Figure 32: In the diagram we see accuracy curves for the test dataset defined using the maximum Von Mises stress. The yellow line corresponds to a GNN trained with a skip connection to add the input macroscale stress to the output of the GNN and the the blue line corresponds to a GNN trained without this skip connection. We observe that the skip connection results in smoother training and higher accuracy.](image)

D.2 Number of Filters

Another important parameter that heavily influences not only the accuracy of the GNN but also the training time and memory requirements is the number of filters in the MLPs. In order to identify the minimum number of filters that result in optimum accuracy we trained 3 GNNs with 64, 128 and 256 filters in the MLPs and we kept all the other parameters the same. Specifically, all networks have the skip connection described in [D.1], 10 GN blocks and each node has at most 10 neighbours. The results
can be found in [Fig 33] where we observe that 128 and 256 filters result in the same accuracy which is higher than the accuracy for the 64 filters, 80% compared to 72%. Additionally, the training time for the GNN with the 256 filters is 12.7 hours and the maximum required memory is 15.4 GB while for the 128 filter GNN the training was completed in 5.8 hours and the maximum required memory was 7.7 GB. By choosing to use 128 filters in the MLPs of the GNN we achieve the same accuracy as in the 256 filters GNN but with a 54% decrease in the training time and a 50% decrease in memory requirements.

Figure 33: In the diagram we see accuracy curves for the test dataset defined using the maximum Von Mises stress. Coloured lines correspond to networks trained with different number of filters, namely 64, 128 and 256. We observe that 128 and 256 filters result in the same accuracy 80% where 64 filters result in a lower accuracy of 72%.

D.3 Independent Decoder

As already discussed in [3.6], in the first layer of the GNN we choose to encode the edge and node features independently into the latent dimension as suggested by [Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Mylonas et al., 2022]. We want to test if this choice results in an improved GNN and thus we perform an experiment between 2 GNNs with the same parameters where one uses a GN block to encode the inputs into the latent dimension and the other encodes them independently. Both of the GNNs have 10 GN blocks, 128 filters in the MLPs, the skip connection described in [D.1] and each node has at most 10 neighbours. The results can be found in [Fig 34] where we observe that both GNNs have similar accuracy curves and they both have a final accuracy of 80%. Nevertheless, we can observe that in the independent encoder case the accuracy starts increasing sooner, at epoch 50, where in the GN block case it starts increasing later, at epoch 120. Lastly, the independent encoder version involves slightly less calculations and results in a 5% decrease in training time. Consequently, we choose to use the independent encoder GNN.

D.4 Number of GN blocks

Another important parameter that affects both the memory requirement and the training time is the number of residual GN blocks. Because we are using residual connections we do not expect a decrease in accuracy as we add more GN blocks but we are expecting that there should be a saturation point where the GNN does not benefit anymore from the GN blocks but it performs unnecessary calculations resulting in increased computational cost. To validate this assumption and determine the most proper number of GN blocks we train 3 GNNs with the same parameters but different number of GN blocks.
Figure 34: In the diagram we see accuracy curves for the test dataset defined using the maximum Von Mises stress. The yellow line corresponds to a GNN trained with an independent encoder and the blue line corresponds to a GNN trained with a GN block as an encoder. We observe that both GNNs have similar accuracy curves with the same final accuracy but the GNN with the independent encoder starts increasing it’s accuracy sooner, epoch 50, compared to the other one, epoch 120.

All the GNNs have 128 filters in the MLPs, the skip connection described in [D.1], independent encoder and each node has at most 10 neighbours. The results can be found in [Fig 35] where we observe that the GNN with only 3 GN blocks presents a decrease of 10% in the test accuracy compared to the other two. Also we can see that the GNNs with 5 and 10 GN blocks have similar test accuracy curves and they both reach a test accuracy of 80%. We decide to opt for the GNN with the 5 GN blocks that presents optimum accuracy without additional computational cost. This results in a training time of 3.1 hours and a maximum memory of 4.3 GB which is a decrease of 45% in training time and 44% in memory requirements compared to the GNN with the 10 GN blocks.

D.5 Number of neighbours

Lastly, we want to study the effect of the maximum number of neighbours on the GNN training. A small number of maximum neighbours will result in a graph where disconnected areas of the FE mesh do not share an edge and thus cannot exchange information, for instance microscale and macroscale features. We train 3 GNNs with the same parameters apart from the maximum number of neighbours that have values 5, 10 and 15. All the GNNs have 5 GN blocks, 128 filters in the MLPs, the skip connection described in [D.1] and independent encoder. The results can be found in [Fig 36] where we observe that 10 and 15 neighbours result in the same accuracy which is higher than the accuracy for the 5 neighbours, 81% compared to 75%. Additionally, the training time for the GNN with the 15 neighbours is 27% higher and the memory requirements 31% higher compared to the GNN with 10 neighbours. We conclude that in our case we do not have a reason to use more than 10 neighbours although in denser meshes this number could be different.

D.6 Geodesic and Euclidean Convolutions

We study the effect of the joint convolutions by training 6 GNNs with different number of Geodesic and Euclidean filters. We keep the total number of filters constant to 128 but we change the ratio between Geodesic and total (Euclidean + Geodesic) filters. We investigate the case where the ratio is 0% (only Euclidean), 25%, 50% and 75%, 87.5% and 100% (only Geodesic). All the GNNs have 5 GN blocks, residual connection, independent encoder, 128 filters and a maximum of 20 neighbours per
node. The results can be found in [Fig 37]. We can observe that both for the accuracy and the loss the worst case is when the ratio is 0, so no Geodesic convolutions are present. Both the accuracy and the loss improve (accuracy increases and loss decreases) as we increase the ratio until the value 75%. After that no further improvement in performance is observed, both the accuracy and loss curves for ratio values 75% and 87.5% are the same. On the contrary when the ratio becomes 1 there is a substantial drop in performance, which is expected since no Euclidean convolutions are performed. We conclude that a 75% ratio is the most beneficial for this case.

D.7 Full stress field VS maximum stress

We investigate the option to directly predict the maximum Von Mises stress on the surface of the dogbone structure. To this end, we train 2 identical GNNs on patches extracted from 100 FE simulations and we evaluate them in a different set of patches extracted from 100 FE simulations. One of the GNNs directly predicts the maximum Von Mises stress and the other the full stress tensor. Both GNNs have 5 GN blocks, independent encoder, 128 filters (32 Euclidean and 96 Geodesic) and a maximum of 20 neighbours per node. The results can be found in [Fig 38]. We observe that the 2 GNNs have similar accuracy, but the one predicting the full stress has slightly higher values. We conclude that predicting the full stress is beneficial since it not only results in better prediction of the maximum Von Mises stress but it also provides us with the full stress tensor that can be used for instance for the online bias corrections.

D.8 Patches VS full structure

Lastly, we want to evaluate to what degree our choice to train the GNN using patches of the geometry affects the quality of the results. This choice stems from the fact that defects only locally affect the macroscale stress field and distant areas do not offer significant information. On one hand, training on patches has the benefit of making the GNN unaware of the specific structure and it encourages it to focus on predicting how the microscale features affect the global stress field, thus leading to improved generalisation. On the other hand, our choice for the size of ROI and patch has to be careful so that the network has all the necessary information to make predictions in the ROI. Choosing to
train the GNN using the entire structure alleviates us from this problem. We perform an experiment to compare the two strategies. We train two GNNs with the same parameters, namely 5 GN blocks, residual connection, independent encoder, 128 filters (32 Euclidean and 96 Geodesic) and a maximum of 20 neighbours per node. The first one is trained using patches from 400 FE simulations (10 patches from each simulation) and the second one is trained directly on the 400 FE simulation results. We evaluate both GNNs on a separate set of patches extracted from 100 FE simulations. The results can be found in [Fig 39]. We can observe that for the GNN that was trained on patches the maximum Von Mises in the ROI of the patches is less scattered around the $y = x$ line, true value. The coefficient of determination for the patch case is 0.79 compared to 0.71 for the full case and the accuracy significantly drops from 70% for the patch case to 55% for full dogbone case. Thus we conclude that our choice to train the GNN with patches indeed leads to improved generalisation. We also compare the Von Mises stress distribution on the surface of a dogbone predicted by the GNN trained with patches with the GNN trained with the entire structure. In [Fig 40] we see that the GNN that was trained using patches more accurately predicts the high stress area compared to the GNN that was trained with the full structure.

Appendix E Ensemble Kalman Method

E.1 Kalman method for data assimilation

The Kalman method can be used to update the $n$-dimensional state vector of a system, $x$, after observing some data, $d$. We assume that the state vector follows a Gaussian distribution, $p(x)$, that we call prior. This distribution has a mean $x_\mu$ and a variance $\Sigma$.

$$p(x) = \mathcal{N}(x_\mu, \Sigma)$$ (17)

The data, $d$, are assumed to be noisy. Specifically, we assume that the data follow a Gaussian distribution, centered around $Hx$ which is the value of data if no noise is present, and with a variance $\Sigma_\epsilon$, that accounts for the noise. The matrix $H$ is called observation matrix. Consequently, the data likelihood can be expressed as
Figure 37: In the diagram on the left (a) we see accuracy curves defined using the maximum Von Mises stress and a 15% threshold for the relative error. Specifically, we see the accuracy as function of the training epochs. In the diagram on the right (b) we see the loss function as a function of the epochs. For both diagrams, coloured lines correspond to GNNs trained with different Geodesic to total (Euclidean + Geodesic) filter ratio, namely 0% (only Euclidean), 25%, 50%, 75%, 87.5% and 100% (only Geodesic).

\[ p(d|x) = N(Hx, \Sigma_e) \]  
Using the Bayes theorem, we can obtain the posterior distribution, \( p(x|d) \).

\[ p(x|d) \propto p(d|x)p(x) = N(x; x_\mu^*, \Sigma^*) \]  
Where the posterior mean is calculated as

\[ x_\mu^* = x_\mu + G(d - Hx_\mu) \]  
And the posterior variance is calculated as

\[ \Sigma^* = (I - GH)\Sigma \]  

E.2 Ensemble Kalman method for data assimilation

The ensemble Kalman method is an approximate version of the Kalman method where the state distribution is represented by a sample (or an ensemble) of the distribution of size \( N \). This leads to the creation of a \([n \times N]\) prior ensemble matrix whose columns correspond to the ensemble samples.

\[ X = [x_1, ..., x_N] \]  
The state covariance, \( \Sigma \), is replaced with the ensemble covariance, \( \hat{\Sigma} \).

\[ \hat{\Sigma} \approx \frac{1}{N-1} \sum_{i=1}^{N} X^{(i)}X^{(i)T} \]
Figure 38: Accuracy curves as a function of the threshold value used for the relative error. The accuracy curves are defined using the maximum Von Mises stress. The blue line corresponds to a GNN that predicts the full microscale stress distribution first and then from this the maximum Von Mises stress in the ROI of the patch. The orange line corresponds to a GNN that directly predicts the maximum Von Mises in the ROI of the patch.

A data matrix, $D$, is created such as

$$D = [d_1, ..., d_N]$$

$$d_i = d + \zeta_i$$

$$\zeta_i \sim \mathcal{N}(0, \Sigma_e)$$

The new Kalman gain matrix is calculated as

$$\tilde{G} = \tilde{\Sigma}H^T(\tilde{H}\tilde{\Sigma}H^T + \Sigma_e)^{-1}$$

The posterior distribution, $X^*$, is calculated as

$$X^* = X + \tilde{G}(D - HX)$$

**E.3 Observation matrix free implementation**

We can further simplify the calculations by avoiding to explicitly define the observation matrix. Instead we can define a function that will provide the noise free value of the data. This function is called observation function and is of the form

$$h(x) = Hx$$
Figure 39: In both diagrams the x-axis corresponds to the maximum Von Mises stress in the ROI of the patches calculated by FE simulations and the y-axis to the maximum Von Mises stress in the ROI of the patches predicted by the GNN. The yellow lines correspond to the 15% relative error threshold. The accuracy is defined as the percentage of points with relative error less than 15%. The image on the left (a) corresponds to a GNN trained with patches extracted by 400 FE simulations and the image on the right (b) to a GNN trained directly on the same 400 FE simulations. We can observe that the GNN trained on patches outperforms the GNN trained on the full structures both in terms of accuracy and coefficient of determination.

The posterior can be written as

\[ X^* = X + \frac{1}{N-1} \Lambda(HA)^T P^{-1} (D - HX) \]  

(29a)

\[ HX = h(X) \]  

(29b)

\[ A = X - \frac{1}{N} X \]  

(29c)

\[ HA = HX - \frac{1}{N} HX \]  

(29d)

\[ P = \frac{1}{N-1} HA(HA)^T + \Sigma_\epsilon \]  

(29e)
Figure 40: Von Mises stress on the surface of a dogbone structure (top). On the bottom of the image we have zoomed in the high stress area. From left to right we see the FE result, the prediction of the GNN trained with patches and the GNN prediction of the GNN trained with full structures. We observe that the GNN that is trained with patches is in better agreement with the FE results compared to the one trained on the full structures.