Journal of Physics:
Conference Series

PURPOSE-LED
PUBLISHING™

PAPER • OPEN ACCESS

Local flow and loads estimation on wake-affected wind turbines using graph neural networks and PyWake

To cite this article: Gregory Duthé *et al* 2023 *J. Phys.: Conf. Ser.* **2505** 012014

View the article online for updates and enhancements.

You may also like

- Keck Observations in the INfrared of Taurus and Oph Exoplanets And Ultracool Dwarfs (KOINTREAU) I: A Planetary-mass Companion and a Disk-obscured Stellar Companion Discovered in Taurus
Samuel A. U. Walker, Michael C. Liu, Dimitri Mawet et al.

- The Subaru Coronagraphic Extreme Adaptive Optics System: Enabling High-Contrast Imaging on Solar-System Scales
N. Jovanovic, F. Martinache, O. Guyon et al.

- Nonlinear Wave Front Reconstruction from a Pyramid Sensor using Neural Networks
Alison P. Wong, Barnaby R. M. Norris, Vincent Deo et al.

This content was downloaded from IP address 173.177.195.252 on 26/02/2026 at 16:15

Local flow and loads estimation on wake-affected wind turbines using graph neural networks and PyWake

Gregory Duthé[1], Francisco de Nolasco Santos[2], Imad Abdallah[1], Pierre-Élouan Réthoré[3], Wout Weijtjens[2], Eleni Chatzi[1] and Christof Devriendt[2]

[1] Institute of Structural Engineering, ETH Zurich, Switzerland
[2] OWI-Lab, Vrije Universiteit Brussel, Pleinlaan 2, Brussels, Belgium
[3] Technical University of Denmark - DTU, Denmark

E-mail: duthe@ibk.baug.ethz.ch

Abstract. The estimation of the flow conditions throughout a modern wind farm is a relevant albeit complex problem involving several different modelling scales. In this context, recent years have seen the rise of the demand - normally industry-driven - of quick, computationally inexpensive, wind farm flow field simulators upon which further calculations might be undertaken (e.g. power output, fatigue loads, etc.). Naturally, the trustworthiness of such models hinges upon their ability to accurately estimate wakes, under the evident trade-off between computational time and accuracy. One such simulator that has been recently gaining traction within the wind community is PyWake, a Python-based flow solver, with several possible add-ons and modularity options. Based on runs from PyWake, this contribution studies the applicability of graph neural networks (GNN) as a simultaneous wind turbine fleet, wind inflow and loads surrogate. As graphs are discrete mathematical sets of dependent objects, one can see how the layout-dependent interaction between wind turbines' aerodynamics, from which wake arises, lends itself to be modelled as a graph. Thus, we introduce the use of GNNs for layout-agnostic joint modelling of rotor-averaged wind speed, turbulence intensity, power production and damage equivalent loads (DEL) on individual turbines of wind farms. To this end, probabilistic samples of inflow conditions from a Weibull distribution for wind speed, uniform wind direction and conditional normal distributions of wind shear and nacelle yaw angles are generated as main inflow properties in the numerical simulations. Additionally, arbitrary wind farm layouts are created based on varied geometric shapes with random parametrization (varying orientations, length/width ratio). Both the arbitrary layouts and the random inflow conditions are then used as inputs for PyWake. PyWake's output, namely, rotor averaged wind speed, turbulence intensity, power production and DELs of each individual wind turbine in the wind farms, are used to train a general GNN model. We elect to implement an Encode-Process-Decode GNN as a graph learning model. We compare the performance of various approaches of the relational dependency representation (edge-forming techniques) amongst the nodes (wind turbines) on the graph, such as Delaunay triangulation, KNN, radius-based methods and fully connected scheme. In our analysis, we evaluate the accuracy of GNNs and its ability to generalize their joint predictions for unseen layouts and varying inflow conditions.


Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

1. Introduction

A key challenge in the development of wind farms is the reliable estimation of the wind inflow conditions and aeroelastic loads on individual wind turbines under uncertainty. Reaching such a goal implies the need to accelerate computational fluid dynamics (CFD) codes and improve the predictive capabilities of low-fidelity aeroelastic models [1]. Based on a swift flow simulator, operators ought to be able to take informed decisions concerning power production and fatigue load management over a farm. Quick aeroelastic simulations may further enable design and research, where one is required to cover a vast hypothesis/parameter space in search of optima. However, farm-level aerodynamic modelling can be a highly complex task, where different scales (micro- and mesoscale and the so-called *terra incognita*[2]) have to be taken into account. Therefore, we investigate in this contribution the hypothesis of a machine learning flow surrogate, capable of balancing computational costs with wake estimation accuracy. We focus specifically on the use of Graph Neural Networks (GNN) as flow and load surrogates for turbines situated within a farm, as these lend themselves to be modelled as a set of relational objects, where changes in the flow conditions caused by a turbine upstream will have an impact on the wind deficit, power output and fatigue loads of downstream turbines. The layout-dependent interaction between wind turbines' aerodynamics, which lends itself to be modelled as a graph, on the one hand, and the ability of graph neural networks to effectively extrapolate [3], on the other, has led GNNs to not only become one of the hottest themes in machine learning, but also being increasingly implemented in wind energy. The vast majority of employments of GNNs in wind energy has been related with two topics: power [4, 5, 6] prediction tasks and wake loss [7, 8, 9]. Despite these numerous examples, there is yet to be presented a generalizeable and wind farm layout-agnostic GNN approach that combines predictions on wind deficit due to wake interactions, with the fatigue loads these farm-wide interactions cause.

The remainder of this article is organized as follows. In Section 2 we summarize the methodological framework based on Graph Neural Networks (GNN). In section 3 we elaborate on the setup for training a GNN model for wind farms. In section 4 we present the ensuing results and provide some discussions and interpretations. We finish with concluding remarks and outlook in section 5.

2. Methodological framework: Graph Neural Networks

2.1. Graphs and Graph Neural Networks

Any wind farm can naturally be represented as an undirected graph $g = (\mathcal{V}, \mathcal{E})$ composed of a set of nodes $\mathcal{V}$ and edges $\mathcal{E}$, where the nodes are the individual turbines and the edges embody the interactions between them. Each node i can be associated with a vector of features $\mathbf{x}_i \in X$ comprising of f_n physical measurements and properties of the equivalent wind turbine (power, loads, effective turbulence intensity, etc.). Similarly, a vector of attributes $\mathbf{e}_{i,j} \in E$ of size f_e (typically the geometrical features) can also be constructed for each edge connecting node i to node j. The graph may also have a vector of global properties U of size f_g, i.e., attributes, which characterize the farm-level inflow conditions. Subsequently, the attribute encompassing graph representation G of a wind farm with n_n turbine nodes and n_e turbine interaction edges can be defined as $G = (\mathcal{V}, \mathcal{E}, X, E, U)$ with $X \in \mathbb{R}^{n_n \times f_n}$, $E \in \mathbb{R}^{n_e \times f_e}$, and $U \in \mathbb{R}^{f_g}$.

Graph Neural Networks [10] have recently emerged as a powerful method for learning physical processes on graphs [11]. The core of the GNN algorithm is the Message-Passing paradigm [12], a stackable two-step process at the end of which an updated set of node features is obtained. In this process, messages $\mathbf{m}_{i,j}$ are first computed on each edge. Then, for each node, the incoming messages are aggregated (using for instance max, mean, etc.), the results of which are then used to update the node features. In the present study, the physical process that is to be emulated by the GNN corresponds to the influence of wind turbine wakes on local flow variables, power production and on the accumulation of short-term Damage Equivalent Loads (DELs) in the

structural components of the turbines. Given the edge attributes of a wind farm graph and a set of inflow conditions, the GNN should learn to estimate the aforementioned quantities for each turbine. The learning objective can therefore be summarized as a multivariate node regression problem, with no node features being known a priori:

$$X_{estimated} = \text{GNN}(\mathcal{V}, \mathcal{E}, E, U) \tag{2.1}$$

2.2. Encode-Process-Decode GNN architecture

Similarly to [11], in the present work we use an Encode-Process-Decode GNN architecture. This design ensures that the Message-Passing processing happens in a latent space with a much higher dimensionality than the physical parameter space, which has the benefit of improving the expressivity of the GNN. Figure 1 displays a graphical overview of the proposed method.

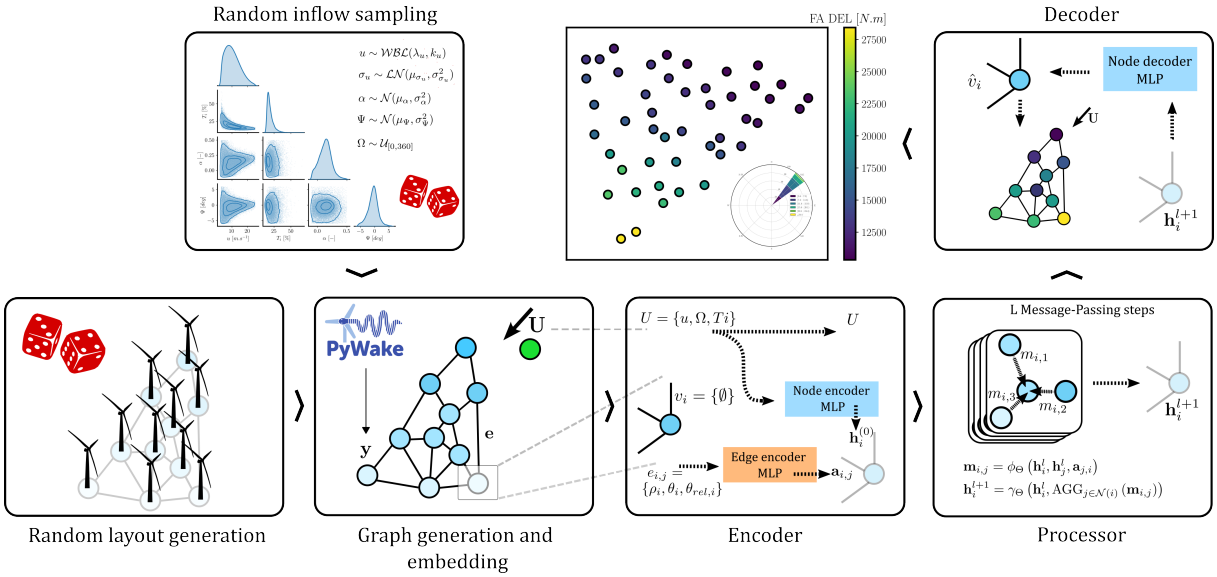


Figure 1: Overview of the proposed GNN method. An Encode-Process-Decode structure is used, ensuring that message-passing takes place in a latent space of large dimensionality. The GNN is trained on randomly generated farm layouts, which are fed into PyWake with random inflow conditions.

2.2.1. Encoding layer In order to operate in a higher-dimensional latent space of size N, the input graph attributes are first passed through an encoding layer. This layer is composed of three separate ReLU activated Multilayer Perceptrons (MLPs) with 2 hidden layers each. The Edge-Encoder takes as inputs the edge features of the graph and outputs the encoded edges $\mathbf{a}_{i,j}$. However, the latent node features $\mathbf{h}_i$ are constructed using the Node-Encoder which takes as an input the vector of inflow globals U. This encoding step is defined as:

$$\mathbf{h}_i^{(0)} = \text{MLP}_{ENC,node}(U), \quad \mathbf{a}_{i,j} = \text{MLP}_{ENC,edge}(\mathbf{e}_{i,j}) \tag{2.2}$$

2.2.2. Message-Passing Processor There are many possible choices for the architecture of the Message-Passing Neural Network (MPNN) Processor layers. In the present work, we opt for the GENeralized Aggregation Networks (GEN) [13] operator. GEN is a modified version of the popular Graph Convolution Network [14], which allows for edge features and other more

advanced message aggregation AGG functions. Our Processor is thus composed of a stack of L GEN Message-Passing layers, where the message-passing update of the latent node features for layer l writes:

$$\mathbf{m}_{i,j} = \text{ReLU}(\mathbf{h}_j^{(l-1)} + \mathbf{a}_{i,j}) + \epsilon \tag{2.3}$$

$$\mathbf{h}_i^{(l)} = \text{MLP}_{(l)}\left(\mathbf{h}_i^{(l-1)} + \text{AGG}(\mathbf{m}_{i,j})\right), \quad j \in \mathcal{N}_i \tag{2.4}$$

2.2.3. Decoding layer The decoding layer serves to obtain the final physical quantities from the latent node features. Similarly to the encoder, it is composed of a ReLU activated MLP with 2 hidden layers.

$$\mathbf{x}_i = \text{MLP}_{DEC,node}(\mathbf{h}_i^{(L)}) \tag{2.5}$$

3. Consideration for the GNN model development for wind farms

3.1. Wind farm layout geometry generation for model training

In the training phase, we generate random (arbitrary) layout geometries for wind farms. For the random sampling, O'Neill's permutation congruential generator (PCG) [15] has been used, as implemented by `numpy.random`. By selecting a minimal distance between turbines (dependent on the rotor diameter), four basic geometries might be defined: rectangle, triangle, ellipse and sparse circles. The equally spaced initial points of each base geometry are subsequently randomly perturbed and rotated by a random angle. In Figure 2, an example of the random layout geometry generation can be seen. For each training sample set, only one of the four defined geometries is selected in order to produce a unique (albeit arbitrary) layout.

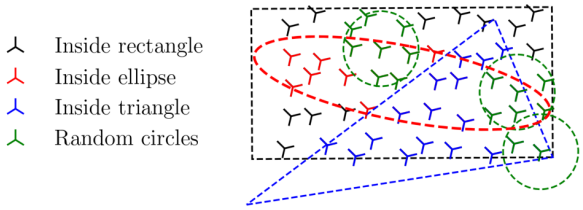


Figure 2: Example of random layout geometry generation for 50 turbines.

3.2. Edge connectivity

There are several ways to establish the graph edge connectivity of a given wind farm layout. One such scheme is based on the Delaunay triangulation [16, 17], a popular algorithm for unstructured triangular mesh generation. This method results in sparse graphs, with a relatively small number of edges connecting the turbines. Another possibility is to use K-Nearest Neighbors (KNN) clustering [18] to form edges between a turbine and and its K closest neighbors. KNN graphs are denser graph compared to the Delaunay method, with more edges connecting compact communities of turbines. Alternatively, the Radial connectivity scheme can be used to construct edges between a turbine and all other turbines within a certain radius. This scheme yields graph topologies similar to KNN graphs, with differences in density depending on the selected radius. These two methods are also similar in the fact that they can result in disconnected communities, as highlighted in Figure 3. Lastly, the Fully Connected scheme forms edges between all turbines, resulting in the densest graph amongst the methods.

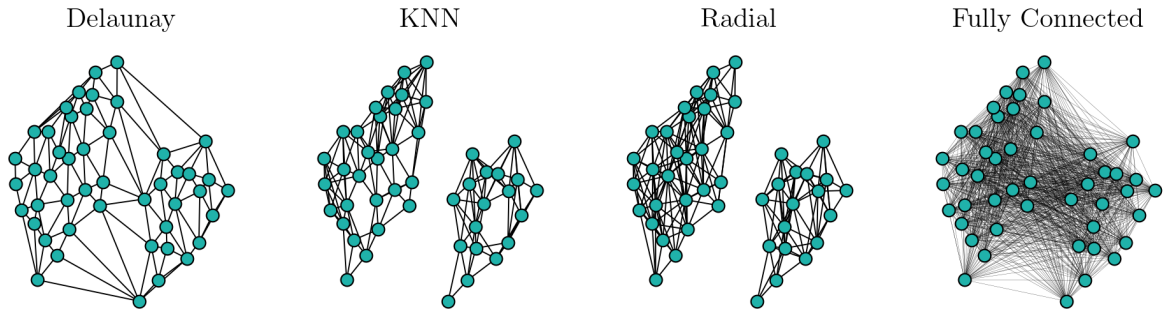


Figure 3: Side-by-side comparison of the different proposed edge connectivity schemes applied to a randomly generated wind farm layout. Both the KNN and the Radial schemes can lead to disconnected communities.

3.3. Sampling of inflow conditions

In order to perform PyWake's aerodynamic simulations, a turbulent inflow wind field must be described for the far-field conditions. The random variables for the turbulent inflow consist of mean wind speed, u, turbulence intensity T_i, wind shear α and the horizontal inflow skewness, Ψ [19]. The mean wind speed follows a truncated Weibull distribution, $u \sim \mathcal{WBL}(\lambda_u, k_u)$, with $[3-25]m.s^{-1}$ bounds, the turbulence intensity the log-normal distribution $\sigma_u \sim \mathcal{LN}(\mu_{\sigma_u}, \sigma^2_{\sigma_u})$, with $I_{ref} = 0.16$ and $T_i = 100 \cdot (\sigma_u)/u$, the wind shear exponent from a truncated normal distribution, $\alpha \sim \mathcal{N}(\mu_\alpha, \sigma^2_\alpha)$ with bounds $[-0.1, 0.5]$, the inflow horizontal skewness, Ψ, is defined by the truncated normal distribution $\Psi \sim \mathcal{N}(\mu_\Psi, \sigma^2_\Psi)$, with bounds $[-6, 6]deg.$ and the wind direction is sampled from a uniform distribution, $\Omega \sim \mathcal{U}_{[0,360]}$. These are sampled through Sobol Quasi-Random sequences [20].

3.4. PyWake simulations

PyWake [21] is used to generate the base simulations upon which the GNN models will be trained. This is a multi-fidelity wind farm flow simulation tool that uses Python and `numpy` to compute flow fields and power production for wind turbines/farms quickly. As this contribution focuses in developing a GNN farm-wide loads and flow surrogate framework, PyWake lends itself perfectly - due to its computationally inexpensive simulations - to this task, by quickly generating simulated results for wind deficit, power and damage equivalent loads of turbines located within a farm without having to invest too much time in running said simulations. In this particular implementation, Niayifar's Gaussian wake deficit model [22] is used, with the Crespo-Hernández's model [23] provides the wake-added turbulence model and the pairwise load surrogate model based on HAWC2 simulations [24] is employed. This combination allows for the fastest simulation time while retaining the necessary elements of physics to enable evaluating the wake induced fatigue loads of each turbine. The selected turbine is IEA $3.4MW$ reference wind turbine [25]. Thus, given a global far-field wind vector, as described in Section 3.4, PyWake is capable of outputting locally (*i.e.*, for each turbine position within a farm) 8 emulated *'sensor'* quantities: local wind speed, turbulence intensity, power, and Damage Equivalent Loads (DELs) for tower fore-aft, tower side-to-side, tower top torsional, blade root flapwise and edgewise.

3.5. Implementation and training details

The final settings of the GNN architecture, which is implemented in `PyTorch-Geometric` [26], were obtained through iterative hyperparameter optimization, using the Bayesian hyperparameter tuning package `Optuna` [27]. The optimal hyperparameter configuration was determined

as 512 for the layer dimension, 6 message-passing steps, the softmax aggregation, 1e-3 starting learning rate, 0.96 for the learning rate decay γ, dropout only in the encoder with 0.05, batch size of 100 and 150 epochs. All models are trained with these parameters, with the exception of the fully connected scheme, where a smaller batch size of 60 was used due to GPU memory constraints. During training, a $L2$ loss was used as the objective function.
Using the data generation framework previously described, we generate three independent datasets for training, validation and final testing, with respectively 3000, 1000 and 1000 arbitrary sampled wind farm layouts, each with 10 inflow conditions, resulting in 30k training graphs, 10k validation graphs and 10k testing graphs.

4. Results and discussions

4.1. Test set prediction accuracy

To assess which of the four connectivity schemes described in Section 3.2 leads to the most accurate power, turbulence and load prediction results, we compare four identical models, trained and evaluated on the same data, each with a different connectivity scheme. The results of this assessment are gathered in Table 1, where for each output variable the Root Mean Squared Error (RMSE) and the Mean Absolute Percentage Error (MAPE) is computed for the K nodes of each test set graph:

$$\text{RMSE} = \sqrt{\frac{1}{K}\sum_{k}^{K}(y_k - \hat{y_k})^2}, \quad \text{MAPE} = \frac{100}{K}\sum_{k}^{K}\frac{|y_k - \hat{y_k}|}{|y_k|} \tag{4.1}$$

with y the true values and $\hat{y}$ the predicted values. These metrics are then averaged over the entire test set. In this table we also compare the GPU VRAM memory footprint and average training iteration time for each connectivity scheme, when trained on an RTX 3090 GPU with a batch size of 60. It can be noted that for most output variables, the Fully Connected method outperforms the other schemes with the exception of wind speed and power prediction. However, despite having the smallest memory and compute requirements (55.6% smaller memory footprint and 47.5% faster train step time compared to Fully Connected), the Delaunay scheme yields comparable results to the Fully Connected method with a maximum increase in the DELs error of only 1.3 percentage points. Moreover, scaling to large graphs is difficult with the Fully Connected method and there are additional difficulties caused by the training instability of this scheme, in part due to its sensitivity to initialization. In light of these results, we recommend using the Delaunay triangulation connectivity method.

Table 1 also highlights which of the variables are the hardest to capture. In general, the loads are more difficult to predict accurately, with the fore-aft, flapwise and side-to-side DELs exhibiting the highest relative errors, irrespective of the chosen connectivity.

4.2. GNN wake learning and connectivity expressivity

To highlight the generalizability of the wake learning ability of the proposed GNN model, a randomly generated farm layout is chosen and simulated with a large range of inflow conditions with a selected dominant wind direction (E-NE). This layout was specifically selected, presenting a difficult setup with three distanced turbine clusters specifically designed to challenge the edge connectivity schemes presented earlier in this paper. In Figure 4, the windrose and Delaunay layout connectivity are shown on the top of the figure, whereas the estimated power, wind speed, turbulence intensity and loads are plotted against the ground truth by wind speed. Wind turbine (WT) 15 (free-flow) and 3 (wake-affected) are studied. Likewise, Figure 5 performs the same for KNN connectivity. Both figures illustrate the ability of the GNN model to capture not only the behaviour of free-flow turbines, but the effects of wake on downstream turbines,

Variable			Delaunay	KNN	Radial	Fully Connected
Power	RMSE	$[kW]$	**33.55**	34.02	39.16	35.86
	MAPE	[%]	3.17	**3.15**	3.46	3.35
Wind speed	RMSE	$[m.s^{-1}]$	0.084	**0.082**	0.111	0.089
	MAPE	[%]	0.98	**0.92**	1.26	1.11
Turbulence Intensity	RMSE	[-]	**0.0020**	0.0022	0.0028	0.0020
	MAPE	[%]	1.05	1.14	1.49	**1.03**
Flapwise DEL	RMSE	$[N.m]$	424.01	440.94	479.47	**399.37**
	MAPE	[%]	11.71	12.21	13.68	**11.20**
Edgewise DEL	RMSE	$[N.m]$	78.76	78.55	89.23	**76.64**
	MAPE	[%]	2.52	2.70	3.05	**2.49**
Fore-aft DEL	RMSE	$[N.m]$	2708.61	2917.06	3398.47	**2450.37**
	MAPE	[%]	13.37	14.67	18.18	**12.19**
Side-to-side DEL	RMSE	$[N.m]$	1042.45	1087.97	1217.17	**971.04**
	MAPE	[%]	11.97	12.80	14.78	**11.17**
Torsional DEL	RMSE	$[N.m]$	259.48	276.47	323.44	**224.85**
	MAPE	[%]	9.37	9.98	12.11	**8.04**
Train VRAM footprint		$[GB]$	**8.02**	13.89	17.7	18.07
Avg. train step time		$[s]$	**117**	126	142	223

Table 1: RMSE and MAPE results for the different connectivity schemes, averaged over the test set for each output variable. Results in bold script indicate the best performing connectivity scheme. Power MAPE metrics were computed while excluding cases where no power was generated.

especially notable on the fatigue loads, specifically fore-aft, torsional and blade flapwise, so long as the connectivity ensures there are no disconnected clusters (as with KNN). This allows us, not only to affirm that the GNN model is successfully learning wakes and their impact on wind deficit, power and loads, but also to qualitatively justify the greater expressivity of the Delaunay triangulation scheme.

4.3. Test case: Horns Rev

Finally, The GNN multivariate predicting model is applied to the Horns Rev 1 offshore wind farm [28]. Installed with 80 Vestas V80 turbines of 2MW, and a grid layout with a spacing of 7 rotor diameters, this is a well-known real-world wind farm layout. In this analysis, as PyWake's calculations are based on IEA's 3.4MW turbine, the V80's properties can not be used. However, along with the exact layout geometry, the PyWake simulations are performed using the actual wind rose for the park, as seen in Figure 6, where, we can easily identify a dominant W-SW wind direction. As an illustrative case of real-world employment of the GNN model, in Figure 6 we present the farm-wide plot for the average fore-aft DELs (500 samples) for the PyWake simulations and the GNN model's estimation). It can be seen that the GNN model is effectively able to understand that, as expected, free-flow facing turbines in the W-SW direction face lower loads, with these progressively increasing throughout the farm. As the wind profile is a complex one (*i.e.*, not a single wind direction), we see how all edge turbines present lower fatigue loads. However, we notice that the model is underpredicting the loads for wake-affected turbines in the middle of the farm (in yellow in middle panel of Figure 6). To determine the severity of the errors on the GNN model (which might appear worse due to the range and color map of the color scale), we can consult Figure 7, where the difference between PyWake's simulations and the GNN's predictions is given as a percentage ($100 \times (DEL_{pywake} - DEL_{gnn})/DEL_{pywake}$). In Figure 7, it can be seen that indeed the model is generally underpredicting (positive difference),

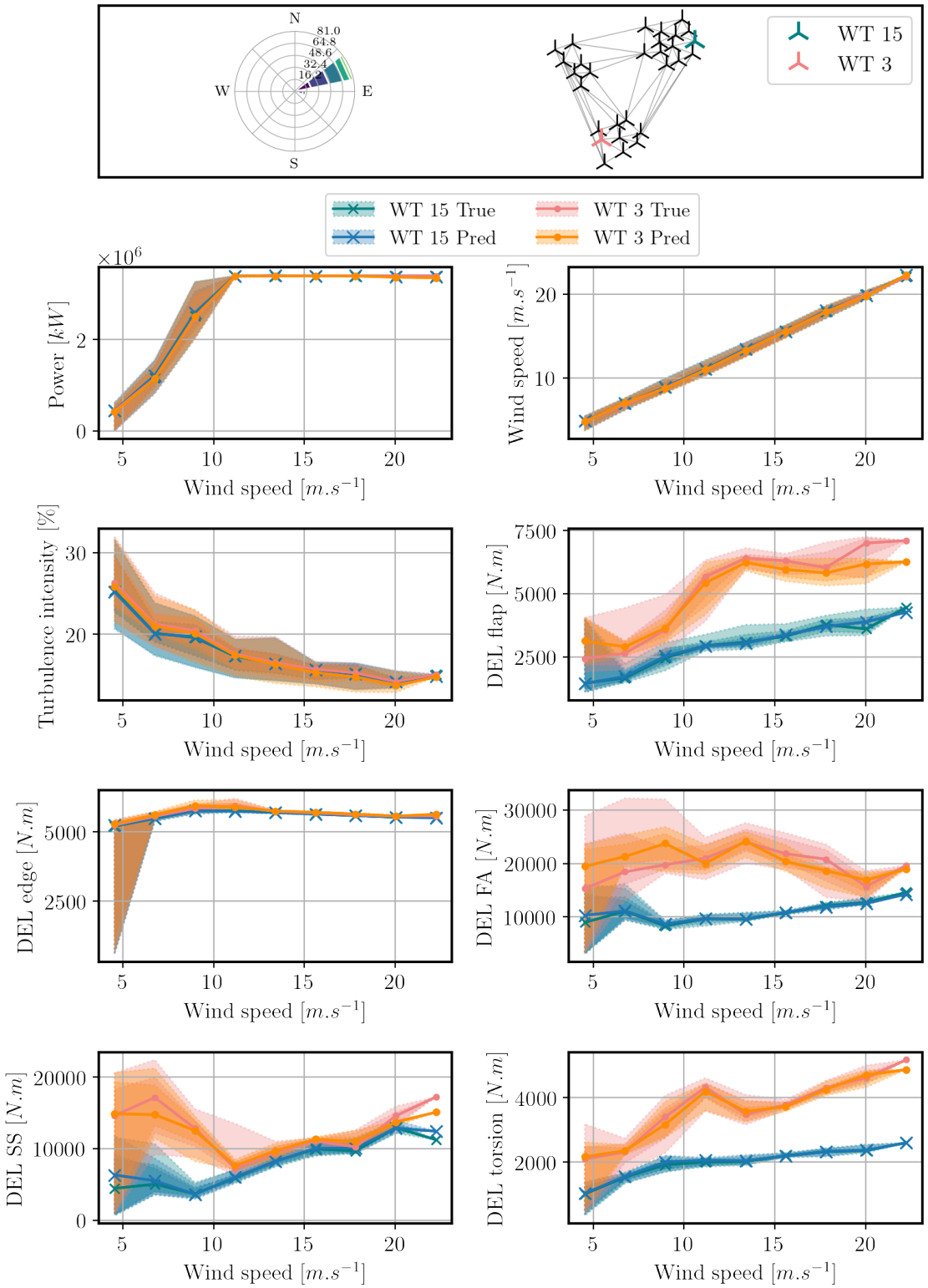


Figure 4: Predicted vs simulated quantities plotted against global far-field wind speed for two wind turbines in different wake conditions, using the Delaunay connectivity scheme. Despite the turbines being in separate clusters and WT 3 not being directly connected to the upstream cluster, the model is able to accurately predict the wake-induced increase in DEL accumulation for this turbine.

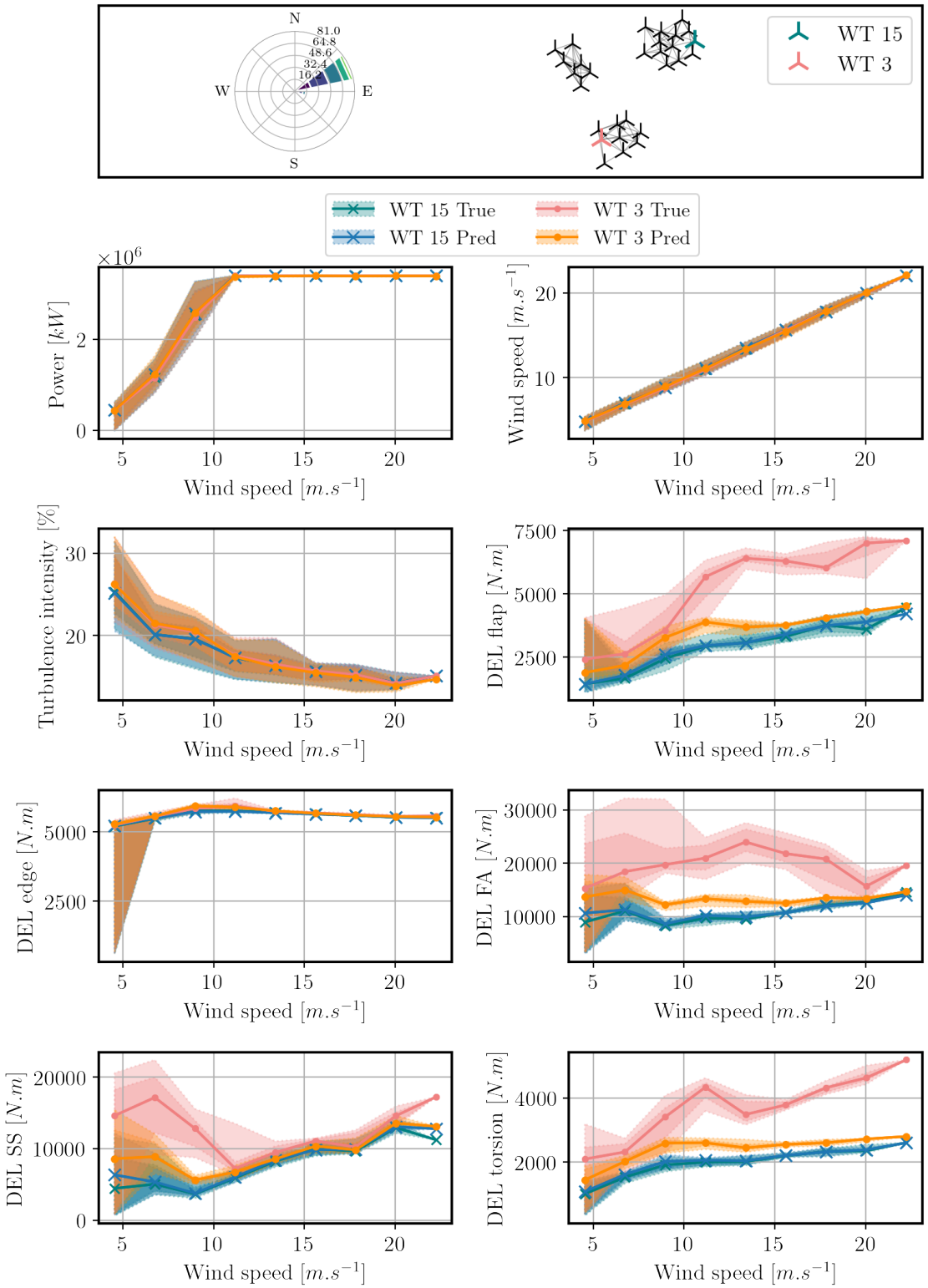


Figure 5: Predicted vs simulated quantities plotted against global far-field wind speed for two wind turbines in different wake conditions, using the KNN connectivity scheme. The model is unable to capture the heightened DEL accumulation for WT 3 due to turbine clusters being disconnected.

but that this value is kept between 2 and 4% for the wake-affected turbines in the middle of the farm. Even the greatest errors, close to the edges of the farm, don't exceed 7%.

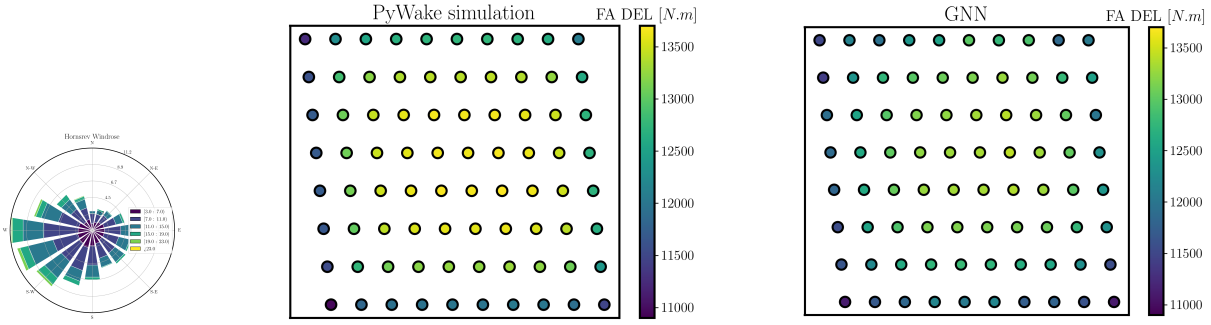


Figure 6: Comparison between PyWake and GNN on Horns Rev geometry and inflow conditions.

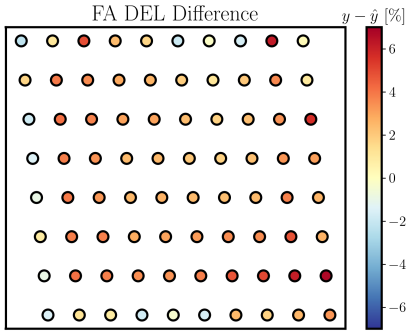


Figure 7: Relative percent error on the GNN model's FA DEL predictions.

5. Conclusions and outlook

In this contribution a novel GNN flow and load surrogate model is presented based on PyWake simulation training data for random wind-farm layouts. The Encoder-Processor-Decoder GEN model was able to accurately predict local fatigue loads, power and wind deficit given a global far-field wind flow vector. Compared to other established methods, such as Gaussian processes, GNNs are able to automatically learn the dependencies between turbines without the need to explicitly encode this information (e.g. through kernels and correlation functions). While other GNN wind farm methods have typically focused on power and flow deficit prediction, our GNN method is able to, in addition, capture the behavior of the DELs, a more challenging task due to the increased sensitivity of the loads to wake conditions and heterogeneity of DEL to the materials' Wöhler curves of various structural components on the wind turbines, which leads to heightened graph heterophily for these variables. In this work, four graph connectivity types were compared (Delaunay, KNN, Radial and Fully Connected), with the best balance between accuracy and computational performance found for the Delaunay triangulation connectivity. Finally, a test-case based on the real-world farm Horns Rev is presented, with good agreement in relation to the simulator ground truth.

This contribution presents a successful implementation of a farm-wide GNN, leaving the door open for further research questions, namely, a deeper study on the GNN architecture (including the message-passing layers) and on the substitution of the PyWake pairwise load surrogate with higher fidelity models along with possible real-world data implementations.

Acknowledgments
The authors would like to acknowledge the Fonds Wetenschappelijk Onderzoek – Vlaanderen (FWO) for its long stay abroad grant.

References
[1] Van Kuik G, Peinke J, Nijssen R, Lekou D, Mann J, Sørensen J N, Ferreira C, van Wingerden J W, Schlipf D, Gebraad P *et al.* 2016 *Wind energy science* **1** 1–39
[2] Wyngaard J C 2004 *Journal of the atmospheric sciences* **61** 1816–1826
[3] Xu K, Zhang M, Li J, Du S S, Kawarabayashi K i and Jegelka S 2020 *arXiv preprint arXiv:2009.11848*
[4] IOP Publishing 2022 *Wind Park Power Prediction: Attention-Based Graph Networks and Deep Learning to Capture Wake Losses* vol 2265
[5] Yu M, Zhang Z, Li X, Yu J, Gao J, Liu Z, You B, Zheng X and Yu R 2020 *Future Generation Computer Systems* **113** 145–157
[6] Park J and Park J 2019 *Energy* **187** 115883
[7] Duthé G, Abdallah I, Barber S and Chatzi E 2023 *arXiv preprint arXiv:2301.03228*
[8] Bleeg J 2020 *Journal of Physics: Conference Series* **1618** 062054
[9] Li S, Zhang M and Piggott M 2022 *arXiv preprint arXiv:2211.13649*
[10] Scarselli F, Gori M, Tsoi A C, Hagenbuchner M and Monfardini G 2008 *IEEE transactions on neural networks* **20** 61–80
[11] Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec J and Battaglia P 2020 Learning to simulate complex physics with graph networks *International Conference on Machine Learning* (PMLR) pp 8459–8468
[12] Gilmer J, Schoenholz S S, Riley P F, Vinyals O and Dahl G E 2017 Neural message passing for quantum chemistry *International conference on machine learning* (PMLR) pp 1263–1272
[13] Li G, Xiong C, Thabet A and Ghanem B 2020 *arXiv preprint arXiv:2006.07739*
[14] Welling M and Kipf T N 2016 Semi-supervised classification with graph convolutional networks *J. International Conference on Learning Representations (ICLR 2017)*
[15] O'Neill M E 2014 *ACM Transactions on Mathematical Software*
[16] Delaunay B *et al.* 1934 *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* **7** 1–2
[17] Musin O R 1997 Properties of the delaunay triangulation *Proceedings of the thirteenth annual symposium on Computational geometry* pp 424–426
[18] Cover T and Hart P 1967 *IEEE transactions on information theory* **13** 21–27
[19] Avendaño-Valencia L D, Abdallah I and Chatzi E 2021 *Renewable Energy* **170** 539–561 ISSN 0960-1481
[20] Sobol' I M 1967 *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* **7** 784–802
[21] Pedersen M M, van der Laan P, Friis-Møller M, Rinker J and Réthoré P E 2019 *Zenodo [code]* **10**
[22] Niayifar A and Porté-Agel F 2016 *Energies* **9** 741
[23] Crespo A and Hernández J 1996 *Journal of wind engineering and industrial aerodynamics* **61** 71–85
[24] Larsen T J and Hansen A M 2007 *How 2 HAWC2, the user's manual* (Risø National Laboratory)
[25] Bortolotti P, Tarres H C, Dykes K L, Merz K, Sethuraman L, Verelst D and Zahle F 2019 Iea wind tcp task 37: Systems engineering in wind energy-wp2. 1 reference wind turbines Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States)
[26] Fey M and Lenssen J E 2019 *arXiv preprint arXiv:1903.02428*
[27] Akiba T, Sano S, Yanase T, Ohta T and Koyama M 2019 Optuna: A next-generation hyperparameter optimization framework *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* pp 2623–2631
[28] Méchali M, Barthelmie R, Frandsen S, Jensen L and Réthoré P E 2006 Wake effects at horns rev and their influence on energy production *European wind energy conference and exhibition* vol 1 (Citeseer Princeton, NJ, USA) pp 10–20