

# MAGNET: A Graph U-Net Architecture for Mesh-Based Simulations

Saurabh Deshpande<sup>a</sup>, Stéphane P.A. Bordas<sup>a</sup>, Jakub Lengiewicz<sup>a,b</sup>

<sup>a</sup>*Department of Engineering; Faculty of Science, Technology and Medicine; University of Luxembourg*

<sup>b</sup>*Institute of Fundamental Technological Research, Polish Academy of Sciences*

---

## Abstract

In many cutting-edge applications, high-fidelity computational models prove too slow to be practical and are thus replaced by much faster surrogate models. Recently, deep learning techniques have become increasingly important in accelerating such predictions. However, they tend to falter when faced with larger and more complex problems. Therefore, this work introduces MAGNET: Multi-channel Aggregation Network, a novel geometric deep learning framework designed to operate on large-dimensional data of arbitrary structure (graph data). MAGNET is built upon the MAg (Multichannel Aggregation) operation, which generalizes the concept of multi-channel local operations in convolutional neural networks to arbitrary non-grid inputs. The MAg layers are interleaved with the proposed novel graph pooling/unpooling operations to form a graph U-Net architecture that is robust and can handle arbitrary complex meshes, efficiently performing supervised learning on large-dimensional graph-structured data. We demonstrate the predictive capabilities of MAGNET for several non-linear finite element simulations and provide open-source datasets and codes to facilitate future research.

*Keywords:* Geometric Deep Learning, Mesh Based Simulations, Finite Element Method, Graph U-Net

---

## 1. Introduction

Computational models are essential tools for studying, designing, and controlling complex systems in many fields, including engineering, physics, biology, economics, and social networks. These models are often based on physical laws and mathematical equations, with partial differential equations (PDEs) being a common tool for describing how quantities change over space and time. In mechanics and physics, the PDEs are most commonly solved with numerical methods upon earlier space- and time- discretization, and a large number of domain-specific computational models have been developed so far, with the finite element method (FEM) and the finite volume method (FVM) being the most commonly used approaches in solid- and fluid mechanics, respectively. However, despite significant advances in computational performance over the last decade, such high-fidelity numerical simulations

remain prohibitively expensive for many important applications, including emerging areas such as real-time feedback/control in the computer-assisted surgery [Johnsen et al., 2015; Bui et al., 2018] or soft robotics [Rus and Tolley, 2015; Goury and Duriez, 2018]. Speeding up such models whilst maintaining the desired accuracy is an active area of research, and one of the main motivations of the present work.

Recently, deep learning (DL) techniques have taken a center stage across many disciplines. The DL models have proven to be accurate and efficient in predicting non-trivial nonlinear relationships in data. For that reason, they have been tried for a variety of applications also in mechanics, such as surrogate modelling [Mendizabal et al., 2019; Deshpande et al., 2022; Krokos et al., 2022b; Šarkić Glumac et al., 2023] or model discovery and calibration [Huang et al., 2020; Thakolkaran et al., 2022]. The deep neural network approaches can be categorized with respect to how they use the data and *a priori* knowledge about the modelled system. In purely data-driven approaches, DL models rely on performing supervised learning on either experimental or numerically generated data and are agnostic to the underlying physics or model. As such, they are able to reproduce the physics-based relationship by implicitly learning on a relatively large amount of data [Runge et al., 2017; Aydin et al., 2019; Daniel et al., 2020; Kochkov et al., 2021]. If the *a priori* information about the modelled system is introduced, such networks are termed as Physics Informed Neural Networks (PINNs) [Raissi et al., 2019; Samaniego et al., 2020; Henkes et al., 2022; Klein et al., 2022; Zhang et al., 2022]. With respect to the purely data-driven approaches, PINNs are generally more accurate, require less data for training, and possess better generalization capabilities. The framework presented in the present work is generally applicable to both cases, however, for the sake of clarity, we will later only focus on purely data-driven types of networks. In any case, once trained, the DL models can be used as fast surrogates for computationally expensive high-fidelity numerical methods.

The focus of the present work is on high-dimensional relationships in which the sizes of inputs and/or outputs are large. Examples of such relationships can be found, for instance, in experimental full-field measurement data, such as our recent work on medical imaging [Lavigne et al., 2022], or in synthetic mesh data generated from finite element simulations, [Lorente et al., 2017; Pellicer-Valero et al., 2020]. Although DL techniques have generally shown great success as efficient surrogates to computationally expensive numerical methods in scientific computing, some of the popular existing machine learning approaches are still based on fully-connected deep networks which are not suitable for high-dimensional inputs/outputs. As an alternative, the application of Convolutional Neural Networks (CNNs) has proven a promising performance in a wide variety of applications, also including accelerating non-linear finite element/volume simulations [Obiols-Sales et al., 2020; Rao and Liu, 2020; Krokos et al., 2022b; Deshpande et al., 2022]. CNNs are designed to learn a set of fixed-size trainable local filters (convolutions), thus reducing the parameter space while being capable to capture non-linearities. In the context of computational mechanics, local convolutions leverage the natural local correlation of nearby nodes, which leads to more efficient neural network architectures, both in terms of training- and prediction times. Moreover, one can observe that the CNN architectures have a close analogy to some iterative solution schemes known in scientific computing [Wang et al., 2020a; Brenner and Scott,

2008]. This provides them with an additional interpretation of being trainable iterative computational schemes to solve sets of non-linear equations, rather than general-purpose black-box approximators.

However, there is one important limitation that prevents CNNs from being of general purpose. The problem is that they only work well with grid-like structure data, such as images or structured meshes, which greatly hinders their use for many real-world applications where data is structured differently. Although there are some attempts to alleviate that problem in the context of FEM data, for instance, combining finite elements with an immersed-boundary method [Brunet et al., 2019], or embedding a precomputed coordinate mapping into the classic grid [Gao et al., 2021], the effectiveness of those methods is limited to simple irregular domains and remains challenging for complex geometries in general. A definitive solution to that problem has only been brought by Graph Neural Networks (GNNs)—architectures that directly handle arbitrarily-structured inputs/outputs. They belong to the recently emerged family of Geometric Deep Learning (GDL) methods which focus on neural networks that can learn from non-Euclidean input such as graphs and, more generally, manifolds [Bronstein et al., 2017][Wu et al., 2021]. Because of their ability to handle more general structured data, GNNs are gaining increasing importance also in surrogate modelling in scientific computing [Sanchez-Gonzalez et al., 2020][Vlassis et al., 2020][Pfaff et al., 2021][Krokos et al., 2022a][Gao et al., 2022]. However, these approaches are based on relatively simple message passing schemes, which are sub-optimal for learning on high non-linear regression tasks. In this work, we propose a novel local aggregation technique, which we denote as Multichannel Aggregation layer, MAg, which performs multichannel localised weighted aggregations, that can be seen as a direct extension of the traditional convolution layer in CNNs. Thanks to that, we are able to directly adapt some of the mechanisms/layers developed for CNNs to create efficient graph neural network architectures.

One mechanism that can improve the efficiency and predictive capabilities of convolutional and graph neural networks is the application of down-sampling (coarsening) and up-sampling (refinement) layers. In the context of CNNs, the focus is on encoder-decoder architecture frameworks, such as U-Net, which has been successfully implemented in various applications, including computer vision [Ronneberger et al., 2015; Çiçek et al., 2016], signal processing [Hennequin et al., 2020; Ren et al., 2021], and scientific machine learning [Mendizabal et al., 2019; Wang et al., 2020b; Pant et al., 2021; Le et al., 2022]. While the CNN-based U-Net approaches are limited to grid data, their graph-based version, known as graph U-Net, can provide the desired generality. Recently, various graph coarsening approaches have been proposed [Bianchi et al., 2019; Lee et al., 2019; Gao and Ji, 2019; Cai et al., 2021], which serve the same function as pooling layers in CNNs, helping to reduce the size of a graph while maintaining essential properties of the processed data. In this work, we propose a novel graph pooling/unpooling operation (coarsening/refinement), that enables us to create a graph U-Net architecture, MAgNET, that can operate on arbitrary graphs. Our pooling layers are directly inspired by CNNs, where we extend the concept of pooling over local patches in regular grids to variable size non-overlapping cliques in graphs. This allows us to precompute coarsened graphs that are only based on the input graph topology,

which is independent of data (i.e., node features). In the context of GNN-accelerated FEM simulations, a similar concept has been proposed by [Black and Najafi, 2022], however, their implementation is limited to regular meshes for simple two-dimensional geometries and linear elastic problems. Our approach enables computationally efficient deep learning models for non-linear problems involving arbitrary meshes, which is an important advancement for this field.

In summary, we introduce a novel graph U-Net framework comprising the proposed MAg and graph pooling/unpooling layers. The MAg layer captures local regularities in the input data, while the interleaved pooling layers reduce the graph representation to a smaller size while preserving important structural information. This enables us to efficiently implement our framework for large-scale problems. The proposed MAg and graph pooling layers are direct analogues of respective CNN U-Net layers and are also compatible with many state-of-the-art graph neural network layers. We elaborate on this point in the paper, providing a qualitative comparison of the proposed MAg layer with several existing graph layers. To validate the predictive capabilities of our framework, we apply it to several non-linear relationships obtained through finite element analysis and cross-validate it with predictions given by our CNN U-Net architecture [Deshpande et al., 2022]. To increase the impact of our work, we provide source codes, datasets, and procedures to generate the datasets utilized in this work, which can be found in the MAgNET repository: <https://github.com/saurabhdeshpande93/MAgNET>.

The paper is organized as follows. In Section 2 we present the novel MAgNET framework, as well as its particular application to the hyperelastic FEM-based datasets. Then, in Section 3, we provide details of implementation and a thorough study of MAgNET for several 2D and 3D benchmark non-linear FEM examples. The conclusions and future research directions are summarised in Section 4.

## 2. MAgNET Deep Learning Framework

In this section, we will propose a novel graph-based encoder-decoder (U-Net) deep-learning framework. We will provide a general formulation, in which inputs and outputs follow a certain graph topology (that is expressed by an adjacency matrix  $\mathbf{A}$ ). The graph expresses an assumed structure of correlations within input/output data and allows us to devise a *robust* DNN architecture defining a non-linear mapping between inputs and outputs. We will apply this general framework to *mesh-based* graphs. Such mesh topology of data is characteristic to spatially discretised numerical solution schemes for PDEs in scientific computing. In particular, we will focus on hyperelastic problems in solid mechanics, for which the training/testing data is obtained through the finite element method (see also the schematics in Figure 1).

In Section 2.1 we will provide an overview of the proposed Graph U-Net framework MAgNET. Next, in Sections 2.2-2.4 we will introduce the building blocks of MAgNET. In particular, in Section 2.2, we will introduce the adjacency matrix representation of the mesh-based



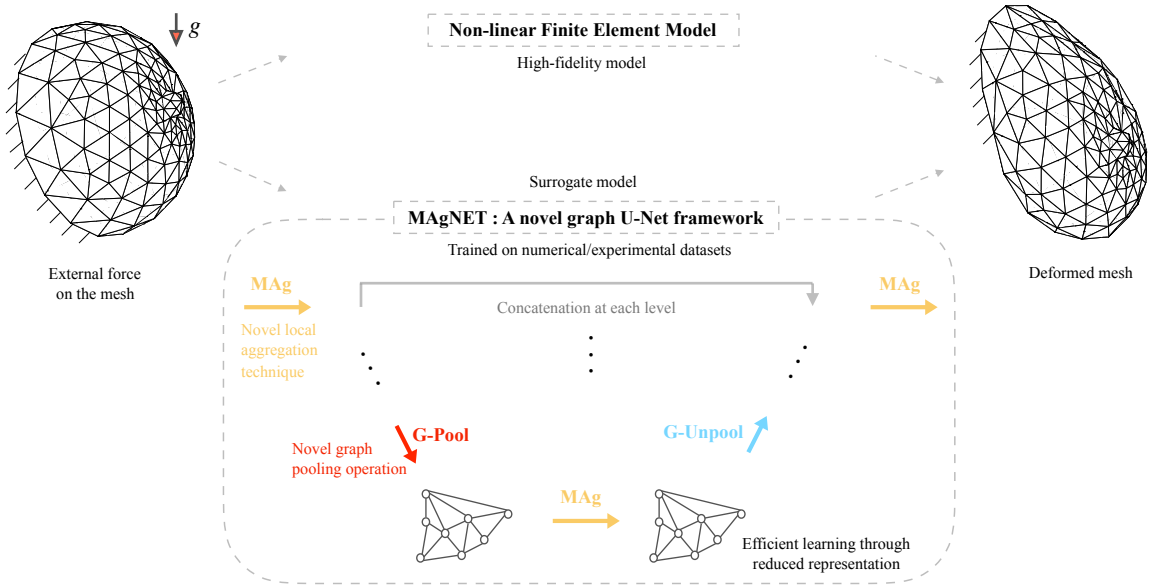


Figure 1: A novel graph U-Net neural network surrogate model for mesh-based simulations. MAgNET accurately captures non-linear FEM responses.

graph, which will be utilised later in this paper; and in Sections 2.3-2.4 we will specify a new graph Multi-channel Aggregation (MAg) layer, as well as new graph pooling/unpooling layers. Afterwards, in Section 2.5, we will provide an information-passing interpretation of the proposed Graph U-Net architecture. Finally, in Section 2.6 we will introduce a particular application of the framework to mesh-based datasets that are generated from FEM solutions of problems in hyper-elasticity.

### 2.1. MAgNET architecture overview

The MAgNET graph neural network architecture can be classified as a graph U-Net network and is an extension to the well-known class of convolution-based U-Net architectures (see [Ronneberger et al., 2015]). As such, the graph U-Net comprises of aggregation (‘convolution’), pooling, unpooling, and concatenation layers (see the schematics in Figure 2), which are here suitably adjusted to work with general (non-grid) topologies of inputs/outputs.

The graph U-Net architecture has two stages: encoding and decoding. In the encoding stage, first, we apply one or more aggregation (MAg) layers, which are analogues of convolution layers in non-graph U-Net networks. Next, we apply a single graph pooling layer, which is a particular contraction of the graph, and which downsamples (coarsens) the problem. This aggregation-pooling sequence is repeated several times to achieve the desired level of contraction (coarsening). At the coarsest level, the MAg aggregation is performed one or more times, after which the decoding stage begins, which is the opposite to the encoding stage. At each level of decoding, the graph unpooling layer is followed by one or more MAg

layers. At the upmost level, the last MAg layer is applied with linear activation to get the output.

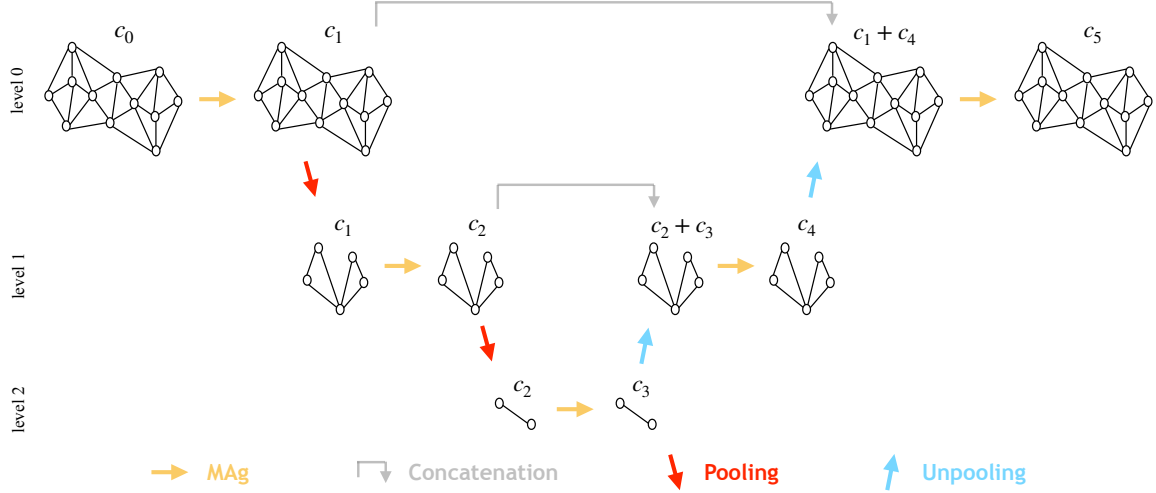


Figure 2: A schematic of Graph U-Net architecture for mesh based inputs. Colors indicate different types of layers.  $c_1, c_2, \dots, c_5$  stand for channel dimensions. Different arrows indicate different layers: the graph Multi-channel Aggregation (MAg) layer, the graph pooling/unpooling layers, and the concatenation layer.

More formally, the Graph U-Net network,  $\mathcal{G}$ , is constructed as follows. First, we set the input layer  $\mathbf{d}^0$  as a vector of  $N$  nodes, each of which being a vector of input values (also known as features or channels) of a constant length  $c_0$ . (Further on, we will refer to the features as the *channels*.) Next, we subsequently add layers,  $\mathbf{d}^l$ , to form a U-Net architecture. The subsequent layers,  $\mathbf{d}^{l-1}$  and  $\mathbf{d}^l$ , are linked by the following relationship

$$\mathbf{d}^l = \mathbf{T}^l(\mathbf{d}^{l-1}; \boldsymbol{\theta}^l), \quad (1)$$

where  $\boldsymbol{\theta}^l$  is a vector of trainable parameters (e.g., weights and biases,  $\boldsymbol{\theta}^l = \mathbf{k}^l \cup \mathbf{b}^l$ ), and  $\mathbf{T}^l(\cdot)$  is one of three already introduced transformations:  $\text{MAg}()$ ,  $\text{gPool}()$  or  $\text{gUnpool}()$ , which will be more precisely defined in the following sections. Additionally, we also consider remote concatenation links between respective layers from the encoding and decoding stages, see Fig. 2. The output layer,  $\mathbf{d}^L$ , is assumed to be of the same mesh format as the input layer but can have a different number of channels (features),  $c_L$ . Finally, we define the Graph U-Net network as a parameterized transformation

$$\mathcal{G}(\mathbf{d}^0, \boldsymbol{\theta}) = \mathbf{d}^L = \mathbf{T}^L(\mathbf{T}^{L-1}(\mathbf{T}^{L-2}(\dots); \boldsymbol{\theta}^{L-1}); \boldsymbol{\theta}^L), \quad (2)$$

where  $\boldsymbol{\theta} = \bigcup_{l=1}^L \{\boldsymbol{\theta}^l\}$  is a concatenated vector of network parameters.

The calibration of the Graph U-Net parameters is done through a supervised learning, by fitting a given known input-output training dataset. The training dataset,

$$\mathcal{D}_{\text{tr}} = \{(\mathbf{f}_1, \mathbf{u}_1), \dots, (\mathbf{f}_{M_{\text{tr}}}, \mathbf{u}_{M_{\text{tr}}})\}, \quad (3)$$

is in the mesh format, and the training is done by minimizing the following mean squared error loss function

$$\mathcal{L}(\mathcal{D}_{\text{tr}}, \boldsymbol{\theta}) = \frac{1}{M_{\text{tr}}} \sum_{m=1}^{M_{\text{tr}}} \|\mathcal{G}(\mathbf{f}_m, \boldsymbol{\theta}) - \mathbf{u}_m\|^2 \quad (4)$$

which gives the optimal parameters

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}_{\text{tr}}, \boldsymbol{\theta}). \quad (5)$$

## 2.2. Adjacency matrix of the mesh-based graph

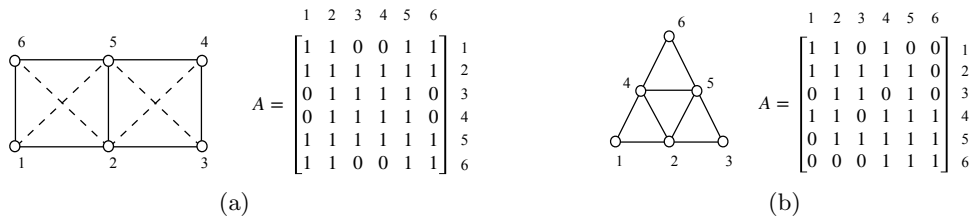


Figure 3: Adjacency matrices for the (a) square and (b) triangular meshes. The dashed lines in (a) represent additional edges that are added to the original mesh.

For the purpose of this work, we will focus on sparse graphs that derive from data that is spatially organised in the form of meshes. Those can be 1D, 2D, or higher-dimensional meshes, of an arbitrary connection topology (see Fig. 3 for examples of 2D meshes). The graph can be conveniently represented by a symmetric, square, Boolean adjacency matrix,  $\mathbf{A}$ , whose order is equal to the number of nodes in the original mesh. To simplify the further notation, all nodes (vertices) are self-connected (have loops), which results in having 1 on the diagonal of  $\mathbf{A}$ . This allows us to more easily express certain graph operations that are used in this work, for instance, the  $k$ -th power of a graph  $\mathbf{A}$ , and the selection of pooling sub-graphs that is presented in Section 2.4.

It is fairly straightforward to generate an adjacency matrix from an element connectivity matrix of the mesh. For that reason we will not discuss it in detail. The only point to be emphasized is that we make all nodes belonging to a given element mutually interconnected in the resulting graph (as they can be assumed to be strongly inter-related). We can visually represent it by adding more links as compared to a standard wire-frame visualization of finite-elements (see, e.g., the dashed lines in Fig. 3a).

*Remark:* In our work we do not consider any attributes for the edges of a graph. Therefore, the data is only represented through nodal features and node-node connections which are defined through the adjacency matrix  $\mathbf{A}$ .

## 2.3. Multi-channel Aggregation (MAg) layer

The proposed novel neural network layer, MAg, is a multi-channel local aggregation layer that can operate on graph-structured data. Its architecture is a direct extension to the

standard convolutional layer in CNNs, in which a shareable convolution window is used, making CNNs restricted to grid-structured data. In the MAg layer, instead, we propose to use fully-trainable local weighted aggregations (the so-called message passing scheme), where the aggregation neighborhood of a given node is prescribed through the graph connectivity (the adjacency matrix). As such, the scheme is very well suited for sparse graphs and can be directly applied to graphs that derive from arbitrary 2D or 3D meshes.

The use of multiple channels aims to improve the capabilities of the network to capture non-linearities. In the multi-channel version, each node represents a vector of values (features), which can be visualised as multiple layers (channels) of the same graph structure (see the schematics in Figure 4a). The transformation between the input- and output multi-channel graphs is realised by applying multiple MAg aggregations on vector data to produce respective multiple components of output vectors. Note that the input/output channels of the whole network have usually a certain meaning, and their sizes are fixed (e.g., three RGB channels of a color image at the input and a single channel of a segmented image at the output). The number of channels in the latent layers can be chosen arbitrarily, which is up to the choice of a designer of a particular graph U-Net architecture.

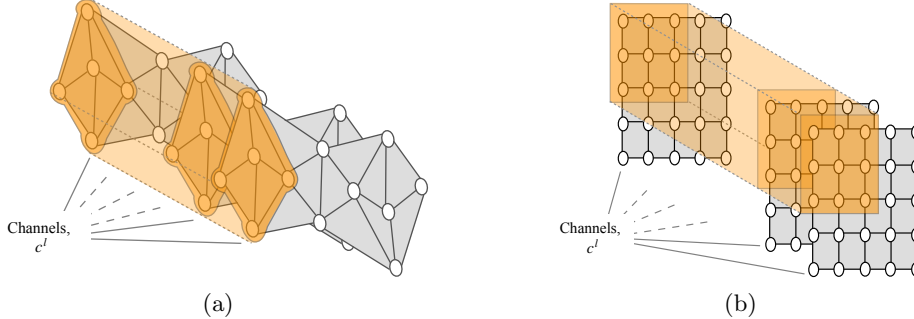


Figure 4: Local aggregation in MAg (a) works very similar to the filter application in CNN (b). However as opposed to CNN, MAg uses different set of weights at different spatial locations with heterogeneous window size. In CNN, a constant filter slides across the channel.

More formally, we will consider the MAg layer as a parameterized transformation between the input and output nodes, defined as

$$d_{i,\alpha}^{l+1} = \sigma(b_{i,\alpha}^{l+1} + \sum_{\beta=1}^{c^l} \sum_{j \in \mathcal{N}_i} k_{i,j,\alpha,\beta}^{l+1} d_{j,\beta}^l), \quad (6)$$

where  $\mathcal{N}_i = \{j \mid A_{ij} = 1\}$  is a set of neighbours of a node  $i$  to be aggregated,  $\alpha$  and  $\beta$  represent the output and input channels, respectively, while  $\mathbf{k}^{l+1}$  and  $\mathbf{b}^{l+1}$  are trainable weights and biases, respectively. In this multi-channel definition, for a given component,  $d_{i,\alpha}^{l+1}$ , of an output, a single aggregation is performed throughout the neighborhood,  $\mathcal{N}_i$ , and all the input channels,  $\beta \in \{1, \dots, c^l\}$ . The kernel parameters of MAg transformation,  $k_{i,j,\alpha,\beta}^{l+1}$ , are not shared, i.e, they can be independently trained for each aggregation window (note the free indexes  $i$  and  $\alpha$ ).

### 2.3.1. Comparison to existing graph aggregation/convolution layers

As already mentioned in the introduction, the very idea of generalisation of convolution layers to arbitrary graph structure is not new. In fact, various concepts have emerged so far, [Zhou et al., 2020][Chen et al., 2020], most of which are compatible with the U-Net framework proposed in the present work. Below, we will discuss several of them, introducing a unified notation that will facilitate a qualitative comparison with respect to the proposed MAg layer, see Table 1.

Layer	Transformation
GCN [Kipf and Welling, 2016]	$d_{i,\alpha}^{l+1} = \sigma\left(\sum_{\beta=1}^{c^l} w_{\alpha,\beta}^{l+1} \sum_{j \in \mathcal{N}_i} \frac{A_{i,j}}{\sum_{k \in \mathcal{N}_i} A_{i,k}} d_{j,\beta}^l\right)$
GAT [Veličković et al., 2017]	$d_{i,\alpha}^{l+1} = \sigma\left(\sum_{\beta=1}^{c^l} w_{t,\gamma,\beta}^{l+1} \sum_{j \in \mathcal{N}_i} \text{softmax}_j(\text{attn}(\mathbf{w}_t^{l+1} \mathbf{d}_i^l, \mathbf{w}_t^{l+1} \mathbf{d}_j^l, \boldsymbol{\theta}_t)) d_{j,\beta}^l\right)$
SemGCN [Zhao et al., 2019]	$d_{i,\alpha}^{l+1} = \sigma\left(\sum_{\beta=1}^{c^l} w_{\alpha,\beta}^{l+1} \sum_{j \in \mathcal{N}_i} \text{softmax}_j(k_{i,j,\alpha,\beta}^{l+1}) d_{j,\beta}^l\right)$
MAg [present work]	$d_{i,\alpha}^{l+1} = \sigma\left(\sum_{\beta=1}^{c^l} \sum_{j \in \mathcal{N}_i} k_{i,j,\alpha,\beta}^{l+1} d_{j,\beta}^l\right)$

Table 1: Comparison of the MAg layer with selected state of the art graph convolution layers (biases are omitted for the sake of brevity). In GAT formulation,  $\alpha = (t - 1)N_g + \gamma$ , which represents the stacking operation for the multi-head attention mechanism, where  $t \in (1, \dots, N_h^{l+1})$  is the attention head index, and  $\gamma \in (1, \dots, N_g)$  is the internal channel index (cardinality of each node in the layer  $l + 1$ ).

Graph neural network layers aim to utilize the information about assumed correlations in data, with the graph structure expressing those correlations. The general approach is to specify a suitable (possibly nonlinear) trainable local transformation that can aggregate the information from a node in consideration and its neighbours. (This aggregation is followed by a chosen activation function before being propagated to the next layer.) Such transformations form a wide class of, so-called, message passing schemes, and can combine shareable (independent of a node) and non-shareable (dependent on a node, i.e., independently-trainable) sets of parameters.

The simplest and most lightweight realisations of the graph aggregation/convolution layer concept only utilise shareable weights, see, e.g., the Graph Convolutional Network (GCN), [Kipf and Welling, 2016]. In those approaches, a non-trainable (arbitrary) weighted aggregation is performed prior to application of a shareable trainable operator – something completely opposite to our MAg layer, which is fully trainable. This enables to keep the number of trainable parameters low, which is achieved at the cost of relatively low capacity of such networks. This low capacity can not be straightforwardly increased by simply deepening the network because of the well-known over-smoothing phenomenon.

We will discuss two out of many available approaches to increase the capacity of graph neural networks. The first approach relies on the multi-head attention mechanism which allows to assign different importance to nodes in the neighbourhood, see, e.g, the Graph Attention Network (GAT), [Veličković et al., 2017]. In the attention mechanism, the weights used in local aggregation depend on input nodal features, which makes the concept qualita-

tively different from all approaches (including the MAg layer) that use input-independent aggregation weights. The second class of approaches resemble the MAg layer more closely. Particular notable examples of that approach are the Spatial-Temporal Graph Convolution Network (ST-GCN), [Yan et al., 2018], and the Semantic Graph Convolution Network (SemGCN), [Zhao et al., 2019], which have been introduced in the particular context of human pose recognition problem (the computer vision domain). The common features of MAg and SemGCN layers are the input-independent learnable weighted aggregation and the use of channels to increase the model capacity. The difference is that the MAg doesn't use a shared transformation matrix ( $\mathbf{w}$ ) nor the softmax normalisation – both used in the case of SemGCN.

To summarize, the proposed MAg layer relies on one of the most flexible message-passing schemes, with no shareable parameters. This promises a very high capacity of the MAg network. Also, as shown above, the proposed MAg layer is compatible with other graph convolution/aggregation layer concepts, and thus can be straightforwardly exchanged, if needed.

#### 2.4. Graph pooling- and unpooling layers

Pooling and unpooling are two fundamental operations allowing U-Nets to encode (compress) and decode (decompress) information, respectively, see Fig. 2. The *pooling* layers are composed of local contracting operations over the mesh-structured data, and are used to coarsen the data at the encoding part of the network. At the decoding part, the original refined mesh structure is restored by the *unpooling* layers (upsampling operations). In U-Nets, the unpooling layer is usually combined with the *concatenation* operation, which creates a direct link between the encoding and decoding part of the network (this will be explained later).

##### *Graph pooling*

In this work, we propose a novel clustering-based graph pooling layer that can be applied to arbitrary graph-structured data. It can be seen as an extension to the pooling layers known from CNN U-Nets that are limited to grid-structured data. In our graph pooling approach, we split the graph into disjoint cliques (fully-connected subgraphs), and perform the contraction of all the identified cliques (i.e., every clique is replaced by a vertex, and new edges represent formerly connected cliques), see Figure 5. The split into cliques is done statically, i.e., at the graph U-Net construction phase. In particular, the split does not depend on the input data.

Below, we will provide a more formal construction of the pooling layer. For a given input graph  $\mathbf{G}$  that is represented by the vertices  $\mathbf{S}$  and the connectivity matrix  $\mathbf{A}$ , we first generate an arbitrary set of  $\tilde{N}$  non-overlapping fully-connected subgraphs (cliques)  $\mathbf{G}_1, \mathbf{G}_2, \dots, \mathbf{G}_{\tilde{N}}$ , i.e.,

$$\mathbf{S} = \bigcup_{i=1}^{\tilde{N}} \mathbf{S}_i, \quad \forall_{\mathbf{s}_i} \forall_{j,k \in \mathbf{S}_i} A_{jk} = 1 \quad \text{and} \quad \forall_{i \neq j} \mathbf{S}_i \cap \mathbf{S}_j = \emptyset, \quad (7)$$

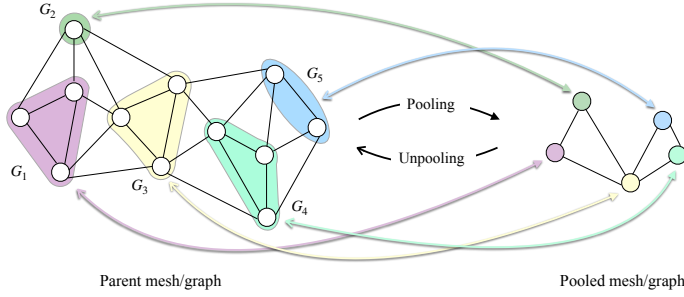


Figure 5: One arbitrary choice of non-overlapping subgraphs to create a pooled graph. Subgraphs  $\mathbf{G}_1, \dots, \mathbf{G}_5$  are represented with different colors and are generated by the Algorithm 1.

where the sets  $\mathbf{S}_i$  represent nodes of the respective subgraphs  $\mathbf{G}_i$ . The procedure to generate these subgraphs and the respective pooled adjacency matrix  $\tilde{\mathbf{A}}$  is described in Algorithm 1. The pooled graph  $\tilde{\mathbf{G}}$  is composed of vertexes  $\tilde{\mathbf{S}} = \{1, \dots, \tilde{N}\}$  with edges defined by the adjacency matrix  $\tilde{\mathbf{A}}$ . The pooling layer is described as:

$$d_{i,\beta}^{l+1} = \text{aggr}_{j \in \mathbf{S}_i} d_{j,\beta}^l, \quad (8)$$

where the 'aggr' operation can be a max/min/avg, etc. Note that graph pooling layers do not modify the number of channels, i.e., the pooling is performed individually per each channel of the input.

Graph pooling can be applied several times at the encoding part of the U-Net, e.g., see Fig. 2. For the purpose of future unpooling operations, after each pooling operation, we save the original graph,  $\mathbf{G}$ , the adjacency matrix,  $\mathbf{A}$ , and the pooling subgraphs,  $\mathbf{G}_i$ . After doing so, we substitute  $\mathbf{G} \leftarrow \tilde{\mathbf{G}}$ , and  $\mathbf{A} \leftarrow \tilde{\mathbf{A}}$ .

#### Graph unpooling

Structure-wise, the graph unpooling is a reverse operation to pooling. More precisely, the output graph of an unpooling layer will have the same topology as the input graph of the related pooling layer, see Fig. 5. The operation is defined via the previously saved subgraphs  $\mathbf{G}_j$  (with nodes  $\mathbf{S}_j$ ) as

$$d_{i,\beta}^{l+1} = d_{j,\beta}^l \quad \text{for } i \in \mathbf{S}_j, \quad (9)$$

and it simply replicates the features of a node  $j$  to the nodes specified by  $\mathbf{S}_j$ . As such, this operation is analogous to the related upsampling operation used in CNNs.

#### Graph unpooling + concatenation

Concatenations, also known as skipped connections, are characteristic to U-Net architectures. Thanks to them, the layers from the decoder part gain a direct access to features from the encoder part. Concatenations help to mitigate the issue of vanishing gradients, and add extra information that could have been lost due to the earlier downsampling (pooling).

In our case, the concatenations are always related to the respective pooling/unpooling operation pairs, see Fig. 2. It is done by stacking the output of an unpooling layer  $l$ , given

---

**Algorithm 1:** Generate a pooled graph from an arbitrary parent graph

---

**Input:**  $N \times N$  adjacency matrix,  $A$

**Result:** list of subgraphs,  $S$ ;  $\tilde{N} \times \tilde{N}$  pooled adjacency matrix,  $\tilde{A}$

```

 $S \leftarrow \{\}$  /* initialisation of the subgraph list */
 $P \leftarrow \{1, 2, \dots, N\}$  /* node indices of the parent graph */
 $A' \leftarrow A$  /* temporary copy of matrix  $A$  */
/* Loop for generating non-overlapping subgraphs,  $S$ , see Figure 5 */
while  $P \neq \text{null}$  do
     $p \in P$  /* randomly select a single node */
     $S_i \leftarrow \{p\}$  /* initialise subgraph */
     $\mathcal{N}_p \leftarrow \{m \neq p \mid A'[m, p] = 1\}$  /* nodes connected to selected node */
    for  $n$  in  $\mathcal{N}_p$  do
        if  $\forall m \in S_i \ A'[m, n] = 1$  then
             $S_i \leftarrow S_i \cup n$  /* append node to the subgraph */
        end
    end
     $P \leftarrow P \setminus S_i$  /* remove subgraph from parent graph */
     $\forall m \in S_i \ A'[m, :] \leftarrow 0; A'[:, m] \leftarrow 0$  /* remove subgraph from parent graph */
     $S \leftarrow S \cup S_i$  /* add subgraph to subgraphs list */
end
 $\tilde{N} = \text{sizeof}(S)$  /* number of pooled nodes = number of pooling subgraphs */
 $\tilde{A} \leftarrow \text{zeros}(\tilde{N}, \tilde{N})$  /* zero initialisation of pooled matrix */
/* Loop for constructing pooled adjacency matrix  $\tilde{A}$  from subgraphs  $S$  */
for  $r$  in  $\{1, 2, \dots, \tilde{N}\}$  do
    for  $c$  in  $\{1, 2, \dots, \tilde{N}\}$  do
        if  $\exists n \in S[r], m \in S[c] \mid A[n, m] = 1$  then
             $\tilde{A}[r, c] \leftarrow 1$  /* if  $S[r]$  and  $S[c]$  are connected by an edge */
        end
    end
end
end

```

---

by Equation (9), with the input of a respective pooling layer  $l'$ :

$$d_{i, c^l + \alpha}^{l+1} = d_{i, \alpha}^{l'}. \quad (10)$$

In the formula above,  $c^l$  is the number of channels in unpooling inputs. As the result, the total number of output channels of unpooling+concatenation is  $c^l + c^{l'}$ .

### 2.5. Information-passing interpretation of MAg and pooling layers

During a single forward pass of the MAg layer, the aggregation is performed locally for each individual node, i.e., each node of the graph will have an access to the aggregated



feature information from its adjacent nodes only, specified by the adjacency matrix,  $\mathbf{A}$ , see Equation (6). Therefore, the nodes that are not directly connected through  $\mathbf{A}$  do not exchange information at a single MAg operation (see Figure 6). Such long-distance exchange across the network is fundamental to allow the neural network model to express correlations between topologically distant input- and output nodes (e.g., how the output displacements at node C depend on the input loads at the node B, in Figure 6).

One way to handle this issue would be to apply the MAg layer several times as shown in Figure 6. However, in that case, the number of subsequent layers would be proportional to the diameter of the underlying graph, which could increase the number of training variables and the depth of the network, deteriorating its performance. A natural simple improvement, also utilised in the present paper, is to increase the support (neighbourhood) of the MAg operations. In the proposed framework, this can be straightforwardly done by using higher powers of the adjacency matrix, e.g.,  $\mathbf{A}^2$  or  $\mathbf{A}^3$ , instead of  $\mathbf{A}$ . This improvement alone, however, would still require the number of MAg layers to be proportional to the graph diameter.

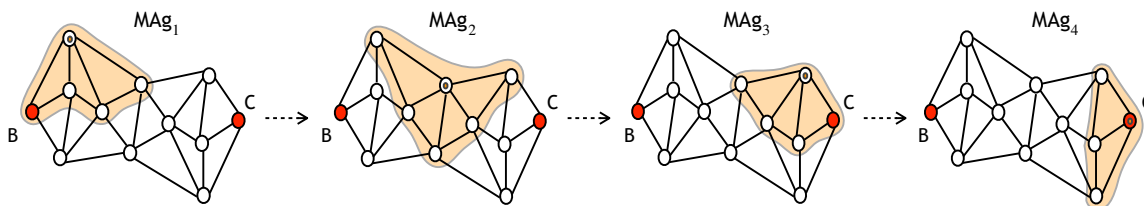


Figure 6: This 2D mesh requires at least 4 subsequent local aggregation operations (orange areas with center nodes marked by dots) to propagate the feature information from node B to the distant node C.

The above observations explain a natural motivation behind using the pooling/unpooling layers, and hence creating the U-Net architecture. The pooled graph can be seen as a reduced space representation of the parent graph, and each pooled node aggregates the feature information corresponding to multiple nodes of the parent graph, see Figure 7. The pooled graph is of a coarsened topology when compared to the parent graph, and this allows for the feature information exchange with a lower number of MAg layers. The pooling/unpooling layers can be nested, which provides an exponential reduction rate of the graphs' diameters.

*Remark:* Note that the pooling layer proposed in this work represents a clique-pooling approach in which the cliques are non-overlapping. This allows us to achieve a very good level of graph coarsening (contraction). It is unlike a similar clique-based strategy that has been recently proposed, [Luzhnica et al., 2019], in which the pooling cliques overlap, providing a much lower level of coarsening.

## 2.6. Application to FEM-based datasets

We will now focus on a particular graph structure of inputs/outputs that will be in a form of finite element mesh. Specifically, the MAgNET framework will be applied as a surrogate

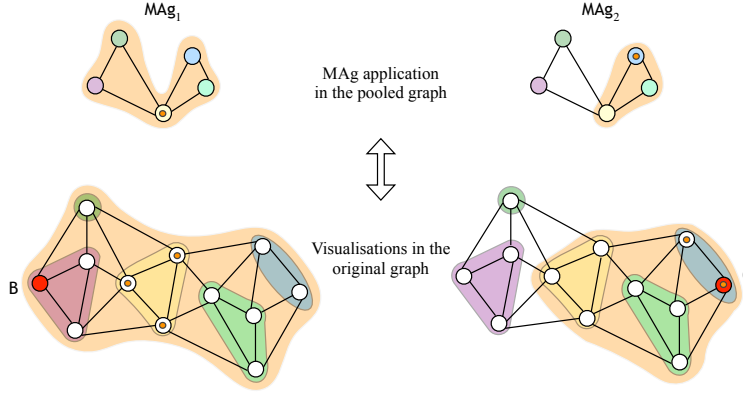


Figure 7: Visualisation of feature information exchange between nodes in the pooled graph. In the pooled space, only 2 MAg operations are sufficient to exchange feature information between spatially further located nodes in the original graph. The orange region shows the window of MAg operation.

model to a finite element model in large-deformation elasticity. The finite element model will be shortly introduced below.

We consider a boundary value problem expressed in the weak form over the domain  $\Omega$ :

$$\int_{\Omega} \mathbf{P}(\mathbf{F}(\mathbf{u})) \cdot \nabla \delta \mathbf{u} \, dV - \int_{\Omega} \rho \bar{\mathbf{b}} \cdot \delta \mathbf{u} \, dV - \int_{\Gamma_t} \bar{\mathbf{t}} \cdot \delta \mathbf{u} \, dS = 0 \quad \forall \delta \mathbf{u}, \quad (11)$$

where  $\mathbf{P}(\bullet)$  is the first Piola-Kirchhoff stress tensor,  $\bar{\mathbf{b}}$  are prescribed body forces,  $\bar{\mathbf{t}}$  are prescribed tractions on the Neumann's boundary  $\Gamma_N$ , while the solution  $\mathbf{u}$  and the variation  $\delta \mathbf{u}$  belong to appropriate functional spaces, with  $\mathbf{u} = \bar{\mathbf{u}}$  and  $\delta \mathbf{u} = \mathbf{0}$  on the Dirichlet boundary  $\Gamma_u$ . The hyperelastic constitutive relationship is expressed through the strain-energy density potential  $W(\mathbf{F})$  as

$$\mathbf{P}(\mathbf{F}) = \frac{\partial W(\mathbf{F})}{\partial \mathbf{F}}, \quad (12)$$

where the deformation gradient tensor  $\mathbf{F} = \mathbf{I} + \nabla \mathbf{u}$ .

For all the cases considered in the present work, the Neo-Hookean hyperelastic law with the following strain energy potential is used, see Simo and Taylor [1982],

$$W(\mathbf{F}) = \frac{\mu}{2}(I_c - 3 - 2 \ln J) + \frac{\lambda}{4}(J^2 - 1 - 2 \ln J), \quad (13)$$

where the invariants  $J$  and  $I_c$  are given in terms of deformation gradient  $\mathbf{F}$  as

$$J = \det(\mathbf{F}), \quad I_c = \text{tr}(\mathbf{F}^T \mathbf{F}), \quad (14)$$

with  $\mu$  and  $\lambda$  being Lamé's parameters, which can be expressed in terms of Young's modulus,  $E$ , and Poisson's ratio,  $\nu$ , as

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}. \quad (15)$$

Note that one can use other forms of the volumetric part of the above potential, see Doll and Schweizerhof [2000], or other hyperelastic models, such as the Mooney-Rivlin and a more general class of Ogden models, see Ogden [2005].

Finite element discretisation transforms the weak form expressed by Eq.(11) into the system of non-linear equations

$$\mathbf{R}(\mathbf{u}; \mathbf{f}_{\text{ext}}) = \mathbf{f}_{\text{int}}(\mathbf{u}) - \mathbf{f}_{\text{ext}} = \mathbf{0}, \quad (16)$$

that expresses the balance between external and internal nodal forces. In this work, the vector of external forces,  $\mathbf{f}_{\text{ext}}$ , represents boundary conditions, which can be surface tractions or body forces. Given  $\mathbf{f}_{\text{ext}} = \mathbf{f}_m$ , the system is solved for an unknown vector  $\mathbf{u}$  with the Newton-Raphson scheme, giving as a result the solution  $\mathbf{u}_m$ . A pair  $(\mathbf{f}_m, \mathbf{u}_m)$  makes an element of the dataset  $\mathcal{D}$  introduced in Eq. (3), and the FE mesh that results from the FE discretization produces the adjacency matrix  $\mathbf{A}$  introduced in Section 2.2.

### 3. Results

In this section, we will study the performance of the proposed framework, and for that purpose, we use four benchmark problems. In Section 3.1, we give a detailed specification of the benchmark problems and the procedure for obtaining FEM-based datasets. In Section 3.2, we provide details of neural network architectures for each of the studied cases and will describe the training procedure. In Section 3.3, we study the predictions of neural network models by cross-validating results from MAgNET and CNN models, and by comparing them with the FEM results. Finally, in Section 3.4 we demonstrate the capabilities of the MAgNET framework to provide a surrogate model for the unstructured mesh cases.

#### 3.1. Generation of FEM based datasets

We consider four benchmark problems, see Fig. 8. Two of them, Fig. 8(a-b), utilise simple meshes, which makes it possible to assure structured (grid) inputs. They will be used to cross-validate between our MAgNET architecture and the standard CNN U-Net architecture. The other two examples, Fig. 8(c-d), are more complex and will serve us to demonstrate the applicability of MAgNET for general (unstructured) meshes. Each of those two groups consists of a 2D and a 3D problem, thanks to which the framework can be tested for four different finite element topologies: triangular, quadrilateral, tetrahedral, and hexahedral.

For all considered cases, we use the neo-Hookean material model, see Section 2.6, with material parameters provided in the Table 2. In order to generate training/testing datasets, for each discretized problem we individually specify a family of boundary conditions, as described below, see also schematics in Figure 8. For the cases shown in Figures 8(a-c), nodes on one side are fixed (Dirichlet boundary conditions), and only a single random nodal force is applied at a selected node in a prescribed region of interest (denoted by red line/surface in Figure 8(a-c)). For the remaining nodes, the external forces are set to  $\mathbf{0}$ . For

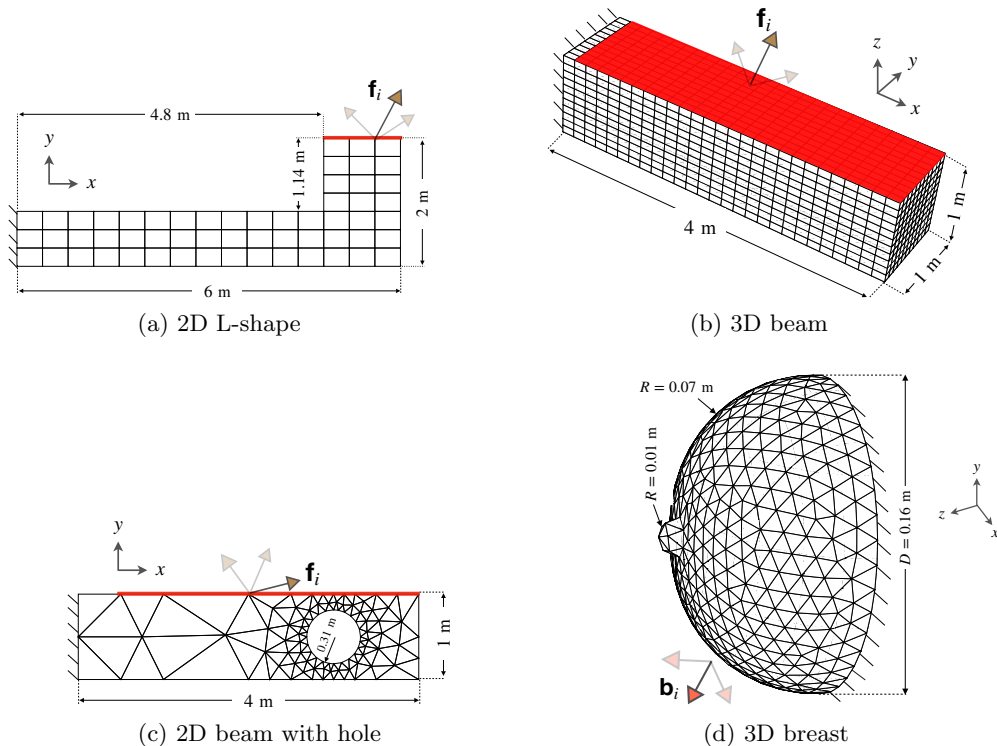


Figure 8: Schematics of four benchmark problems. (a) 2D L-shape geometry (quad mesh), (b) 3D beam geometry (hexahedron mesh), (c) 2D beam with hole geometry (triangular mesh), and (d) 3D breast geometry (tetrahedron mesh). In examples (a)-(c), single nodal loads are applied on the region of boundary indicated with red color. In example (d), only body forces are considered.

the case shown in Figure 8(d), the uniform body force density is prescribed (force per unit mass, with density  $\rho = 1000 \text{ kg/m}^3$ ). The body force field is integrated through element shape functions to obtain respective nodal forces that are used in datasets.

Problem (element topology)	Is structured?	Young's modulus, $E$ [Pa]	Poisson's ratio, $\nu$	Density, $\rho$ [kg/m <sup>3</sup> ]
a) 2D L-shape (quad)	No (Yes)	500	0.4	-
b) 3D beam (hexahedron)	Yes	500	0.4	-
c) 2D beam with hole (triangular)	No	500	0.3	-
d) 3D breast (tetrahedron)	No	800	0.4	1000

Table 2: Material properties used for the benchmark cases.

All the finite element computations were implemented and performed within the Ace-Gen/AceFem framework [Korelc, 2002]. For a given problem, for each loading case,  $i$ , the entire vector  $\mathbf{f}_{(i)}$  of external nodal forces and the vector  $\mathbf{u}_{(i)}$  of computed nodal displacements were saved, which allowed to generate the final training/testing dataset  $D = \{(\mathbf{f}_{(1)}, \mathbf{u}_{(1)}), \dots, (\mathbf{f}_{(M_{\text{tr}}+M_{\text{te}})}, \mathbf{u}_{(M_{\text{tr}}+M_{\text{te}})})\}$ . The datasets were randomly split into training sets,  $M_{\text{tr}}$  (95%), and testing sets,  $M_{\text{te}}$  (5%). The sizes of datasets and the distribution of force magnitudes are provided in Table 3.

Problem		N.of FEM DOFs ( $\mathcal{F}$ )	Range (External forces/ body force density)	Dataset size $M_{tr} + M_{te}$	samples per node
a)	2D L-shape	160	$f_x, f_y = -1$ to $1$ N	$3800 + 200$	1000
b)	3D beam	12096	$f_x, f_y, f_z = -2$ to $2$ N	$33858 + 1782$	110
c)	2D beam (hole)	198	$f_x, f_y = -5$ to $5$ N	$4560 + 240$	400
d)	3D breast	3105	$b_x, b_y = -6$ to $6$ N/kg , $b_z = -3$ to $3$ N/kg	$7600 + 400$	-

Table 3: Specification of FE-based datasets. For cases (a-c), the external force is applied at a selected node, and for case (d), external body forces are applied. The magnitudes of forces are randomly sampled from the multivariate uniform distribution, with ranges specified in the table. For cases (a-c), multiple samples per node are generated, for all nodes in the prescribed area of interest.

### 3.2. Design, implementation and training of neural network models

The implementation of the layers and mechanisms of the MAgNET framework described in Section 2 and of CNN U-Net framework introduced in [Deshpande et al., 2022] has been performed within the TensorFlow libraries. We use them to build and train deep neural network models for the cases described in Section 3.1. Table 4 outlines individual properties of the network architectures implemented in this work. The codes and datasets are publicly available open source [Deshpande et al., 2023], which makes it possible for other researchers to reproduce the present results and also to extend our frameworks to new cases/problems.

Example	Network type	(N. of poolings, N. of MAg/conv. layers per level, window size)	N. of channels per level	N. of parameters
2D L-shape	MAgNET	(3, 2, $A^2$ )	16, 32, 64, 128	$\sim 4$ E+6
	CNN U-Net	(2, 2, $3 \times 3$ )	64, 128, 512	
3D beam	MAgNET	(5, 1, $A^2$ )	3, 3, 3, 12, 24, 48	$\sim 75$ E+6
	CNN U-Net	(4, 2, $3 \times 3 \times 3$ )	256, 256, 256, 512, 512	
2D beam (hole)	MAgNET	(3, 2, $A^2$ )	8, 16, 32, 64	$\sim 2$ E+6
3D breast	MAgNET	(4, 1, $A^2$ )	6, 12, 12, 24, 48	$\sim 19$ E+6

Table 4: Neural network architectures implemented in this work. The leaky ReLU activation function is used in all MAgNET cases, while ReLU activation is used for CNN cases. For the last layers, the linear activation function is always applied.

To provide a complete understanding of the neural network architectures listed in Table 4, we will now delve into the details of the MAgNET architecture used for the 2D L-shape example. Its schematics is shown in Figure 9. As indicated in the third column in Table 4, it is a three-level graph U-Net architecture with two MAg operations at each level. The fourth column specifies the number of channels utilized for the MAg operations at each level of the graph U-Net. The forward pass starts with the input mesh (2D), to which the MAg layer is applied twice (with 16 output channels). This is followed by the graph pooling layer, which coarsens the mesh and transitions to the next level of the U-Net (from zeroth to the first level). This process repeats twice, with the first and second levels of the graph U-Net having MAg layers with 32 and 64 output channels, respectively, leading to the coarsest third level of the U-Net. At this level, two MAg layers (with 128 output channels) are applied. In

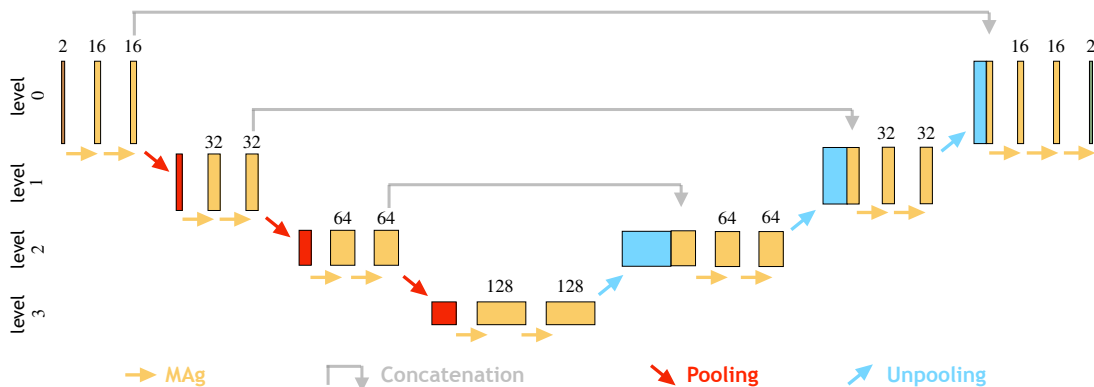


Figure 9: MAgNET architecture used for the 2D L-shape example.

the subsequent decoding phase, the graph unpooling layer is employed with the concurrent concatenation operation, and followed by two MAg operations (with 64 output channels). This upsampling sequence repeats twice with the use of 32- and 16-channel MAg layers. Finally, a single MAg layer (with 2 output channels) is applied, using a linear activation to produce the desired output mesh (the displacement mesh must have the same structure as the input mesh of forces). It is worth noting that analogous architectures of CNN U-Net networks are similar, with the only distinction being the use of convolution layers in place of MAg layers and CNN U-Net max poolings instead of graph poolings.

As demonstrated in Table 4, both 2D L-shape and 3D beam examples have been modeled utilizing both MAgNET and CNN U-Net architectures. These networks were designed to have a similar number of trainable parameters, thus facilitating a fair comparison of their fitting capabilities. The number of parameters in both types of networks is controlled by having a higher number of channels in the CNN U-Net architecture compared to its corresponding MAgNET architecture. This difference in the number of channels is attributed to the convolution operators in the CNN architecture sharing parameters across a layer, which may necessitate a larger number of channels to ensure an optimal fit, while the aggregation operators in the MAg layer use individual weights per aggregation window, allowing for more flexible fitting across the mesh with a smaller number of channels. However, caution must be exercised when selecting the number of channels, as setting it too low can result in increased prediction errors (as seen in Figure 15).

The number of neural network levels (pooling operations) and the size of convolution/aggregation windows have been adjusted on a case-by-case basis to obtain the desired fitting capabilities while keeping the number of trainable parameters low and comparable between the respective CNN U-Net and MAgNET models. The fitting capabilities heavily depend on the successful propagation of information from the input throughout the network. This can be compromised when the number of poolings or the window size is too small, as explained in Section 2.5. For this reason, a larger number of pooling operations is used for mesh graphs with larger diameters (e.g., the 3D cases in Table 4). Additionally, in the case of MAgNET models, a global optimization of graph pooling operations is performed to reduce

the number of nodes at the coarsest level. In this optimization, Algorithm 1 is run 1000 times with different random seeds, and the case with the least number of nodes at the lowest level is selected.

*Remark:* Note that the example presented in Fig. 8(a) utilizes a non-structured mesh. As such, it can not be directly used by the CNN U-Net model, and an additional preprocessing step needs to be done to make the input and output meshes structured. In this case, we apply zero padding to convert the L-shape mesh into a structured mesh, see Figure 10, which is then used for training with the CNN U-Net architecture. We do not need to do this preprocessing step for the MAgNET architecture.

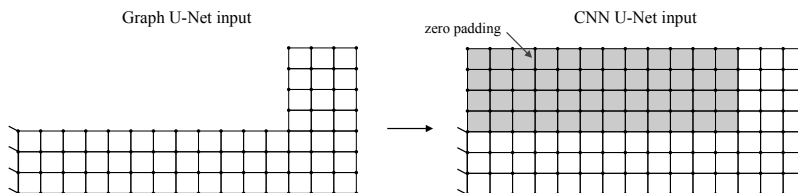


Figure 10: Zero-padding is applied to make the L-shape topology compatible with the CNN framework. The additional nodal values for inputs (forces) and outputs (displacements) are set as zero vectors.

The models presented in Table 4 were trained by minimizing the loss function, as described in Equation 4, using the datasets introduced in Section 3.1. The Adam optimizer, an extension of the stochastic gradient descent algorithm, was used for this purpose. A mini-batch size of 4 and an initial learning rate of  $1 \times 10^{-4}$ , with a linear decay to  $1 \times 10^{-6}$  during training, were employed. The number of epochs (i.e., iterations of the Adam optimizer) was manually tailored on a case-by-case basis to achieve low values of the loss function. An example of the training loss is provided in Figure 11. The network trainings were conducted using TensorFlow on a Tesla V100-SXM2 GPU at the HPC facilities of the University of Luxembourg, see [Varrette et al., 2014].

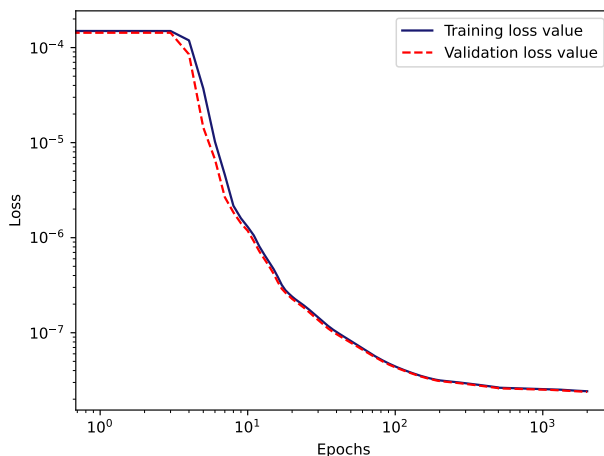


Figure 11: Training loss curve for the 3D breast MAgNET model.

### 3.3. Cross validation of CNN U-Net and MAgNET predictions

We are going to compare the predictions of MAgNET and CNN U-Net models for two problems with structured inputs/outputs that were introduced in Figure 8(a,b). Let us look at the individual examples with the highest nodal displacement magnitudes. In the 2D L-shape example, shown in Figure 12a, MAgNET predictions visually coincide with the reference FEM solution very well. This is quantitatively shown in Figures 12b and 12c, where the the  $L_2$  error field

$$\text{err}(\mathbf{X}) = \|u_{\text{FEM}}(\mathbf{X}) - u_{\text{pred}}(\mathbf{X})\|_2 \quad (17)$$

is presented for MAgNET and CNN U-Net, respectively, demonstrating low level of errors for both models. A similar tendency can be observed in the 3D beam case shown in Figure 13. Here, although the level of errors is relatively a bit higher than in the 2D example, the MAgNET and CNN U-Net perform similarly, which proves good capabilities of the proposed MAgNET model as compared to the CNN U-Net model.

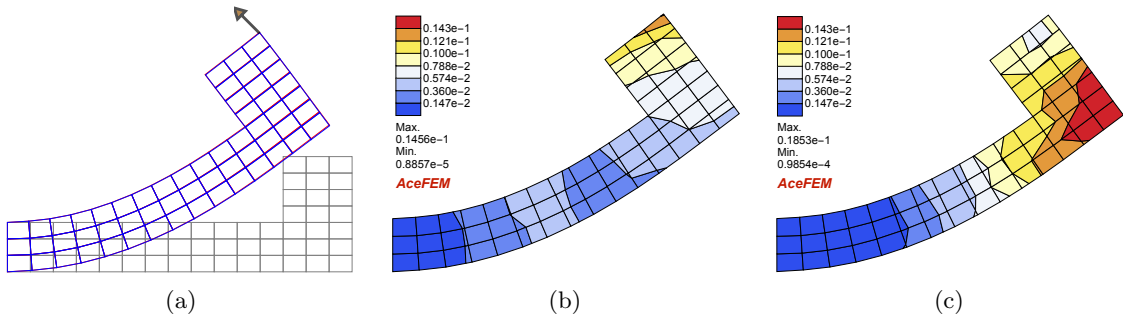


Figure 12: Deformation of 2D L-shape under point load  $(-0.93, 0.91)\text{N}$  on the corner node (a) Deformed mesh predicted using MAgNET (blue), for comparison FEM solution is presented (red) (b)  $L_2$  error of nodal displacements between MAgNET and FEM solution. The relative error for the corner node displacement using MAgNET is 0.5% (c)  $L_2$  error of nodal displacements between CNN U-Net and FEM solution. The relative error for the corner node displacement using CNN is 0.3%.

In the following, we will analyze and compare the performance of both models for all cases in the text datasets. For that purpose, we need aggregated error metrics. As an error metric for a single test example, we use the mean absolute error,

$$e_m = e(\mathbf{f}_m, \mathbf{u}_m) = \frac{1}{\mathcal{F}} \sum_{i=1}^{\mathcal{F}} |\mathcal{G}(\mathbf{f}_m)^i - \mathbf{u}_m^i|, \quad (18)$$

where the force-displacement pair  $(\mathbf{f}_m, \mathbf{u}_m)$  is an element of the test dataset

$$\mathcal{D}_{\text{te}} = \{(\mathbf{f}_{M_{\text{tr}}+1}, \mathbf{u}_{M_{\text{tr}}+1}), \dots, (\mathbf{f}_{M_{\text{tr}}+M_{\text{te}}}, \mathbf{u}_{M_{\text{tr}}+M_{\text{te}}})\}, \quad (19)$$

and  $\mathcal{F}$  is the number of dofs of the mesh. The metric  $e_m$  gives us the notion of error between an expected finite element solution,  $\mathbf{u}_m$ , and the prediction of the neural network,  $\mathcal{G}(\mathbf{f}_m)$ .



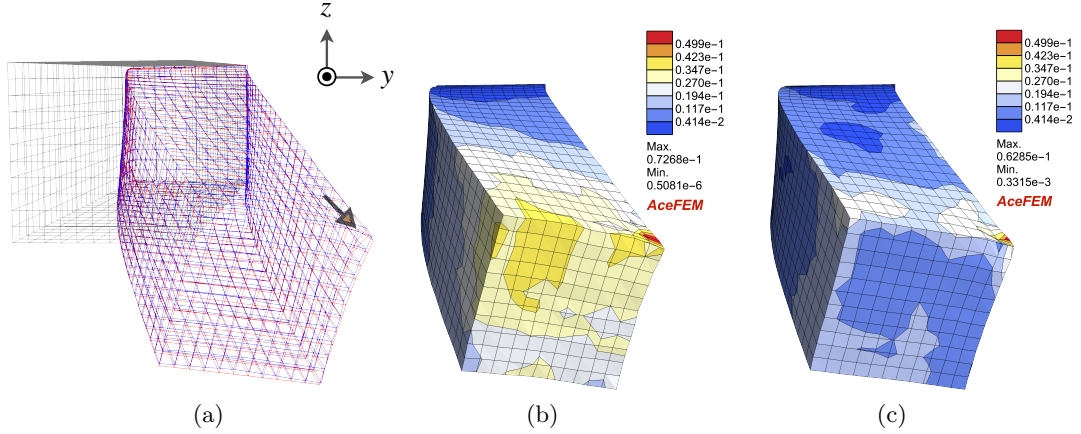


Figure 13: Deformation of the 3D beam under point load  $(-1.75, 1.31, -1.7)$ N on the second last node (a) Deformed mesh predicted using MAGNET (blue), for comparison FEM solution is presented (red) and undeformed mesh is represented by gray (b)  $L_2$  error of nodal displacements between MAGNET and FEM solution. The relative error in predicting displacement of the node of application of load using MAGNET is 4.4% (c)  $L_2$  error of nodal displacements between CNN U-Net and FEM solution. The relative error in predicting displacement of the node of application of load using CNN is 3.0%.

To analyze the overall quality of fitting, we define a single error metric over the entire test set as the average mean absolute error

$$\bar{e} = \frac{1}{M_{te}} \sum_{m=M_{tr}+1}^{M_{tr}+M_{te}} e_m, \quad (20)$$

with the corrected sample standard deviation (standard deviation of averaged errors) defined as

$$\sigma(e) = \sqrt{\frac{1}{M_{te} - 1} \sum_{m=M_{tr}+1}^{M_{tr}+M_{te}} (e_m - \bar{e})^2}. \quad (21)$$

Finally, in addition to that, we also use the maximum error per degree of freedom over the entire test set

$$e_{\max} = \max_{m,i} |\mathcal{G}(\mathbf{f}_m)^i - \mathbf{u}_m^i|. \quad (22)$$

Example	$M_{te}$	$\bar{e}$ [m]	$\sigma(e)$ [m]	$e_{\max}$ [m]
2D L-shape (MAGNET)	200	0.5 E-3	0.2 E-3	1.1 E-2
2D L-shape (CNN U-Net)		0.7 E-3	0.6 E-3	1.8 E-2
3D beam (MAGNET)	1782	0.8 E-3	0.7 E-3	7.7 E-2
3D beam (CNN U-Net)		0.7 E-3	0.5 E-3	5.4 E-2

Table 5: Error metrics for the structured mesh examples.  $M_{te}$  stands for the number of test examples, and  $\bar{e}$ ,  $\sigma(e)$ ,  $e_{\max}$  are error metrics defined by Equations (20)-(22).

The aggregated error metrics for the entire test sets of structured mesh examples obtained using MAGNET and CNN approaches are summarised in Table 5. The first observation is

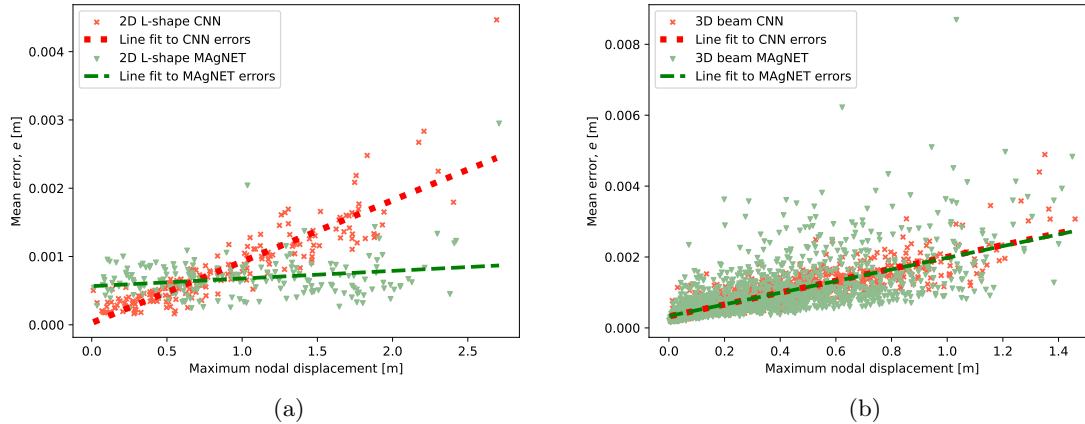


Figure 14: Mean absolute errors (see Equation (18)) as a function of maximum nodal displacements for all test examples for 2D L-shape and 3D beam cases for CNN U-NET and MAgNET networks.

that both MAgNET and CNN models exhibit similar prediction accuracy, demonstrating that the MAgNET architecture can achieve a comparable predictive capacity to the CNN U-Net architecture for a similar number of trainable parameters. The prediction errors, with respect to a characteristic length of 1m, fall below 0.1% for the average mean absolute error (Equation (20)), which is a promising result given the presence of geometric and constitutive nonlinearities. Additionally, we analyze the performance of the MAgNET model as a function of the maximum nodal displacement per test example. This dependency is visualized in Figure 14 for both benchmark examples. Although there is a general trend of increased errors for larger maximum displacement magnitudes, the sensitivity is low, and the errors remain small (the regression lines are  $e(d) \propto 1.0 \cdot d \cdot 10^{-4}$  (2D L-shape) and  $e(d) \propto 1.6 \cdot d \cdot 10^{-3}$  (3D beam) for MAgNET and  $e(d) \propto 7.0 \cdot d \cdot 10^{-4}$  (2D L-shape) and  $e(d) \propto 1.6 \cdot d \cdot 10^{-3}$  (3D beam)) for the CNN U-NET case.

As explained in Section 2.3, one can modulate the model capacity to capture non-linearities in the underlying data by modulating the number of channels in MAg and CNN layers. Importantly, the convolution windows are shareable in CNN architectures, whereas the aggregation windows in MAg architectures are independent. To this end, we expect CNN networks to require more channels than their respective MAgNET networks to achieve the same level of accuracy. We used this fact when designing CNN and MAgNET architectures in Section 3.2. To verify this hypothesis and demonstrate this effect, we trained five MAgNET and five CNN U-Net networks on the L-shape dataset with different numbers of channels. In all analyzed cases, we used 4-level MAgNET and 3-level CNN U-Net architectures, with two MAg/Conv layers per level, and with a constant number of channels at all levels. Figure 15 shows that there is indeed a strong dependency of accuracy on the number of channels for both analyzed network architectures. For a comparable number of trainable parameters, CNN U-Nets can use more channels than their respective MAgNETs, providing them with comparable predictive accuracy. We can also observe that too few channels

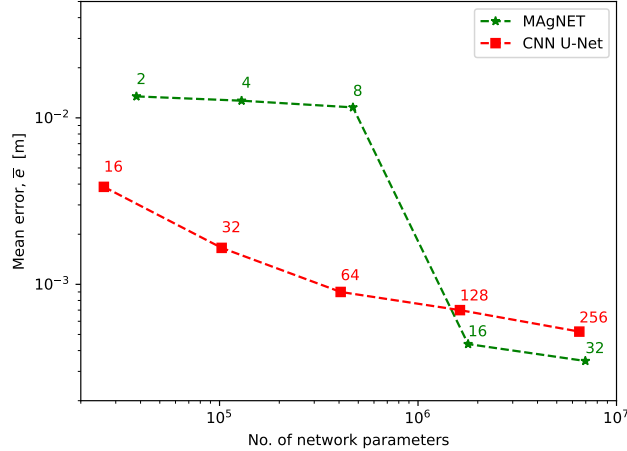


Figure 15: Average mean error over the test set for the L-shape case as the number of network parameters is changed by altering the number of channels used for MAgNET and CNN U-Net architectures. The numbers in the plot represent the number of channels used at each level in the MAgNET and CNN U-Net networks.

significantly reduce the fitting capabilities of both networks, with a step jump between the 8- and 16-channel case for MAgNET. For the two largest cases (16 and 32 channels for MAgNET and 128 and 256 channels for CNN U-Net), the accuracy of both architectures is comparable.

### 3.4. Predictions of MAgNET for general (unstructured) meshes

In Section 3.3, we demonstrated that the MAgNET architectures can achieve very good predictive capabilities for structured mesh cases, which was also cross-validated against respective CNN U-Net architectures. In this section, we aim to show that the high prediction accuracy of MAgNET can also be expected for unstructured mesh cases, which is the central point of the results section. We consider two cases: the first one is deformation under the application of point loads (the 2D beam with hole case), similar to the case of structured examples, and the second is deformation under body forces (the 3D breast case).

Let us first analyze individual examples. In Fig. 16 we present two particular loading cases of the 2D beam with hole. One of them is loaded at the tip, featuring the highest nodal displacement magnitude of all test cases, and the other one is loaded close to the hole, representing high local distortions. Similarly, for the three-dimensional problem, in Fig. 17 we show the case featuring the highest nodal displacement magnitude of all test cases. In all mentioned examples we can observe overall good accuracy when visually comparing MAgNET predictions with the respective FEM solutions. This can also be checked quantitatively by analyzing maximum displacement errors. In the cases of the 2D beam with hole, those errors are 1.4% and below when related to the characteristic length of 1m. In the case of 3D breast geometry, such relative maximum error is higher, reaching almost 3.1% (related to the breast diameter of 0.16m). Despite this fact, we can observe

that high local shape distortions are very well recovered. This property is more emphasized in Fig. 17c where one can additionally observe that also the Dirichlet boundary conditions are very well predicted, even though they were only introduced implicitly by training data.

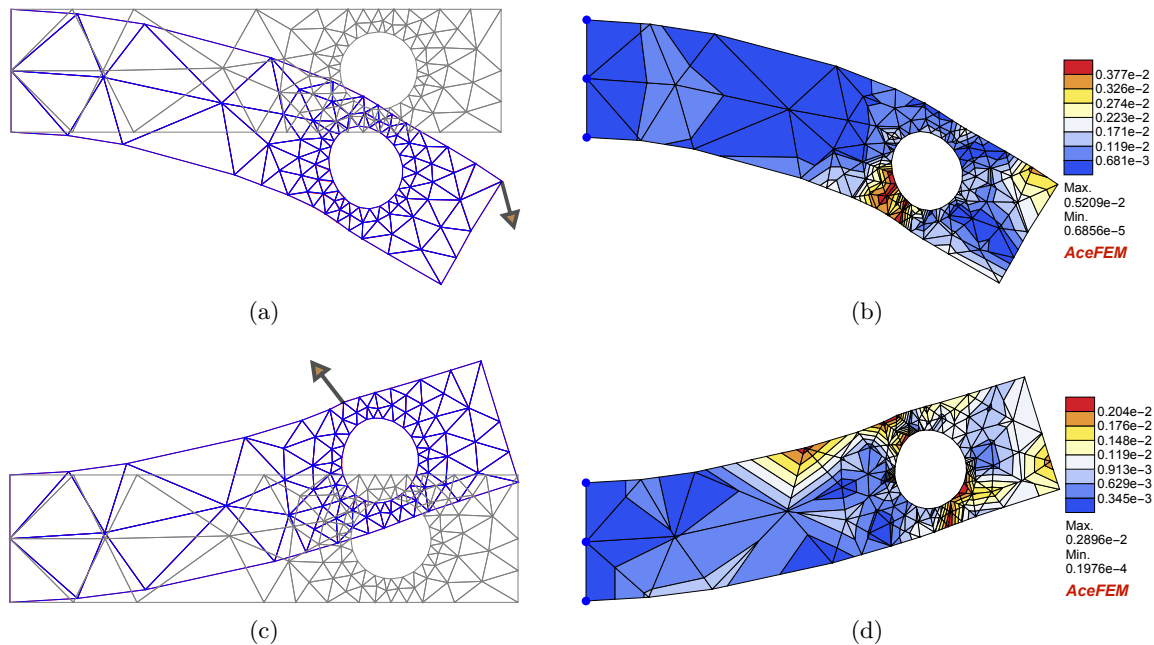


Figure 16: Deformation of the 2D beam under two different point loads (upper case:  $(1.28, -4.43)\text{N}$ , lower case:  $(-3.38, 4.04)\text{N}$ ). (a)&(c) Deformed meshes computed using MAGNET (blue) and FEM (red), with the undeformed configuration (gray). (b)&(d)  $L_2$  error of nodal displacements between MAGNET and FEM solutions.

The aggregated error metrics for the entire test sets are provided in Table 6. The maximum displacement errors over all test cases,  $e_{\max}$ , are at the levels observed for particular cases in Figures 16 and 17. At the same time, the average mean errors,  $\bar{e}$ , are at least an order of magnitude lower, which suggests that the errors close to maximum levels are not that often. The average mean errors are further analyzed in a case-by-case manner in Fig. 18, which is analogous to the analysis done for the structured cases in Fig. 14. Again, we plot the mean error  $e$ , of each test example as a function of the maximum nodal displacement. The regression lines  $e(d) \propto 5.0 \cdot d \cdot 10^{-4}$  (2D-beam) and  $e(d) \propto 8.0 \cdot d \cdot 10^{-4}$  (3D Breast) show low sensitivity of the MAGNET predictions to displacement magnitudes.

Example	$M_{te}$	$\bar{e}$ [m]	$\sigma(e)$ [m]	$e_{\max}$ [m]
2D beam (hole)	240	0.7 E-3	0.4 E-3	1.4 E-2
3D breast	400	8.9 E-5	3.1 E-5	5.1 E-3

Table 6: Error metrics for the unstructured mesh examples.  $M_{te}$  stands for the number of test examples, and  $\bar{e}$ ,  $\sigma(e)$ ,  $e_{\max}$  are error metrics defined in Section 3.3.

Above, we have demonstrated a good prediction accuracy of MAGNET within the test dataset (which is located in the interpolated domain). However, it is well known that

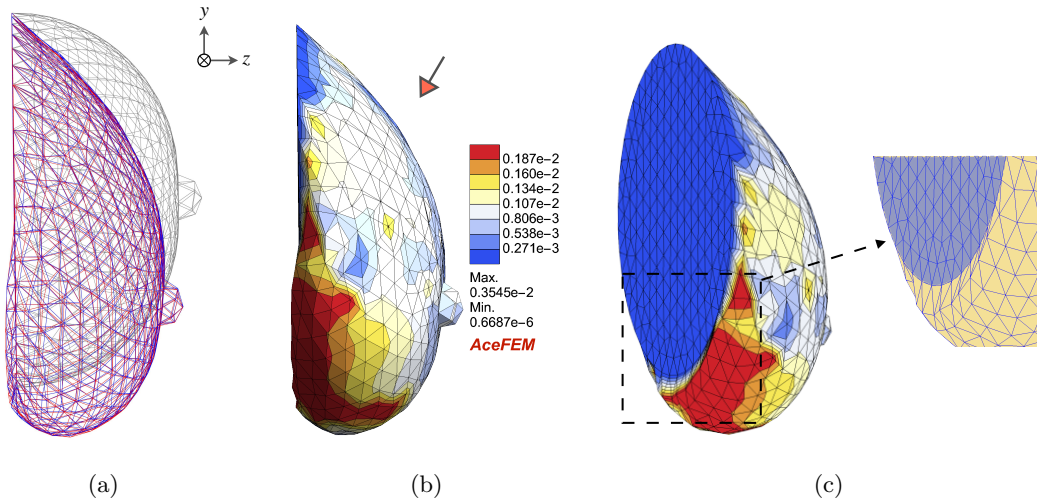


Figure 17: Deformation of the 3D breast geometry with force density of  $(-5.94, -5.23, -2.56)$  N/kg. (a) Deformed meshes computed using MAgNET (blue) and FEM (red), with the undeformed configuration (gray). (b)  $L_2$  error of nodal displacements between MAgNET and FEM solutions. (c) Titled view of the figure(b), MAgNET efficiently captures fixed boundary and nearby high non-linear deformations by learning implicitly from the data.

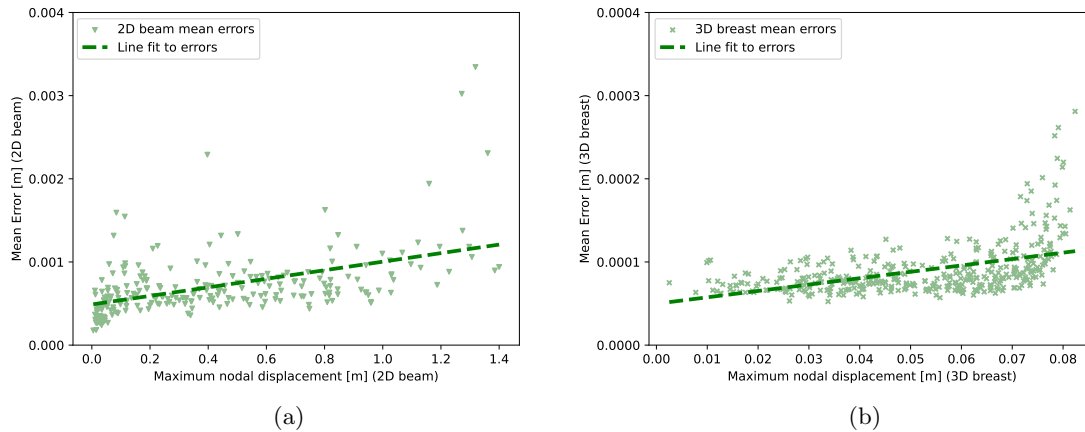


Figure 18: Mean absolute errors (see Equation (18)) as a function of maximum nodal displacements for all test examples (with unstructured meshes) predicted using MAgNET for (a) 2D beam with hole (b) 3D breast case.

this accuracy can gradually deteriorate when moving to the extrapolated region, see, e.g., Deshpande et al. [2022]. We are going to study this effect for MAgNET for a particular case that is based on the 3D breast geometry. As described in the Table 3, during the training, the  $b_z$  component of body force density is varied from -3 to 3 N/kg only. At the inference time, we applied  $b_z$  from -7 to 7 N/kg (keeping other components 0) to see how the predictions perform within and outside the training magnitudes. Figure 19a shows

that the error is fairly low and is not increasing within the training region and it increases rapidly outside, which confirms this well-known effect. Figures 19b and 19c show deformed meshes predicted for  $b_z = 5$  and  $b_z = 9$  N/kg, respectively, both outside the training data region. MAgNET is observed to give visually acceptable results although the accuracy of the framework decreases as we move away from the training data.

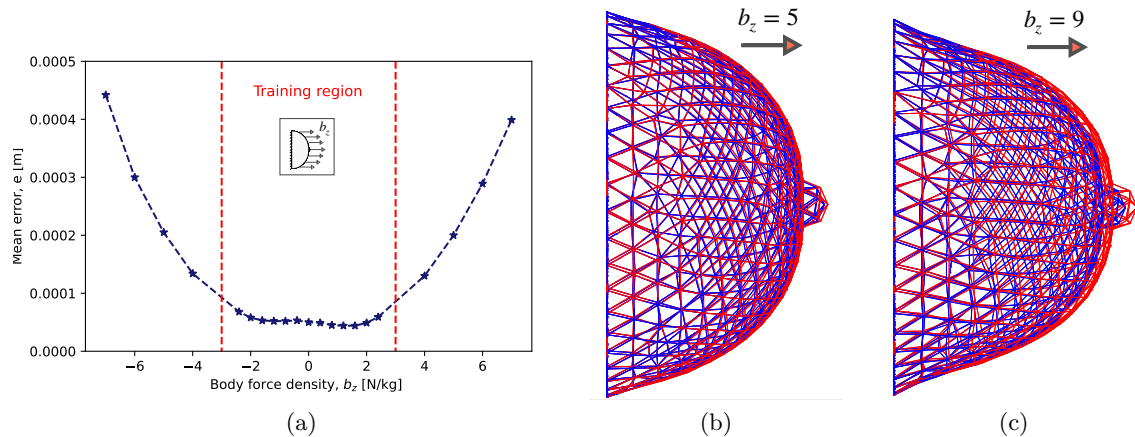


Figure 19: 3D Breast deformation under horizontal body force densities,  $(0, 0, b_z)$  N/kg. (a) Mean absolute error for testing cases in interpolated and extrapolated regions. The error increases rapidly in the extrapolated region while it remains low in the training (interpolated) region. (b)&(c) Visualisation of deformed meshes for force densities outside the training region computed using MAgNET (blue) and FEM solution (red).

### 3.4.1. A note on physics-informed errors

The proposed MAgNET framework has only been trained by minimizing the loss function for displacement errors, with no additional explicit information about the underlying physics/mechanics. As demonstrated earlier in this work, such training can provide very good accuracy in terms of predicted displacements. However, this accuracy is not of machine precision. To this end, a natural question arises: how far the displacement errors can violate physics? To answer that question, we are going to analyse some problem-based quantities of interest, such as residuals (balance of forces) or stresses, in comparison to the expected ground-truth results.

In Figure 20 we show nodal internal residual forces for the 2D beam with hole cases that we introduced earlier (compare Figure 16 for respective displacement errors). Ideally, the residual forces should be zero (the balance of forces), except for the boundary condition areas in which they should be exactly opposite to the reaction at the support and the applied external force. However, due to inaccuracies in displacements obtained from the MAgNET model, differences with respect to the ground true residuals can be noted. In Figure 20, we can observe the expected high residual forces in the areas where Dirichlet and Neumann boundary conditions are applied, however, also localised residual force spots are present in the fine mesh region around the hole. The magnitude of those errors in



the localised spots can go up to 20% of the maximal magnitude of applied forces. Also, when more closely analysing the residuals at boundary condition areas, it turns out that they do not fully match the respective FEM residuals. For instance, the relative error in residuals at the support in Figure 20a is almost 5%. When analysing the entire test set, we observed that the mean error in retrieving residuals at the Dirchilet boundary related to the maximum external force is 2.4%. A similar relative mean error value for retrieving the Neumann boundary residual is 14.6%. The higher error for Neumann boundary residual is attributed to high local non-linear deformations at the vicinity of the point of application of force. Though the displacement errors provoked by these non-linearities are not so high, they can result in high residual errors through the relatively high magnitude of element stiffness.

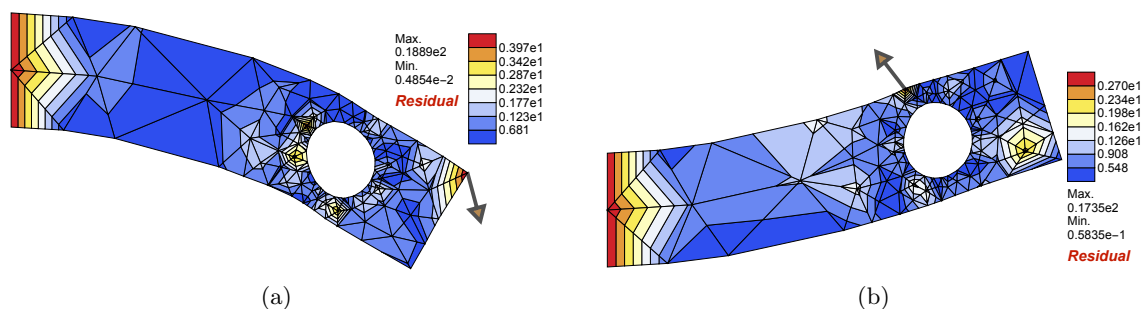


Figure 20: Nodal residual forces obtained using MAGNET solutions for the examples in Figure 16 (plotted on deformed meshes). The relative error for retrieving the total reaction force at the fixed interface is (a) 4.7% for the first example (b) 0.1% for the second example.

The errors observed in Figures 16 and 20 can have a direct impact on some application-dependent quantities of interest. As an example, in Figure 21, we present the field of von Mises stresses, which is a commonly used measure of shear stresses. We can observe that the MAGNET solution provides similar profiles of stresses as compared to respective FEM solutions, however, high localised errors are present at the fine mesh region (up to 30% of the reference FEM maximal von Mises stresses).

The above mentioned localised errors in residual forces, von Mises stresses and other relevant quantities of interest can be reduced by enriching the loss function with physics-informed terms. For instance, in the context of mesh-based force-displacement data, in [Odot et al., 2021] such enrichment has been introduced by scaling individual components of the loss function with the respective computed residual values, which proved to reduce residual errors. In [As’ad et al., 2022], the authors introduced an energy-based approach that provided purely physics-informed training for the Gauss-point stress-strain relationship, which allowed them to satisfy the expected frame indifference. Similar concepts of physics-informed loss functions can be seamlessly integrated into the MAGNET framework, which would convert it into a Physics Informed MAGNET.

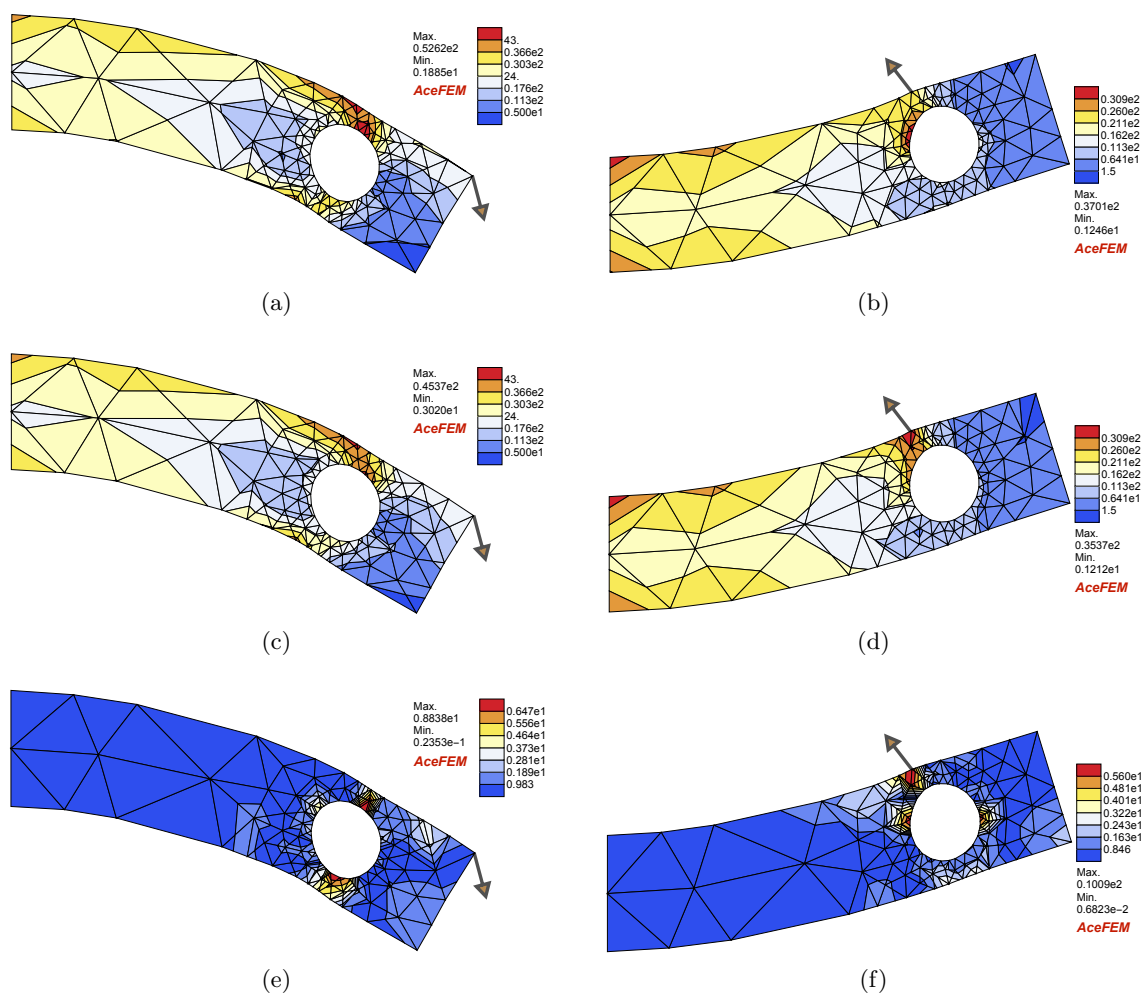


Figure 21: Von Mises stresses obtained for the two examples as in Figure 16 using (a)&(b) MAGNET solution (c)&(d) FEM solution. In (e)&(f) the absolute error between the MAGNET and FEM von Mises stresses is shown.

#### 4. Conclusion

In this work we proposed MAGNET, a novel framework for efficient supervised learning on graph-structured data using geometric deep learning. The framework comprises two neural network operations: MAg and graph pooling/unpooling layers, which together form a graph U-Net architecture capable of learning on large-dimension inputs/outputs. Notably, the MAGNET framework is not restricted to any particular input→output relationship or any specific mesh- or discretization scheme, making it superior to existing convolutional neural network architectures. MAGNET allows for arbitrary non-grid inputs/outputs, meaning it can handle arbitrary meshes and support complex geometries and local mesh refinements, making it suitable for a wide range of engineering applications.



We demonstrated and studied the capabilities of MAgNET in capturing nonlinear relationships in data. For this purpose, we conducted quantitative cross-validation of predictions made by MAgNET and the well-known convolutional U-Net architecture, both of which have been verified against the ground-truth results obtained with FEM. The benchmarks have proved that MAgNET has similar predictive capabilities as CNN U-Net for structured meshes, and it can also be extended to arbitrary meshes while preserving similar accuracy of predictions.

There are several natural directions for extending the capabilities of MAgNET. Firstly, the inclusion of underlying physics into the training process would enhance the overall performance of the framework. Although we have numerically demonstrated that the current data-based MAgNET framework can capture the underlying physics of the problem, we have also identified areas of increased errors, especially near regions of refined mesh. Integrating quantities of interest with the learning objective function would improve performance, and the implementation of a Physics Informed MAgNET is a very natural extension of the framework in the immediate future. Secondly, extending MAgNET to path-dependent processes, such as elasto-plasticity, is a promising direction, which would require keeping track of the evolution of state variables in a time-stepping manner. Recurrent neural network architecture, as described in [Mozaffar et al., 2019] or recent developments from our group [Vijayaraghavan et al., 2021], may be utilized for this purpose. Finally, there is a direct possibility to extend the proposed MAg layer to a Bayesian version, which would convert MAgNET into a probabilistic version. This could be achieved by performing local aggregations with probability distributions instead of discrete weights, similar to what we did in the case of CNNs in our previous work [Deshpande et al., 2022]. A Bayesian MAgNET would be capable of tracking uncertainties that are inherent to the choice of network architecture, as well as those inherent to real-world data.

We have made all the codes, datasets, and examples presented in this paper available open-access in the MAgNET repository at <https://github.com/saurabhdeshpande93/MAgNET>. Given the generality of MAgNET in supporting arbitrary non-linear relationships and arbitrary discretizations, we believe that the repository will provide a useful surrogate modeling framework for researchers and practitioners in various application areas across disciplines. We see it not only as a ready-to-use machine-learning library but also as a reference point and foundation for future developments and extensions in this emerging direction of research. The generality of MAgNET will enable the community to explore a range of new applications and modeling scenarios. By sharing our work, we hope to foster collaboration and advance the state-of-the-art in deep-learning surrogate modeling.

*Acknowledgements:*



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. Jakub Lengiewicz would like to acknowledge the support from EU Horizon 2020 Marie Skłodowska Curie Individual Fellowship *MOrPhEM* under Grant 800150. 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 Bordas and Jakub Lengiewicz are grateful for the support of the Fonds National de la Recherche Luxembourg FNR grant QuaC C20/MS/14782078. Stéphane 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.

## References

- As’ad, F., Avery, P., Farhat, C., 2022. A mechanics-informed artificial neural network approach in data-driven constitutive modeling. doi:10.2514/6.2022-0100.
- Aydin, R.C., Braeu, F.A., Cyron, C.J., 2019. General multi-fidelity framework for training artificial neural networks with computational models. *Frontiers in Materials* 6, 61.
- Bianchi, F.M., Grattarola, D., Alippi, C., 2019. Spectral clustering with graph neural networks for graph pooling. URL: <https://arxiv.org/abs/1907.00481>, doi:10.48550/ARXIV.1907.00481.
- Black, N., Najafi, A.R., 2022. Learning finite element convergence with the multi-fidelity graph neural network. *Computer Methods in Applied Mechanics and Engineering* 397, 115120. URL: <https://www.sciencedirect.com/science/article/pii/S004578252200305X>.
- Brenner, S.C., Scott, L.R., 2008. *Finite Element Multigrid Methods*. Springer New York, New York, NY. pp. 155–173. doi:10.1007/978-0-387-75934-0\_7.
- Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P., 2017. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 18–42. doi:10.1109/MSP.2017.2693418.
- Brunet, J.N., Mendizabal, A., Petit, A., Golse, N., Vibert, E., Cotin, S., 2019. Physics-based deep neural network for augmented reality during liver surgery, in: Shen, D., Liu, T., Peters, T.M., Staib, L.H., Essert, C., Zhou, S., Yap, P.T., Khan, A. (Eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, Springer International Publishing, Cham. pp. 137–145.
- Bui, H.P., Tomar, S., Courtecuisse, H., Cotin, S., Bordas, S.P.A., 2018. Real-time error control for surgical simulation. *IEEE Transactions on Biomedical Engineering* 65, 596–607. doi:10.1109/TBME.2017.2695587.

- Cai, C., Wang, D., Wang, Y., 2021. Graph coarsening with neural networks, in: International Conference on Learning Representations. URL: <https://openreview.net/forum?id=uxpzitPEooJ>.
- Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y., 2020. Simple and deep graph convolutional networks, in: III, H.D., Singh, A. (Eds.), Proceedings of the 37th International Conference on Machine Learning, PMLR. pp. 1725–1735. URL: <https://proceedings.mlr.press/v119/chen20v.html>.
- Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O., 2016. 3d u-net: learning dense volumetric segmentation from sparse annotation, in: International conference on medical image computing and computer-assisted intervention, Springer. pp. 424–432.
- Daniel, T., Casenave, F., Akkari, N., Ryckelynck, D., 2020. Model order reduction assisted by deep neural networks (rom-net). Advanced Modeling and Simulation in Engineering Sciences 7, 1–27.
- Deshpande, S., Bordas, S., Lengiewicz, J., 2023. Magnet: A graph U-Net architecture for mesh-based simulations [dataset]. doi:10.5281/zenodo.000000.
- Deshpande, S., Lengiewicz, J., Bordas, S.P., 2022. Probabilistic deep learning for real-time large deformation simulations. Computer Methods in Applied Mechanics and Engineering 398, 115307. URL: <https://www.sciencedirect.com/science/article/pii/S004578252200411X>, doi:<https://doi.org/10.1016/j.cma.2022.115307>.
- Doll, S., Schweizerhof, K., 2000. On the development of volumetric strain energy functions. J. Appl. Mech. 67, 17–21.
- Gao, H., Ji, S., 2019. Graph u-net. URL: <https://openreview.net/forum?id=HJePRoAct7>.
- Gao, H., Sun, L., Wang, J.X., 2021. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. Journal of Computational Physics 428, 110079. URL: <https://www.sciencedirect.com/science/article/pii/S0021999120308536>.
- Gao, H., Zahr, M.J., Wang, J.X., 2022. Physics-informed graph neural galerkin networks: A unified framework for solving pde-governed forward and inverse problems. Computer Methods in Applied Mechanics and Engineering 390, 114502. URL: <https://www.sciencedirect.com/science/article/pii/S0045782521007076>, doi:<https://doi.org/10.1016/j.cma.2021.114502>.
- Šarkić Glumac, A., Jadhav, O., Despotović, V., Blocken, B., Bordas, S.P., 2023. A multi-fidelity wind surface pressure assessment via machine learning: A high-rise building case. Building and Environment 234, 110135. URL: <https://www.sciencedirect.com/science/article/pii/S0360132323001622>, doi:<https://doi.org/10.1016/j.buildenv.2023.110135>.

- Goury, O., Duriez, C., 2018. Fast, generic, and reliable control and simulation of soft robots using model order reduction. *IEEE Transactions on Robotics* 34, 1565–1576.
- Henkes, A., Wessels, H., Mahnken, R., 2022. Physics informed neural networks for continuum micromechanics. *Computer Methods in Applied Mechanics and Engineering* 393, 114790. URL: <https://www.sciencedirect.com/science/article/pii/S0045782522001268>, doi:<https://doi.org/10.1016/j.cma.2022.114790>.
- Hennequin, R., Khlif, A., Voituret, F., Moussallam, M., 2020. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software* 5, 2154. doi:10.21105/joss.02154.
- Huang, D.Z., Xu, K., Farhat, C., Darve, E., 2020. Learning constitutive relations from indirect observations using deep neural networks. *Journal of Computational Physics* 416, 109491. URL: <https://www.sciencedirect.com/science/article/pii/S0021999120302655>, doi:<https://doi.org/10.1016/j.jcp.2020.109491>.
- Johnsen, S.F., Taylor, Z.A., Clarkson, M.J., Hipwell, J., Modat, M., Eiben, B., Han, L., Hu, Y., Mertzaniidou, T., Hawkes, D.J., et al., 2015. Niftysim: A gpu-based nonlinear finite element package for simulation of soft tissue biomechanics. *International journal of computer assisted radiology and surgery* 10, 1077–1095.
- Kipf, T.N., Welling, M., 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 .
- Klein, D.K., Ortigosa, R., Martínez-Frutos, J., Weeger, O., 2022. Finite electro-elasticity with physics-augmented neural networks. *Computer Methods in Applied Mechanics and Engineering* 400, 115501. URL: <https://www.sciencedirect.com/science/article/pii/S004578252200514X>, doi:<https://doi.org/10.1016/j.cma.2022.115501>.
- Kochkov, D., Smith, J.A., Alieva, A., Wang, Q., Brenner, M.P., Hoyer, S., 2021. Machine learning-accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences* 118, e2101784118. URL: <https://www.pnas.org/doi/abs/10.1073/pnas.2101784118>, doi:10.1073/pnas.2101784118, arXiv:<https://www.pnas.org/doi/pdf/10.1073/pnas.2101784118>.
- Korelc, J., 2002. Multi-language and multi-environment generation of nonlinear finite element codes. *Engineering with Computers* 18, 312–327. doi:10.1007/s003660200028.
- Krokos, V., Bordas, S.P.A., Kerfriden, P., 2022a. A graph-based probabilistic geometric deep learning framework with online physics-based corrections to predict the criticality of defects in porous materials. URL: <https://arxiv.org/abs/2205.06562>, doi:10.48550/ARXIV.2205.06562.
- Krokos, V., Bui Xuan, V., Bordas, S.P.A., Young, P., Kerfriden, P., 2022b. A bayesian multiscale cnn framework to predict local stress fields in structures with microscale features. *Computational Mechanics* 69, 733–766. doi:10.1007/s00466-021-02112-3.

- Lavigne, T., Mazier, A., Perney, A., Bordas, S., Hild, F., Lengiewicz, J., 2022. Digital volume correlation for large deformations of soft tissues: Pipeline and proof of concept for the application to breast ex vivo deformations. *Journal of the Mechanical Behavior of Biomedical Materials*, 105490 URL: <https://www.sciencedirect.com/science/article/pii/S1751616122003952>, doi:<https://doi.org/10.1016/j.jmbbm.2022.105490>.
- Le, T.T.H., Kang, H., Kim, H., 2022. Towards incompressible laminar flow estimation based on interpolated feature generation and deep learning. *Sustainability* 14. URL: <https://www.mdpi.com/2071-1050/14/19/11996>, doi:10.3390/su141911996.
- Lee, J., Lee, I., Kang, J., 2019. Self-attention graph pooling. URL: <https://arxiv.org/abs/1904.08082>, doi:10.48550/ARXIV.1904.08082.
- Lorente, D., Martínez-Martínez, F., Rupérez, M., Lago, M., Martínez-Sober, M., Escandell-Montero, P., Martínez-Martínez, J., Martínez-Sanchis, S., Serrano-López, A., Monserrat, C., Martín-Guerrero, J., 2017. A framework for modelling the biomechanical behaviour of the human liver during breathing in real time using machine learning. *Expert Systems with Applications* 71, 342–357. URL: <https://www.sciencedirect.com/science/article/pii/S0957417416306728>, doi:<https://doi.org/10.1016/j.eswa.2016.11.037>.
- Luzhnica, E., Day, B., Lio', P., 2019. Clique pooling for graph classification. URL: <https://arxiv.org/abs/1904.00374>, doi:10.48550/ARXIV.1904.00374.
- Mendizabal, A., Márquez-Neila, P., Cotin, S., 2019. Simulation of hyperelastic materials in real-time using deep learning. *Medical Image Analysis* 59, 101569. doi:10.1016/j.media.2019.101569.
- Mozaffar, M., Bostanabad, R., Chen, W., Ehmann, K., Cao, J., Bessa, M.A., 2019. Deep learning predicts path-dependent plasticity. *Proceedings of the National Academy of Sciences* 116, 26414–26420. URL: <https://doi.org/10.1073/pnas.1911815116>, doi:10.1073/pnas.1911815116.
- Obiols-Sales, O., Vishnu, A., Malaya, N., Chandramowliswharan, A., 2020. Cfdnet: A deep learning-based accelerator for fluid simulations, Association for Computing Machinery, New York, NY, USA. URL: <https://doi.org/10.1145/3392717.3392772>, doi:10.1145/3392717.3392772.
- Odot, A., Haferssas, R., Cotin, S., 2021. Deepphysics: a physics aware deep learning framework for real-time simulation. URL: <https://arxiv.org/abs/2109.09491>, doi:10.48550/ARXIV.2109.09491.
- Ogden, R.W., 2005. Non-linear elastic deformations. Dover Publications.
- Pant, P., Doshi, R., Bahl, P., Farimani, A.B., 2021. Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations. *Physics of Fluids* 33, 107101. URL: <https://doi.org/10.1063/1.50062546>, doi:10.1063/1.50062546.

- Pellicer-Valero, O.J., Rupérez, M.J., Martínez-Sanchis, S., Martín-Guerrero, J.D., 2020. Real-time biomechanical modeling of the liver using machine learning models trained on finite element method simulations. *Expert Systems with Applications* 143, 113083. URL: <https://www.sciencedirect.com/science/article/pii/S0957417419308000>, doi:<https://doi.org/10.1016/j.eswa.2019.113083>.
- Pfaff, T., Fortunato, M., Gonzalez, A., Battaglia, P., 2021. Learning mesh-based simulation with graph networks, in: *International Conference on Learning Representations*. URL: [https://openreview.net/forum?id=roNqYLO\\_XP](https://openreview.net/forum?id=roNqYLO_XP).
- Raissi, M., Perdikaris, P., Karniadakis, G., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* 378, 686–707. URL: <https://www.sciencedirect.com/science/article/pii/S0021999118307125>, doi:<https://doi.org/10.1016/j.jcp.2018.10.045>.
- Rao, C., Liu, Y., 2020. Three-dimensional convolutional neural network (3d-cnn) for heterogeneous material homogenization. *Computational Materials Science* 184, 109850. doi:[10.1016/j.commatsci.2020.109850](https://doi.org/10.1016/j.commatsci.2020.109850).
- Ren, X., Zhang, X., Chen, L., Zheng, X., Zhang, C., Guo, L., Yu, B., 2021. A causal u-net based neural beamforming network for real-time multi-channel speech enhancement, pp. 1832–1836. doi:[10.21437/Interspeech.2021-1457](https://doi.org/10.21437/Interspeech.2021-1457).
- Ronneberger, O., P.Fischer, Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, in: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Springer. pp. 234–241. URL: <http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a>. (available on arXiv:1505.04597 [cs.CV]).
- Runge, G., Wiese, M., Raatz, A., 2017. Fem-based training of artificial neural networks for modular soft robots, in: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 385–392. doi:[10.1109/ROBIO.2017.8324448](https://doi.org/10.1109/ROBIO.2017.8324448).
- Rus, D., Tolley, M.T., 2015. Design, fabrication and control of soft robots. *Nature* 521, 467–475.
- Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V., Guo, H., Hamdia, K., Zhuang, X., Rabczuk, T., 2020. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering* 362, 112790. URL: <https://www.sciencedirect.com/science/article/pii/S0045782519306826>, doi:<https://doi.org/10.1016/j.cma.2019.112790>.
- Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., Battaglia, P.W., 2020. Learning to simulate complex physics with graph networks. *Learning to simulate complex physics with graph networks* , 8459 – 8468 URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85094802982&partnerID=40&md5=57a515ee79e6915a2266fa3f2bc4870c>. cited by: 53.

- Simo, J., Taylor, R., 1982. Penalty function formulations for incompressible nonlinear elastostatics. *Computer Methods in Applied Mechanics and Engineering* 35, 107–118.
- Thakolkaran, P., Joshi, A., Zheng, Y., Flaschel, M., De Lorenzis, L., Kumar, S., 2022. Nn-euclid: Deep-learning hyperelasticity without stress data. *Journal of the Mechanics and Physics of Solids* 169, 105076. URL: <https://www.sciencedirect.com/science/article/pii/S0022509622002538>, doi:<https://doi.org/10.1016/j.jmps.2022.105076>.
- Varrette, S., Bouvry, P., Cartiaux, H., Georgatos, F., 2014. Management of an academic hpc cluster: The ul experience. doi:10.1109/HPCSim.2014.6903792.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y., 2017. Graph attention networks. URL: <https://arxiv.org/abs/1710.10903>, doi:10.48550/ARXIV.1710.10903.
- Vijayaraghavan, S., Wu, L., Noels, L., Bordas, S.P.A., Natarajan, S., Beex, L.A.A., 2021. Neural-network acceleration of projection-based model-order-reduction for finite plasticity: Application to RVEs. arXiv URL: <https://arxiv.org/abs/2109.07747>, doi:10.48550/ARXIV.2109.07747.
- Vlassis, N.N., Ma, R., Sun, W., 2020. Geometric deep learning for computational mechanics part i: anisotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering* 371, 113299. URL: <https://www.sciencedirect.com/science/article/pii/S0045782520304849>, doi:<https://doi.org/10.1016/j.cma.2020.113299>.
- Wang, F., Eljarrat, A., Müller, J., Henninen, T., Erni, R., Koch, C., 2020a. Multi-resolution convolutional neural networks for inverse problems. *Scientific Reports* 10, 5730. doi:10.1038/s41598-020-62484-z.
- Wang, F., Eljarrat, A., Müller, J., Henninen, T., Erni, R., Koch, C., 2020b. Multi-resolution convolutional neural networks for inverse problems. *Scientific Reports* 10, 5730. doi:10.1038/s41598-020-62484-z.
- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S., 2021. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 4–24. doi:10.1109/TNNLS.2020.2978386.
- Yan, S., Xiong, Y., Lin, D., 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. URL: <https://arxiv.org/abs/1801.07455>, doi:10.48550/ARXIV.1801.07455.
- Zhang, W., Li, D.S., Bui-Thanh, T., Sacks, M.S., 2022. Simulation of the 3d hyperelastic behavior of ventricular myocardium using a finite-element based neural-network approach. *Computer Methods in Applied Mechanics and Engineering* 394, 114871. URL: <https://www.sciencedirect.com/science/article/pii/S0045782522001724>, doi:<https://doi.org/10.1016/j.cma.2022.114871>.

- Zhao, L., Peng, X., Tian, Y., Kapadia, M., Metaxas, D.N., 2019. Semantic graph convolutional networks for 3d human pose regression, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE. URL: <https://doi.org/10.1109%2Fcvpr.2019.00354>, doi:10.1109/cvpr.2019.00354.
- Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M., 2020. Graph neural networks: A review of methods and applications. *AI Open* 1, 57–81. URL: <https://www.sciencedirect.com/science/article/pii/S2666651021000012>, doi:<https://doi.org/10.1016/j.aiopen.2021.01.001>.