

Bayesian graph neural networks for strain-based crack localization

C. Mylonas¹ G. Tsialiamanis² K. Worden² E. N. Chatzi¹

¹Institute of Structural Engineering, ETH Zürich,
Stefano-Francini-Platz 5, CH-8093 Zürich, Switzerland

²Dynamics Research Group, Department of Mechanical Engineering,
University of Sheffield Mappin Street, Sheffield S1 3JD, UK

Abstract

A common shortcoming of vibration-based damage localization techniques is that localized damages, i.e. small cracks, have a limited influence on the spectral characteristics of a structure. In contrast, even the smallest of defects, under particular loading conditions, cause localized strain concentrations with predictable spatial configuration. However, the effect of a small defect on strain decays quickly with distance from the defect, making strain-based localization rather challenging. In this work, an attempt is made to approximate, in a fully data-driven manner, the posterior distribution of a crack location, given arbitrary dynamic strain measurements at arbitrary discrete locations on a structure. The proposed technique leverages Graph Neural Networks (GNNs) and recent developments in scalable learning for Bayesian neural networks. The technique is demonstrated on the problem of inferring the position of an unknown crack via patterns of dynamic strain field measurements at discrete locations. The dataset consists of simulations of a hollow tube under random time-dependent excitations with randomly sampled crack geometry and orientation.

Keywords: GraphNets, Strain-based crack localization, Bayesian Graph Neural Network

1 Introduction

When the region around a defect is loaded, strain concentration occurs. For different local loading conditions, depending on the geometric configuration of the defect, different strain concentration patterns arise. The technique proposed in this work, allows for strain-based localization of cracks with learned, rather than hand-engineered, features. In contrast to the approaches based on 2D Convolutional Neural Networks (CNNs) already in the literature [1], the approach presented herein can effectively deal with arbitrary (potentially sparse) strain sensor positions, and not only sensing positions arranged in a grid as CNN-based computation dictates. This feature is achieved by employing a message-passing Graph Neural Network (GNN) [2, 3]. The GNN is used for recognising spatial strain patterns, and learns these patterns directly from data. These patterns depend non-linearly on the sensor position relative to the crack and the sensor position in the structure.

In essence, as CNNs perform local aggregation defined by a spatial 2D grid, message-passing graph neural networks allow for local aggregation on the neighborhood of the nodes of an arbitrary graph. In contrast to other works in the literature [4, 5], a model that predicts directly the crack position from data is learned and no optimization is performed during prediction time¹.

Training Data And Strain-PCA contrasting: In this study, the dynamics of 450 hollow tubes with randomly-positioned and rotated ellipsoidal defects were simulated with the Finite Element Method under random white-noise

¹This work formed part of the PhD thesis of the first author [6].

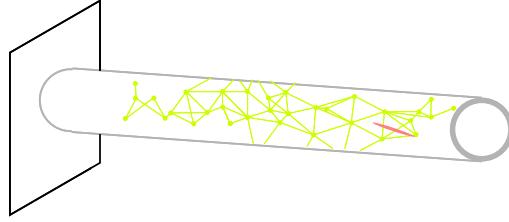


Figure 1: Schematic of the construction of a strain sensing graph. Each node corresponds to a full strain-sensor. The neighborhood of each sensor is defined by the sensors closer to it. An ellipsoidal crack is depicted in red.

excitations in three dimensions, different for every run. The full strain tensors on a 150×150 grid of values on the surface of the tube were extracted. All simulated tubes are 10m in length and 1m in diameter.

A basis is first obtained from a Principal Components Analysis (PCA) of strain snapshots of a crack-free structure. In the presented results, 50 components of the PCA basis are retained (variance explained 88.3%). A separate PCA for each component of the strain tensor and the strain-invariants is learned. The strain fields corresponding to structures that contain a defect are pre-processed by projecting their strain fields to the corresponding PCA-derived bases. The projected strains are removed from the observed strains. Assuming a linear superposition of the strain related to the defect and the strain of the defect-free structure, the strain associated with the defect should appear in the residuals of the projection. Indeed, this effect is demonstrated in the right-most plot of Figure 2. During prediction, only a sparse set of sensors is expected to be available. Therefore, the estimates of the strain field are going to be less accurate compared to the case where the full strain field is available. The projection using a sparse mask of strain sensors, is equivalent to performing a simple linear regression with the PCA bases. The PCA approximation of $\hat{\varepsilon}$, for a strain component ε based on a sparse set of sensors at i positions $S : \{s_1, \dots, s_i\}$, is computed from

$$\begin{aligned} \bar{\Psi}_\varepsilon &\xleftarrow{S} \Psi_\varepsilon \\ \hat{c} &= (\bar{\Psi}_\varepsilon^T \bar{\Psi}_\varepsilon)^{-1} \bar{\Psi}_\varepsilon^T \bar{\varepsilon} \\ \hat{\varepsilon} &= \bar{\Psi}_\varepsilon \hat{c} \\ r_\varepsilon^{PCA} &= \hat{\varepsilon} - \varepsilon \end{aligned}$$

where Ψ_ε is the matrix of PCA components, $\bar{\Psi}_\varepsilon$ is the matrix of PCA components, sparsified at the sensor positions, \hat{c} is a vector of coefficients and r_ε^{PCA} is the PCA-computed residual for that strain component. In what follows, for notational convenience the PCA-derived strain residuals $r_{\varepsilon_{ij}}^{PCA}$ are referred to simply as strains ε_{ij} .

A representative example of the quality of the approximation for the investigated structure, and for the crack geometries used is depicted in Figure 2.

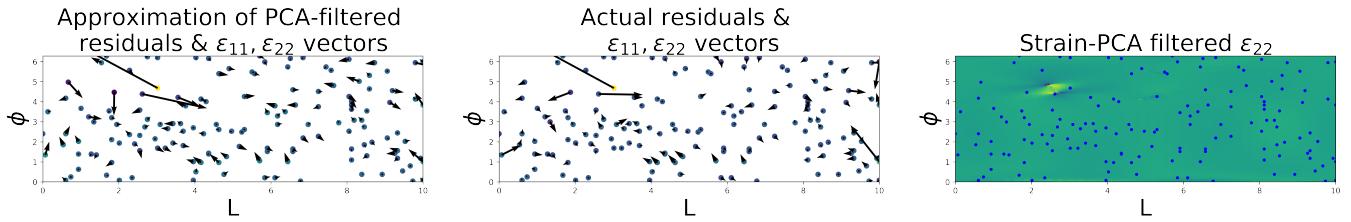


Figure 2: Contrasting sparse measurements using strain field PCA. The positions on the plots correspond to length-wise and angular cylinder coordinates. The left-most plot is an approximation of the defect-dependent residual based on the projection on a limited number of sensors with $\bar{\Psi}_\varepsilon$. The middle plot is the actual residual based on a projection using the dense measurements and full PCA components Ψ_ε . The rightmost plot is the full residual strain field for ε_{22} for a time-instant where ε_{22} reveals the position of the strain.

2 Message-Passing Graph Neural Networks

A directed attributed graph $G : (V, E)$, (referred to in the following simply as “graph”) is defined as a set of vertices (nodes) $V : \{\mathbf{v}_1, \dots, \mathbf{v}_{N^e}\}$ and a set of edges $E : \{(\mathbf{e}_1, s_1, r_1), \dots, (\mathbf{e}_{N^e}, s_{N^e}, r_{N^e})\}$. Nodes contain attributes, and edges are described by an attribute \mathbf{e}_i , and sender s_i and receiver r_i indices. Indices s_i, r_i are integers representing the sender and receiver nodes. *Attributes* may be discrete or continuous vectors or tensors. In [3] the *GraphNet* (GN) computational block is proposed, which is a generalisation of several approaches to computation using attributed graphs, with a focus on implementing *inductive biases*. GNs are algorithms that operate on graphs and return graphs. The attributes of the vertices and edges, parametrize how the entities in the graph interact. In addition, graph-level (global) attributes may exist, which depend on the aggregation of all vertex and node information. More concretely, for the problem considered, the strain sensors have the role of vertices, their relative positions are edge features (which dictate which nodes will interact to yield a prediction), and the position and orientation of the crack is the global variable which needs to be inferred from the sensor observations.

A GraphNet computational block consists of

- an edge function $f^e : \mathbf{e} \rightarrow \mathbf{e}'$ which parametrizes what information is passed along an edge ;
- an edge message aggregation function $\rho^{e \rightarrow v}$, which is a permutation invariant function that summarizes the messages to be passed according to the graph connectivity;
- a vertex (node) function that transforms the vertex attributes $f^v : \mathbf{v} \rightarrow \mathbf{v}'$, optionally using the aggregated incoming messages, the node attributes and the global attributes, and finally;
- a global function $f^u : \mathbf{u} \rightarrow \mathbf{u}'$; which uses global attributes. Node and edge aggregated information which are aggregated by two additional aggregators:
- the edge-to-global aggregator $\rho^{e \rightarrow u}$ and
- the node-to-global aggregator $\rho^{v \rightarrow u}$.

The interaction of vertices in GraphNets occurs through a process that resembles *message passing* [2], with a learned message passing function. In addition to the message-passing function, transformations of the edge and node attributes are performed, for instance by a Multi-Layer Perceptron (MLP). Graph Neural Networks (GNNs) are GNs that use MLPs for one or several of the aforementioned functions. The reader is referred to [3] for a more detailed and complete exposition on GraphNets. A custom TensorFlow-based GraphNets library was used². The actual computation graph used in this work is given in Figure 4. The construction of the graphs and the choices for the node, edge, global and aggregation functions is, in general, problem specific. In the following, the feature construction and some architectural choices are detailed.

Node feature construction: Since the strain-input of the network is a time-series, it is natural to process it with a 1D convolutional neural network. The raw strain time-series is augmented by the two first rotational strain invariants, which read,

$$I_1(t, \mathbf{p}_i) = \varepsilon_{11}(t, \mathbf{p}_i) + \varepsilon_{22}(t, \mathbf{p}_i) + \varepsilon_{33}(t, \mathbf{p}_i) \quad (1)$$

$$I_2(t, \mathbf{p}_i) = \varepsilon_{12}(t, \mathbf{p}_i)^2 + \varepsilon_{23}(t, \mathbf{p}_i)^2 + \varepsilon_{13}(t, \mathbf{p}_i)^2 - \quad (2)$$

$$\varepsilon_{11}(t, \mathbf{p}_i)\varepsilon_{22}(t, \mathbf{p}_i) - \varepsilon_{22}(t, \mathbf{p}_i)\varepsilon_{33}(t, \mathbf{p}_i) - \varepsilon_{33}(t, \mathbf{p}_i)\varepsilon_{11}(t, \mathbf{p}_i) \quad (3)$$

where $\mathbf{p}_i = [p_L, p_\phi]$ is the position of sensor i . The position of the sensors along the length of the tube and the angular dimension, is appended as an additional filter dimension. The angular position p_ϕ is encoded as $\sin(p_\phi), \cos(p_\phi)$.

Edge features: In order to localize the crack, information between several nearby sensors may need to be taken into account. Groups of sensors should collectively detect patterns, parametrized by their relative position. This is implemented by endowing the edges of the created graph with features that encode this information. Denoting s_i and r_i the sender and receiver node, the features given in Figure 3 are used. The 1D convolutional network captures temporal variations whereas the GNN should capture parameterized spatial patterns that are associated with the existence of a crack.

²Code can be accessed at: <https://github.com/mylonasc/tf-gnns>

The diagram illustrates the computation of edge features \mathbf{e}_k between two nodes. A source node s_k (yellow dot) has position $\mathbf{v}_{s_k} = \begin{bmatrix} p_L^{s_k} \\ p_\phi^{s_k} \end{bmatrix}$. A target node r_k (yellow dot) has position $\mathbf{v}_{r_k} = \begin{bmatrix} p_L^{r_k} \\ p_\phi^{r_k} \end{bmatrix}$. The edge function \mathbf{e}_k is defined as:

$$\mathbf{e}_k = \begin{bmatrix} \sin(2\Delta\phi) \\ \cos(2\Delta\phi) \\ \cos(\Delta\phi) \\ \sin(\Delta\phi) \\ \Delta L \\ (\Delta L)^2 \\ \sqrt{\Delta L} \end{bmatrix}$$

Relative geometric information includes the angle $\Delta\phi$ and the distance ΔL .

Figure 3: Edge features parametrize the manner in which the different nodes in the GNN exchange information. To that end, non linear relative geometric information is included in the edge function.

Construction of the strain-sensor graph: Since local features of the strain field are more informative for the problem, the spatial proximities of the randomly-placed strain sensors are used to define the graphs employed in the computations, as depicted in Figure 1. For each sensor graph, a set of 150 strain sensors are used ($\frac{150}{10 \cdot 2 \cdot \pi \cdot 0.5 m^2} \approx 4.8 [sensors]/[m^2]$). Sensors at a distance less than 10cm from the crack are removed to avoid detecting the crack only from the local strain concentration magnitude, which renders the problem much less challenging. Different and unique strain sensor graphs and strain sensor positions were used for each training and testing input. In essence, the crack positions, the loading conditions and the strain sensor configurations encountered at testing, are never encountered during training. The edges in the network are bi-directional, i.e. for each directional edge from node i to node j there is a directional edge from node j to node i .

Local reparametrization and uncertainty quantification Since, in general, the defect localization problem may be ill-posed³, the uncertainty in the predictions and learned model needs to be represented. Recent scalable approaches to representing model uncertainty in deep learning include Monte-Carlo Dropout [7] and variational inference with stochastic gradient descent [8, 9]. Variational inference relies on an approximation of the maximum likelihood objective, namely the Evidence Lower Bound (ELBO). In variational techniques, a distribution over the parameters of network layers is inferred from data. The local reparametrization from [8] is used, since 1D regression experiments showed faster and more consistent training. Recent literature for Bayesian GNNs [10] supports these observations.

2.1 Model architecture

The node inputs are passed through a 1D-CNN that acts as the node function of the input GNN $GN^{(input)}$. A simple 2-layer CNN with global max-pooling was chosen, as it is expected that the dynamics play a minor role in identifying the position of the crack from strain. Simple two-layer ReLU Multi-Layer Perceptrons (MLPs) were used as an edge function for the input GNN. No message passing is performed in the input GNN (Graph-independent GNN block). Both $GN^{(input)}$ edge function outputs and $GN^{(input)}$ node-function outputs have a size of 32 units. Networks with fewer units also demonstrated relatively good performance. Following the input GNN, two full GraphNet (core) layers are applied. Residual connections were used for all MLPs involved in $GN^{(core)}$. In the core MLP, instead of the simple two-layer ReLU MLPs, for both the edge and node MLPs an MLP that contains a Bayesian Neural Network layer (BNN layer) is used. The `tensorflow-distributions` implementation [11] of the local-reparametrization layer [8] was used. The inputs are first passed to a fully-connected layer, a single Bayesian Neural Network (BNN) layer with ReLU activation is then applied, and the MLPs conclude with dense layers with no activations. A mean-aggregator was used for the edge-to-node aggregators ($\rho^{e \rightarrow v}$).

In the final computation step, a graph-to-global layer is employed ($GN^{(output)}$), which directly returns estimates on the position of the crack. The nodes and edges output from the $GN^{(core)}$ are mean-aggregated and concatenated. The same type of BNN-MLP used in the core network is used, with different output of three units, corresponding to p_ϕ, p_L, p_ψ , which are the crack parameters.

Some preliminary experiments were performed on the performance of the proposed architecture for different choices of hyper-parameters. Experiments were performed with more $GN^{(core)}$ steps (message-passing through repeated

³Consider the cases where the structure is unloaded, or when the structure is loaded in a manner that does not deform the defect. In these two extreme cases, the problem is not solvable. Moreover, there are multiple material configurations that yield the same discrete strain observations for the same loading.

application of the $GN^{(core)}$ network) with and without BNN layers, and it was found that two and three layers offered comparable results. More than three $GN^{(core)}$ steps did not offer any improvements. Networks with a single message-passing step, regardless of the size of the involved MLPs, performed badly. This observation provides some preliminary evidence that aggregating the information from several sensors is important for the crack localization problem.

Training loss and train-test split Denoting as G^{in} the input graphs, as $\mathbf{p} : [p_\phi, p_L, p_\psi]$ the corresponding crack positions, $\mathcal{D} : \{G^{in}, \mathbf{p}\}$ the dataset, as $p(w)$ a prior distribution over the parameters of the network, the loss function (ELBO) reads

$$\mathcal{L}(\phi, \mathbf{w}, \mathcal{D}) = -D_{KL}(q_\phi(\mathbf{w}) || p(\mathbf{w})) + L_{\mathcal{D}}(\phi)$$

where \mathbf{w} are all the parameters of the associated MLPs, ϕ are the parameters of the variational approximation to the posterior $q_\phi(\mathbf{w})$, $p(\mathbf{w})$ is a prior distribution (a factored Gaussian). Moreover, $D_{KL}(q_\phi(\mathbf{w}) || p(\mathbf{w}))$ is the Kullback-Leibler divergence, and $L_{\mathcal{D}}$ is

$$L_{\mathcal{D}}(\phi) = \sum_{(G^{in}, \mathbf{p}) \in \mathcal{D}} E_{q_\phi(\mathbf{w})}[\log p(G^{in} | \mathbf{p}, \mathbf{w})]$$

which is the *expected negative log-likelihood*. In addition, we are assuming $p(\mathbf{p}) \sim \mathcal{N}(\mu_{\mathbf{p}}, \sigma_{\mathbf{p}}^2)$, and $q_\phi(\mathbf{w}) \sim \mathcal{N}(\mu_{\mathbf{w}}, \sigma_{\mathbf{w}}^2)$. Low variance stochastic gradient estimators exist for the loss function [8, 12].

350 experiments were used as training set and the remaining 100 as testing set. The loss on the testing set was monitored and early stopping was applied after 50 epochs. Random segments of 150 timesteps from the 401 that are available (0.5 seconds) were used from each experiment of the training set, different at each epoch. Interestingly, the network did not suffer from over-fitting, as evaluated in the test-set of unseen sensor configurations and crack positions, even though the number of experiments in the raw dataset number is relatively small (350). This effect is probably a result of the combination of the small networks used and the fact that the learning signal from the dataset is richer per datapoint. Specifically, each graph contains around 150 nodes and several thousands of edges. Essentially, each graph contributes to learning the edge and node functions in proportion to the number of edges and nodes in the graph, not as a single datapoint⁴.

⁴This observation motivates augmenting the dataset with more random graphs even for the same experiment, not a single one as it is the case in this work. This was not attempted due to resource constraints but this investigation may form part of future works.

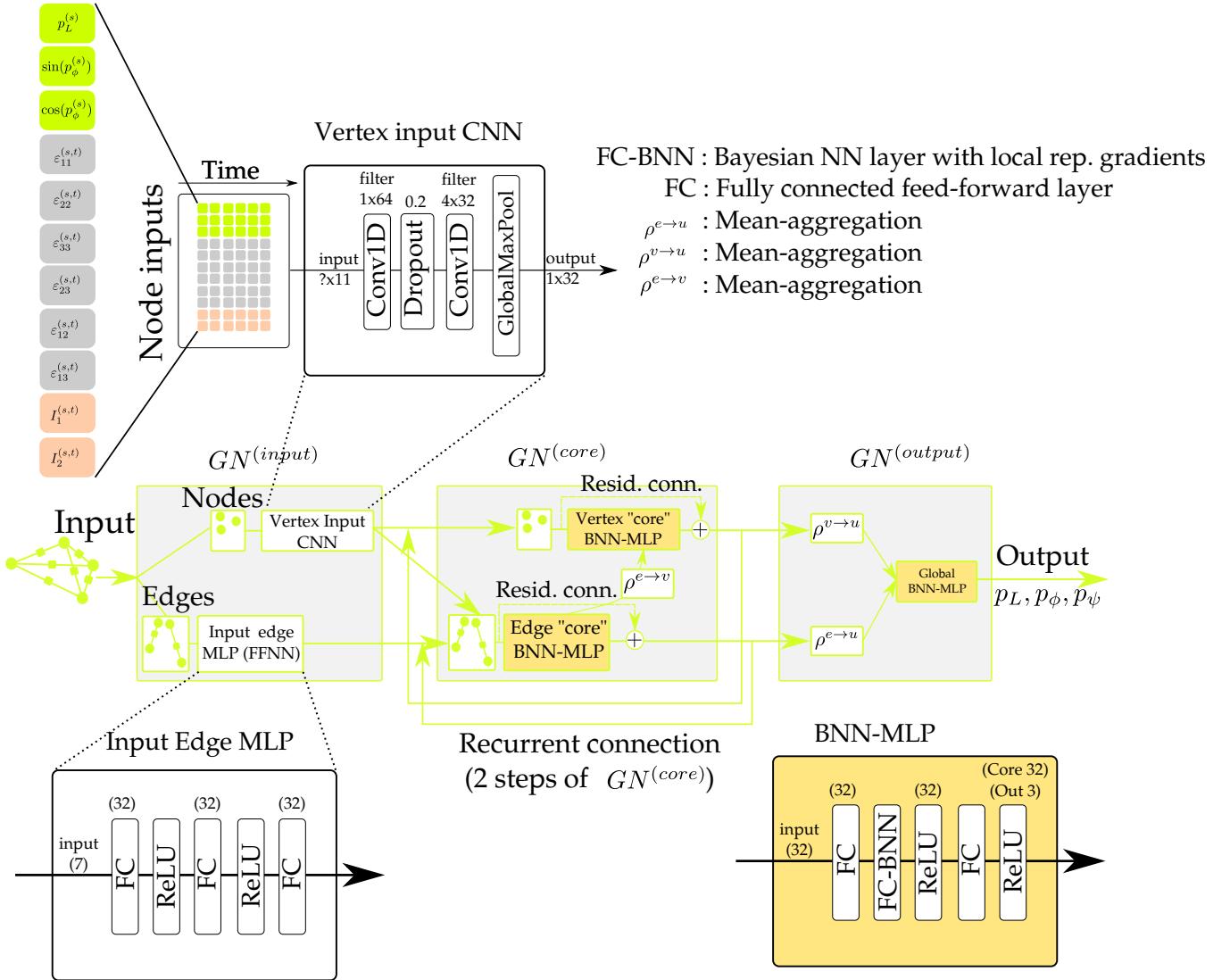


Figure 4: The computational graph used. The $GN^{(core)}$ block is repeatedly applied in order to allow for message passing between a larger neighborhood for each node.

3 Results

The strain localization results for some random cases from the test-set are given in Figure 5. The GNN predicts a set of points close to the actual node position. The results on the angle of the crack were less accurate than the position estimates. Because of the incorporation of Bayesian neural network layers, every forward estimation of the GNN the outputs are different, and each evaluation can be thought of as corresponding to a different model sampled from the posterior distribution of GNN models that was learned from the data. The Normalized Root Mean-Squared Error L^{NRMSE} , computed as,

$$L^{NRMSE}(y_{pred}, y_{actual}) = \frac{\sqrt{\frac{\sum^N (y_{pred} - y_{actual})^2}{N}}}{\max(y_{act}) - \min(y_{act})} \quad (4)$$

is given together with all the training and test set examples in Figure 7. For comparison, localization results on the test-set for a deterministic GNN trained on least-squares loss, with the BNN layers replaced with simple FFNN layers, is given in Figure 6. Although the results are relatively close to the correct predictions, no estimates on the uncertainty of the predictions can be obtained.

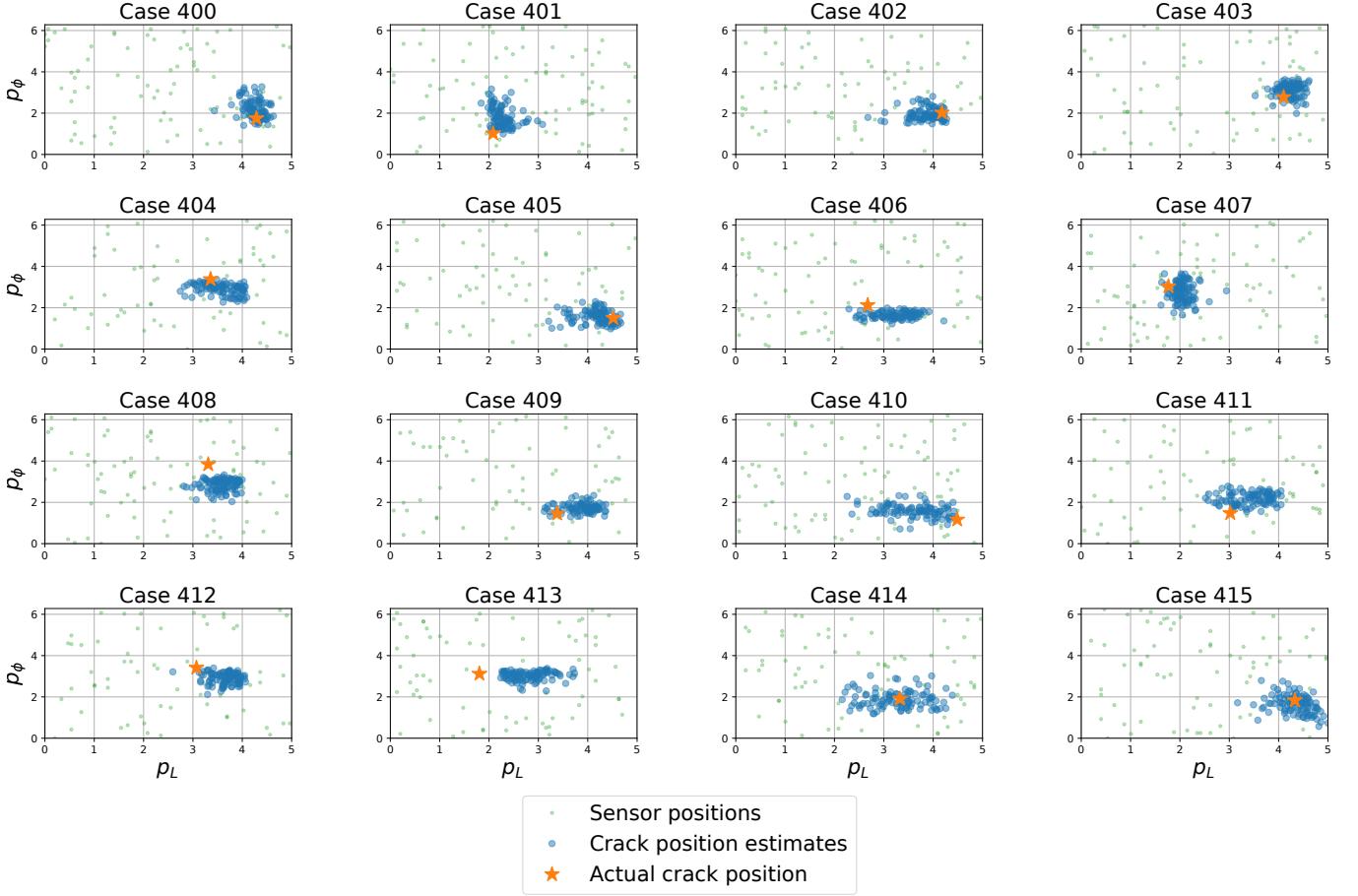


Figure 5: Strain-based localization results on test-set for some random cases for the **Bayesian GNN** model. The localization is performed on time-windows of 200 observations (250ms). Predictions are denoted in blue. The scatter on the predictions, is partially because of the Bayesian neural network layers and partially because of the different inputs for each segment. It is expected that for some time instants the localization is inaccurate since the crack does not contribute to the strain field surrounding it and cannot be detected.

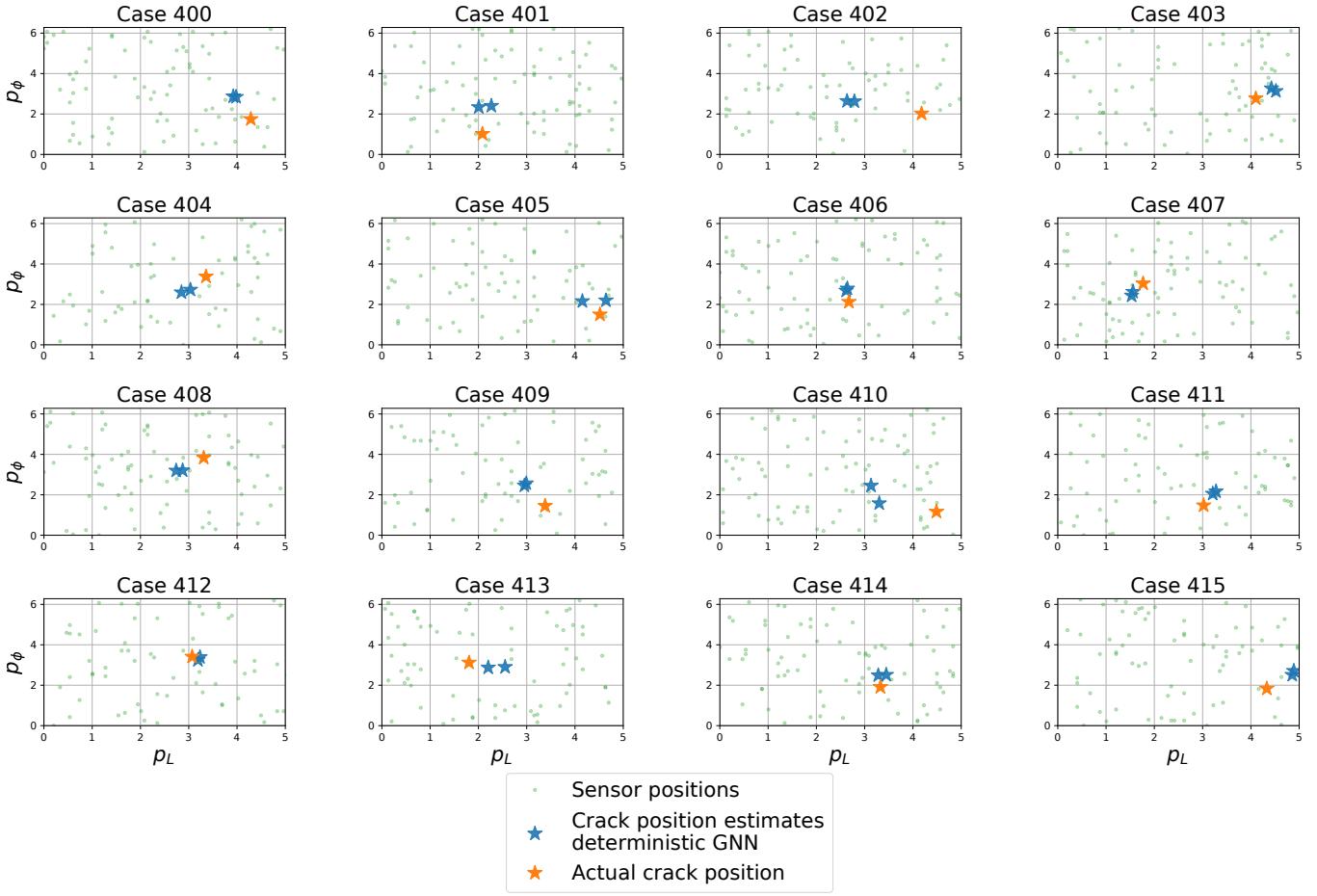


Figure 6: Strain-based localization results on test-set for some random cases for the **least-squares GNN** model. The localization is performed on two non-overlapping time-windows of 200 observations (250ms). Predictions are denoted in blue. The network returns deterministic and incorrect predictions with confidence.

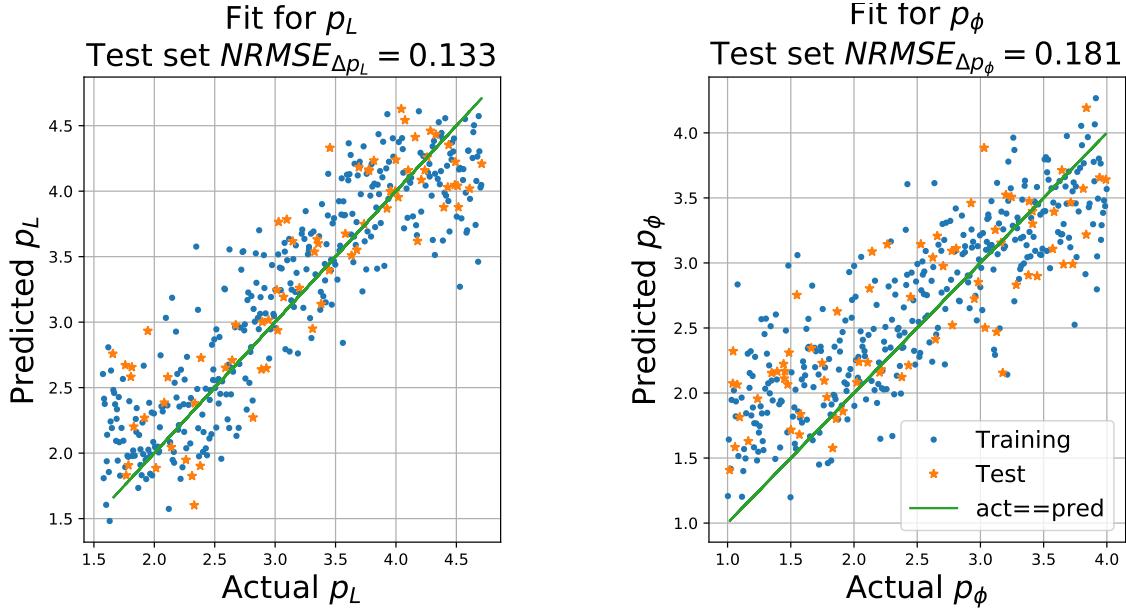


Figure 7: Strain-based localization results - all training and validation sets.

4 Conclusions

In this work, graph neural networks with temporal convolution feature extractors were employed for a defect localization problem, using arbitrary configurations of sensors. Because of the end-to-end differentiability of GNNs, future extensions of this work include optimal sensor placement optimization problems [13]. The incorporation of data from multiple sensor types, such as active piezo-electric sensors can also be naturally implemented in the GNN framework. In addition, GNN-based computation may be a viable alternative to Bayesian PDE inversion problems [14], where observations are sparse, in place of the grid-based CNN computational substrate. Finally, GNNs offer great potential for inference of data-driven Reduced Order Models (ROMs), directly from physical observations. GNNs have already been successfully applied to learning complex physics with particle methods [15] and multi-body dynamics [16] on simulation data.

Acknowledgement C. M. and E. C. would like to gratefully acknowledge the support of the European Research Council via the ERC Starting Grant WINDMIL (ERC-2015- StG #679843) on the topic of Smart Monitoring, Inspection and Life-Cycle Assessment of Wind Turbines. G. T. and K. W. were supported by the EC Marie Skłodowska-Curie ETN grant “DyVirt” (764547).

References

- [1] Nur Sila Gulgec, Martin Takáč, and Shamim N Pakzad. Convolutional neural network approach for robust structural damage detection and localization. *Journal of Computing in Civil Engineering*, 33(3):04019005, 2019.
- [2] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.
- [3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
- [4] Konstantinos Agathos, Eleni Chatzi, and Stéphane PA Bordas. Multiple crack detection in 3d using a stable xfem and global optimization. *Computational mechanics*, 62(4):835–852, 2018.
- [5] Konstantinos Agathos, Stéphane PA Bordas, and Eleni Chatzi. Parametrized reduced order modeling for cracked solids. *International Journal for Numerical Methods in Engineering*, 121(20):4537–4565, 2020.
- [6] Charilaos Mylonas. *Machine Learning for Structural Health Assessment under Uncertainty: with applications in Wind Energy*. PhD thesis, ETH Zurich, 2021.
- [7] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.
- [8] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583, 2015.
- [9] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.
- [10] George Lamb and Brooks Paige. Bayesian graph neural networks for molecular property prediction. *arXiv preprint arXiv:2012.02089*, 2020.
- [11] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.
- [12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [13] Costas Argyris, Sharmistha Chowdhury, Volkmar Zabel, and Costas Papadimitriou. Bayesian optimal sensor placement for crack identification in structures using strain measurements. *Structural Control and Health Monitoring*, 25(5):e2137, 2018.

- [14] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinnns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *arXiv preprint arXiv:2003.06097*, 2020.
- [15] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W Battaglia. Learning to simulate complex physics with graph networks. *arXiv preprint arXiv:2002.09405*, 2020.
- [16] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*, 2018.