Responses to Reviewers' Comments for Manuscript 10.5194/wes-2025-261

Graph Neural Operator for wind farm wake flow

Addressed Comments for Publication to

WES Wind Energy Science

by

Jens Peter Schøler, Frederik Peder Weilmann Rasmussen, Julian Quick,

and Pierre-Elouan Réthoré

Document Overview

This document contains the authors' responses to the reviewers' comments for manuscript wes-2025-261, "Graph Neural Operator for wind farm wake flow." The document is structured as follows:

- **Section 1:** Responses to Reviewer 1's comments
- **Section 2:** Responses to Reviewer 2's comments
- **Appendix:** Manuscript with tracked changes

The appendix presents a marked-up version of the revised manuscript generated using `latexdiff`, with deletions shown in red strikethrough and additions shown in blue. Please note that the figure and table formatting in the tracked changes version is suboptimal and for inspecting those the main file should be consulted.

Authors' Response to Reviewer 1

General Comments. 2025-12-12 The manuscript wes-2025-261 presents an advanced deep-learning approach for constructing a surrogate model to predict wind-speed deficits within and downstream of wind farms. The model is trained on a large data set generated by a low-fidelity, steady-state wake model (PyWake). The authors highlight prediction speed and generalisation to a range of wind-farm layouts and inflow conditions as key advantages. The deep-learning architecture is described in detail, and many test results are provided. Although the model architecture is sophisticated and the method is thoroughly described, I believe the manuscript requires major revisions for the following reasons:

Response: Thank you for your thorough review of our manuscript.

Your detailed feedback has improved the quality and clarity of the paper. We appreciate the time you invested in examining both the methodology and presentation of our work. We have addressed your major concerns, by:

- Toning down claims of "breakthrough" in favor of more measured language emphasizing our methodological contribution
- Clarifying the nature of our contribution relative to existing GNN-based approaches in the literature
- Revising the abstract to remove potentially misleading metrics, and renamed MAPE to MANE as to not use incorrect definitions
- Moving technical implementation details to the appendix to improve readability for the WES audience
- Adding new visualizations including complete flow map comparisons with error plots (new Figure 11)

We have also addressed the specific comments, making corrections to formatting, clarifying explanations, and adding requested information throughout the manuscript. Below, we provide point-by-point responses to each of your comments.

Major Comments

Comment 1

The manuscript concludes that the proposed method represents a breakthrough in data-driven wind- farm modelling and provides a robust baseline for efficient flow prediction. However, the literature already contains ML-based surrogate models, including GNN-based and other approaches. The manuscript does not demonstrate that the proposed model outperforms these (simpler) existing alternatives or offers clear additional value. To substantiate the claimed contributions, a comparative analysis is needed.

Response: We appreciate the reviewer's comment and have toned down the claim of "breakthrough" in favor of more measured language emphasizing the methodological contribution, as also suggested by Reviewer 2.

We have also tried to make the nature of our contribution clearer in the revised manuscript. The novelty lies not in outperforming existing GNN-based models on the same task, but in addressing a different prediction target: spatially continuous flow fields across the entire wind farm, rather than turbine-level quantities. As discussed in Section 1, previous GNN applications to wind farms focus on predicting power or loads at turbine locations, while other machine learning approaches has their own limitations. We have added the following text to Section 1 clarifying the advantages of the graph-based approach over alternatives such as CNNs and MLPs:

" The graph-based formulation offers distinct advantages over alternative neural network architectures for this application. Convolutional Neural Networks (CNNs) require fixed grid structures and, therefore, struggle to accommodate varying turbine layouts across farms with different sizes and densities. Simultaneously, they need a fixed grid, often with limited resolution, which imposes an upper resolution limit, whereas a GNN-based approach enables inference at the exact positions of interest. Multilayer Perceptron (MLP) based surrogate models for single-wake prediction have been proposed, but these still require a classical superposition scheme to reconstruct the full farm flow, inheriting the limitations of algebraic wake summation. In contrast, the message-passing framework inherent to GNNs naturally represents turbine-turbine interactions through graph connectivity, enabling the model to learn meaningful turbine interactions, including nonlinear interactions, that generalize across diverse layouts without relying on explicit superposition. For a review on data-driven methods in wind farms fluid flow, see e.g. Zehtabiyan-Rezaie et al. (2022). "

Comment 2

The performance metrics reported in the abstract and conclusion of the manuscript (RMSE and MAPE) are difficult to interpret (without having read in detail the complete manuscript) and, in my view, give an overly optimistic impression. As the primary purpose of PyWake—and by extension of the surrogate model—is to accurately represent wake flow velocity deficits, the MAPE should be defined with respect to the ground-truth velocity deficit, instead of to the free-stream wind speed. As shown in Figure 9, for moderate to high wind speeds, the predicted deficits in waked zones deviate by more than 100% from the ground-truth wind velocity deficits. In addition, the reported RMSE value is averaged over a large domain that may include large zones without any wind velocity deficit, where the prediction problem is trivial. This may inflate this performance metric and should be clarified or adjusted. In order to assess the accuracy of the proposed GNO approach, it may also be useful to compare its accuracy to the accuracy of PyWake itself (either relative to real-world data or high-fidelity models), as reported in previous studies.

Response: We agree that presenting a non-standard variant of MAPE in the abstract was somewhat misleading, and that the remaining metrics were also challenging to interpret without the full context of the problem. To address this, we have removed the performance metrics from the abstract. Regarding the MAPE metric specifically, we have renamed it to Mean Absolute Normalised Error (MANE) throughout the manuscript to avoid confusion with the standard definition. We have also added an explanation clarifying its relationship to conventional MAPE and why standard MAPE is problematic in this application, specifically, normalising by the target velocity deficit leads to numerical instability when deficits approach zero in lightly waked regions.

Regarding the concern that RMSE is inflated by unwaked regions that are supposedly trivial to predict, we would argue that accurate prediction in these areas is nonetheless important: a model that erroneously predicts wake deficits where none exist would be equally problematic for downstream applications. Furthermore, as illustrated in Fig. 3, the adaptive bounding box used for data generation ensures that the evaluation domain is constrained to regions of physical interest, thereby limiting the influence of far-field areas with negligible wake effects.

Finally, regarding the suggestion to compare the GNO's accuracy against PyWake's own accuracy relative to high-fidelity data or measurements: while we appreciate the value of such a comparison, we consider it outside the scope of the present work. The focus of this study is on the development and evaluation of the GNO as a surrogate, rather than on validating PyWake itself. Validation studies of PyWake have been reported elsewhere in the literature.

Comment 3

The abstract notes as motivation for implementing a surrogate model, the many required simulations for applications such as wind-farm layout optimisation and consideration of neighbouring wind farms. It is therefore recommended that the manuscript explicitly addresses whether—and by how much—the proposed surrogate model really adds value for these targeted applications, as well as any limitations that may arise from using the GNO-approach compared to using PyWake.

Response: Thank you for this comment. We acknowledge that the original abstract was misleading regarding the model's intended use for layout optimization. The abstract has been revised to emphasize that the primary motivation is assessing wake effects from neighboring farms, which requires many simulations. We now mention that such surrogates may make it easier to consider neighboring farm wakes during layout optimization, rather than suggesting the model is directly intended for WFLO. In Section 3.2, we have added a discussion of how the GNO's two-stage architecture suits the neighboring farm scenario. Specifically, a neighboring farm only needs to be encoded once, after which the latent turbine states can be reused to evaluate wake deficits at arbitrary query locations. This allows candidate positions to be assessed without reprocessing the neighbor each time, which is advantageous when exploring many layout alternatives. We believe this framing better reflects the model's current capabilities and its most promising near-term application.

Comment 4

Readers of Wind Energy Science are typically not AI specialists. The manuscript contains many low- level implementation details that obscure the core ideas and may discourage readers from completing the lecture of the manuscript. Therefore, consider moving purely technical implementation aspects to an appendix. As the software code is openly available (which is greatly appreciated), some details that are generic to deep-learning workflows might even be omitted entirely from the manuscript.

Response: Thank you for acknowledging our sharing of the code. We understand that we may have included too many details; however, we believe that precisely because WES readers are not AI specialists, they may benefit from a more thorough description of the implementation. We also believe that this is in line with the change to a stronger focus on the methodological aspects of the paper.

That said, we have moved some of the generic deep learning details to the appendix, including dropout regularization and layer normalization. The grid search tables from the results section have also been relocated to the appendix.

Specific Comments

Abstract

Comment 5

Line 3: The text implies that the classical superposition principle is a disadvantage of existing approaches, yet the surrogate model is trained by a model that implements such an approach. Please rephrase to avoid suggesting a contradiction that is only clarified in the conclusion of the manuscript.

Response: We have now added the distinction that this is something that will be valuable in the future when we train on high-fidelity data.

Comment 6

Line 8: "simulated wind farms": I recommend that, for clarity, it is mentioned explicitly that it concerns PyWake or, more generally, a low-fidelity steady-state engineering wake model.

Response: The suggestion has been added menitioning PyWake and explaining it is a low-fidelity steady-state engineering wake model.

Comment 7

Line 10: "underestimated ... wake effects". For clarity, add: "compared to the simulated values".

Response: This has been added.

1. Introduction

Comment 8

Line 61: Many WES readers will not be familiar with Graph Neural Operators (GNOs). Since GNNs are introduced earlier and better known, consider briefly describing how GNOs relate to GNNs and why they may yield improved performance.

Response: Thank you for this suggestion. We have added a brief explanation clarifying the relationship between GNOs and GNNs.

Comment 9

Line 74: Use "Section" (capital S). In general, ensure consistent capitalisation and punctuation throughout the manuscript (e.g., figure 2, line 121, line 129, line 140, line 209, line 240, line 320, line 404, line 416, ...)

Response: We have corrected the Section issue and have made the correction to the WES Copernicus style "Sect." now, except for when starting a sentence where it is referred to as "Section". We are not exactly sure what the issues around Figure referred to in the lines, qua the WES guideline "Figure" if it occurs as the first word and "Fig." inside the text. We have cleaned up some inconsistencies in this formatting now and believe it is done correctly.

2. Methodology

Comment 10

Line 78: "Here, ... ". I suggest improving the formulation. (at multiple occasions in the manuscript)

Response: We agree that this formulation has been used much especially in the Methodology section. We have therefore rephrased Line 78 and line 275. (Some similar occurrences has been moved to the appendix, due to input of Reviewer 2)

Comment 11

Line 104: Alternative formatting: R_0^+

Response: Updated, that looks better.

Comment 12

Line 120: Should appear on line 119.

Response: Fixed.

Comment 13

Lines 121 – 126: Difficult to follow; please rephrase for clarity.

Response: The order of the information in the paragraph has been reworked, and split into two parts. It now reads:

"Correlated inflow conditions are generated using Quasi-Monte-Carlo sampling with the improved Sobol sequence (Joe and Kuo, 2008). While Dimitrov et al. (2018) employed the Halton sequence for similar purposes, we tested both approaches and found that the Sobol sequence, also used by de Santos et al. (2024), better captures extreme values in the distribution tails.

Figure 1 (e) illustrates the two-dimensional quasi-random samples from the Sobol sequence before transformation. The sampled component used to generate the free-stream velocity U is denoted p_U, and p_{TI} denotes the component used to generate the ambient TI."

Comment 14

Line 130 - : This section presents extensive detail on selected model parameters. Clarify which parameters materially affect results. Non-essential implementation details may be moved to an appendix. Additionally, because not all readers are familiar with PyWake, please summarise its key assumptions (steady-state flow, homogeneous inflow, no yaw misalignment or curtailment, etc.). Also, clarify whether the same CT- curve is used for all simulations, and discuss implications for generalisation. (As you may know, examples in literature exist attempting to integrate the CT-curve as node features in the GNN).

Response: We agree that too many details was given here and as we use default parameters this much emphasis in not necessary. We have therefore decided to move these details to the appendix, as also suggested by Reviewer 2 (Comment 1). In addition we have summarised the key assumption for unfamilar readers now.

We have added the following paragraph:

" PyWake is an open-source steady-state engineering wake modeling framework that computes wake deficits and turbine interactions. The framework supports spatially varying inflow conditions, multiple turbine types, and yaw misalignment via wake deflection models. However, to establish a baseline dataset, several simplifications were adopted: (i) homogeneous inflow with uniform free stream velocity and turbulence intensity; (ii) a single turbine type (DTU-10-MW) with a fixed C_{T}-curve; and (iii) all turbines aligned with the inflow direction. These choices reduce the input parameter space but may limit applicability to scenarios involving heterogeneous inflows, mixed turbine fleets, or active wake control."

Comment 15

Line 133: Citation formatting issue.

Response: We have split the citation to be explicit about its interpretaion. It now reads: The wakes are modeled with the updated self-similar Gaussian single wake deficit model NiayifarGaussianDeficit by Niayifar and Porté-Agel (2016), which is a further development of the Gaussian wake model by Bastankhah and Porté-Agel (2014).

Comment 16

Line 133: "This model was chosen due to its relative simplicity". Does this have any influence on the generalisation of the results of the manuscript? Please make this explicit in the manuscript.

Response: The sentence was changed to highlight the TI dependence rather than simplicity. "This model was chosen for its TI-dependent wake expansion to ensure a coupling between TI and velocity deficits."

We have also added the following text after introducing the numerical schemes, to better capture the nature of the design choices for the dataset generation, the changed also address comment 20:

"However, the use of All2AllIterative also benefits the GNO application: Because turbine interactions are resolved iteratively in both directions, the resulting input-output mapping constitutes a non-linear operator, providing a more challenging and representative test case for the GNO than a simple PropagateDownwind scheme. Although PyWake employs linear wake superposition (Eq. 2), the coupled iterative solution introduces non-linearity that the GNO must learn to approximate. More broadly, the dataset was designed to capture diverse physical effects, including blockage, turbulence-dependent wake expansion, and bidirectional interactions, rather than to maximize fidelity to any single high-accuracy model. Such diversity ensures that the GNO learns a sufficiently complex operator, demonstrating its capacity to generalize across varied flow physics. Fu- ture work should investigate the impact of both different superposition methodologies and higher-fidelity training data on GNO performance."

Comment 17

Line 135: "The authors fitted the model to a LES." Why is this relevant here?

Response: This has been reformulated and moved to the appendix as additional details for interested readers.

Comment 18

Line 143: "...model...chosen for its simplicity". Does this have any influence on the generalisation of the results of the manuscript? Please make this explicit in the manuscript.

Response: The direct reference to being chosen for simplicity has been removed as it is not fully accurate. With regards to generalisation see reply to comment 16

Comment 19

Line 145: When "TI" is used as a variable, the formatting shall be that of a variable. (This comment is applicable throughout the manuscript.)

Response: Thank you for the comment.

We have updated TI in equations and as a variable to be I_0 for ambient TI and I_a for added TI.

Comment 20

Line 170: "A linear sum is used for the wake summations." Here, it could be stated/repeated that the GNO theoretically may also handle non-linear superposition, but this may be verified in future work.

Response: We have updated the explanation of the use of linear-superposition slightly further down in the text now, to clarify some of this and acknowledge that further studies are needed. See reply to comment 16

Comment 21

Line 179: "The GNO is gird invariant." Begin a new paragraph.

Response: We have reworked the sentence to start a new paragraph here.

Comment 22

Line 180: "...from flow away from..." ?

Response: This was addressed in rewrite mentioned in comment 21.

Comment 23

Line 186: $\bar{x}$ could be interpreted as a vector or maximum; consider using: x'

Response: We agree that the overline was an unfortunate choice, but we have reserved the use of the apostrophe for neural network outputs. Therefore, we have instead changed to using $\tilde{x}$ which is more commonly used for coordinate transformations than the original overline.

Comment 24

Figure 3: Clarify why the coordinate origin differs from the farm centre. How the coordinate origin is determined? Does this influence (the features of) the GNO?

Response: The layouts are created with the PlayGen and are not necessarily centered during generation, hence neither will the flow maps be. It doesn't change anything because the GNO is only concerned with the relative internal distances between nodes. The distance formulation has been clarified in response to Reviewer 2's comment 3.

Additionally a sentence has been added to the describe that it is a non issue: "As an artifact of the PlayGen generation methods, the wind farm center and coordinate center do not always coincide. Because the GNO uses relative internal coordinates, this does not affect its performance. A sentence explaining this has been added and more details are given in Sect. 2.2"

Comment 25

Lines 191- 193: Appear irrelevant; consider removing.

Response: We believe most readers of WES will not be familiar with graph theory graphs and therefore we believe having a bit of historical background makes the introduction slightly more elegant and engaging for the reader.

Comment 26

Line 198: "Instead, they are copied to each node as node features." Why did you make this decision? provide references if applicable.

Response: We have added the source of inspiration, to this sentence.

Comment 27

Line 206: Consider adding $V = V_{\mathrm{wt}} \cup V_{\mathrm{p}}$

Response: Added, thank you for the suggestion.

Comment 28

Line 209: Consider adding the analogous equation to Eq. 7.

Response: Added, thank you for the suggestion.

Comment 29

Line 209: "The inter-turbine edges...": start a new paragraph for readability.

Response: We have added a new paragraph.

Comment 30

Line 204: "Probe node...": start a new paragraph for readability.

Response: We believe this was meant to refer to line 215 not 204. We have made the change based on that assumption while also addressing comment 31. So that it now reads:

" Probe node connectivity is straightforward because each probe connects to all wind turbines. Inter-turbine connectivity is more complex because it requires an algorithmic approach to determine which turbines should be connected. Several algorithms exist for constructing the connectivity of the turbine nodes. Duthé et al. (2023) investigated four graph connectivity schemes: Delaunay triangulation (Delaunay, 1934; O'Rourke, 1988; Fey and Lenssen, 2019), K-Nearest Neighbors (KNN), a radius- based method, and a fully connected scheme. They concluded that Delaunay triangulation provides the best balance between accuracy and computational performance. Accordingly, we also adopt Delaunay triangulation in this work to derive inter-turbine connections. "

Comment 31

Line 216: "... it uses Delaunay triangulation..." This is repeated in line 220. Consider rephrasing.

Response: Addressed in reply for comment 30.

Comment 32

Line 224: You state that Cartesian coordinates are used as edge features (and not as node features, which intuitively would make more sense), which seems to contradict later text. Please clarify and justify the design choice for the node and edge features; add references if similar implementations exist. Also specify how the coordinate origin is chosen, as it has an influence on the Cartesian coordinates.

Response: The concerns addressed in this comment has been addressed in replies to comments 24, 26 and reviewer 2 comment 3.

The main summary being we had not clearly managed to describe the actual behavior of the model and that we now have put more emphasis into accurately describing that the GNO uses relative positions between nodes, and that the center of the graph is therefore not of importance.

Comment 33

Lines 230 – 235: The sequence and interaction of the models are unclear; Consider rephrasing.

Response: We rephrased it to first introduce the GNO and then introduce the developments of the GNO.

Comment 34

Line 239: Clarify "common" and "abstraction".

Response: This has been rephrased to:

"The encoder-approximator-decoder configuration is a widely adopted architectural pattern in GNNs, providing a modular framework for transforming input graphs into predictions (Battaglia et al., 2018)"

Comment 35

Line 240: "initial two stages": Clarify whether these are the encoder and approximator?

Response: It is, it has been updated to: "the encoder and approximator stages"

Comment 36

Line 245: "sped up"

Response: We have interpreted this as preferring a more formal language. The sentence has been changed to:

"This means that predicting in a flow field incurs lower computational cost than a fully integrated prediction scenario."

Comment 37

Line 248: "Fore instance, in fully couple formulations, . . .": is this the case in your implementation?

Response: Indeed, it is. We believe the confusion around this has been addressed in the reply to comment 16, outlining the meaning of the All2AllIterative scheme.

Comment 38

Lines 256 – 260: This information deserves a separate subsection somewhere else in the manuscript.

Response: We have moved it into Section 2.4 Training and evaluation as its own subsection.

Comment 39

Line 267: "Q": here a capital letter, in contrast to in line 278.

Response: Thank you for noticing, we have updated line 278 to use all capital letters.

Comment 40

Line 278: The equation comes back (and better) in lines 292 – 294

Response: We have corrected the one in line 278, see previous reply. The reason they come back is that in line 278 it is intended to introduce MLPs in general, and the next time we see the actual dimensions, the considered MLPs transform.

Comment 41

Equation (10a): k = 1,. . .,K; Move be on the same line.

Response: This is done in preparation for the final format using two columns, in which there is not enough space for the second part.

Comment 42

Lines 292-294: For clarity, repeat here the numeric values in your implementation for |E|, fe, . . ., and Q.

Response: We agree keeping track of all the dimension are quite difficult. We have added the following:

"For convenience, the key dimensions are restated here: the cardinalities $|V|$ and $|E|$ vary with the wind farm configuration; $f_v = 2$ and $f_e = 3$ are the initial node and edge feature dimensions; $K = 9$ is the number of RBF kernels; and Q is the latent-space dimension."

Comment 43

Line 300 and equation 12b: Shouldn't this be $\phi_{h'}$? (ref. notation in Figure 5)

Response: You are correct, there is a mistake here. The issue was Eq. 12b it has been corrected to $\phi_{h'}^{(m)}$

Comment 44

Line 309: Define $\widehat{x}_j$

Response: Thank you for the comment.

A more thorough and consistent introduction of this variable has now been implemented; it has also been renamed $\widehat{x}_q$ as to better fit the established Q for latent-space dimension:

" In this work, Softmax aggregation is used as $\rho_{e\to v}$, it uses Softmax to scale the latent space of each feature dimension independently. The incoming messages to node i from its $|\mathcal{N}(i)|$ neighbors form a matrix of shape $|\mathcal{N}(i)| \times Q$. For each feature dimension, let $\hat{\boldsymbol{x}}_q \in \mathbb{R}^{|\mathcal{N}(i)|}$ denote the vector containing the q-th feature value across all neighbors.

$$\rho_{e\to v} = \sum_{\hat{\boldsymbol{x}}_q \in \mathcal{X}} \frac{\exp(\hat{\boldsymbol{x}}_q)}{\sum_{\hat{\boldsymbol{x}}_r \in \mathcal{X}} \exp(\hat{\boldsymbol{x}}_r)} \cdot \hat{\boldsymbol{x}}_q \tag{1}$$

where $\mathcal{X} = \{\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_Q\}$ is a collection of neighborhood features to be aggregated independently. "

Comment 45

Figure 7: Nothing appears marked yellow.

Response: The wind turbine nodes after the Approximator ($\mathcal{A}$) are marked in yellow. We have added an additional visual queue at the center to make it look slightly more like a rotor/nacelle assembly.

Comment 46

Equation 18b: To be a relevant performance measure for accuracy, the MAPE of the predicted wind speed deficit should compare the predicted wind speed deficit relative to the ground-truth wind speed deficit (instead of comparing to the free inflow wind speed).

Response: As discussed in the response to comment 2 MAPE has limitations, but we agree that referring to the metric we used as MAPE is incorrect and have changed the naming to MANE.

Comment 47

Line 371: "limiting factor has been the 72-hour walltime": Was the performance the GNO still improving at the end of this walltime? If yes, why could the model not be saved to resume the training of the model, starting a new walltime?

Response: Indeed the model were converged, this is not relevant information and have therefore been removed.

Comment 48

Line 385: Grammar issue.

Response: "Is" corrected to "are"

Comment 49

Line 397: Please explain more clearly.

Response: The following explanation has been added:

"Where the term $(1 + f_e)$ accounts for the fact that each edge stores both its f_e features and an index pair identifying the connected nodes in the adjacency list representation."

Comment 50

Line 400: "To evaluate the model under realistic conditions." This suggests that the other layouts would not be realistic. Consider rephrasing.

Response: Indeed, thank you for pointing that out. We have changed it: "To evaluate the model on an established benchmark, the IEA Wind 740-10-MW reference wind farm is employed."

Comment 51

Line 401: "farm layout"

Response: Corrected to farm layout.

Comment 52

Line 405: "simple operation": Does the applied power curve not depend on TI? If yes, how is TI at the turbine locations predicted by the GNO?

Response: No, the CT curves in PyWake do not include TI dependence by default. TI affects the wake expansion through the wake deficit model, which augments the resulting velocity deficit and ultimately the effective windspeed used to determine the power and CT state of turbines operating in wakes.

3. Results and Discussion

Comment 53

Line 419: "internal MLP": Clarify what is meant by "internal".

Response: The following has been added for clarification:

"For the model configuration, the encoder, wind turbine processor, and probe processor MLPs are collectively referred to as the internal MLPs."

Comment 54

Line 229: "probes per graph": how are these selected? Randomly in the large box around the wind farm?

Response: During training, probe locations are selected using uniform random sampling across the bounding box surrounding each wind farm. This choice was made to ensure broad coverage of the domain without introducing additional hyperparameters. The revised manuscript now clarifies this in the methodology section. Please also see our response to Reviewer 2, Comment 10, for further details.

Comment 55

Line 429: "number of batched graphs": Do you mean, the number of graphs per batch?

Response: Yes, it has been corrected

Comment 56

Line 436: "increases the confidence that the right model is chosen": Why?

Response: MSE and MAE weight errors differently: MSE penalizes larger errors more heavily due to the squaring, while MAE treats all errors linearly. When both metrics rank models similarly, it suggests that the best-performing model is not merely minimizing a few large errors at the expense of many smaller ones (or vice versa), but is performing consistently well across the error distribution. This consistency across different error weightings increases confidence that the model selection is robust and not an artifact of the chosen metric.

The following text has been added as an explanation: "Since these metrics weigh errors differently, this increases confidence that the model selection is robust and not overly sensitive to the choice of metric."

Comment 57

Table 2: "lowest best metrics marked with b": What is meant by that?

Response: This was a typo it has been corrected to bold, although this table has been moved to the Appendix.

Comment 58

Table 2: Add the unit of each performance indicator.

Response: We have added the units for the performance indicators, although this table has been moved to the Appendix.

Comment 59

Table 3: Add the unit of each performance indicator.

Response: We have added the units for the performance indicators.

Comment 60

Table 3: The values of the performance indicators for the test set seem to be of another order of magnitude compared to those of the validation set. How is that possible?

Response: We had accidentally forgotten to unscale the metrics relating to the training; hence, the originally reported metrics were dimensionless for the training metrics and had units for the test metrics. This has been corrected, and now all the metrics are in the correct unscaled form.

Comment 61

Table 3: Consider adding baseline metrics for naïve deficit profile(s) (such as the zero-deficit profile), so that the magnitude of the metrics for the GNO models can be better interpreted.

Response: We have added a "naive" baseline now. Thank you for the suggestion, you are right, this makes the metrics more meaningful in a general sense.

Comment 62

Line 455: Include a wake profile(s) inside the farm (i.e., for $\bar{x} < 0$), so that not only the far-wake profile is analysed, but also the near-wake modelling is analysed.

Response: A new figure (Figure 12) has been added showing full flow map comparisons between GNO and PyWake, including the near-wake region within the farm. This figure also includes error plots to highlight where the model falls short. See also our response to Reviewer 2, Comment 11.

Comment 63

Line 461: Similar to one of the main comments: What is the range of the ground-truth velocity deficits relative to U? This may also be small.

Response: Thank you for this clarification request. The 2-5% error range reported in the original manuscript refers to errors in effective wind speed relative to U, not errors in the velocity deficit itself, as you correctly have observed.

Using the target deficit $\Delta\widehat{u}$ as a reference has proved problematic because it can be zero, leading to an explosion in the relative error. However, this is not the case for the relative error with respect to the target effective wind speed $\widehat{u}$, because the velocities of Fig. 9 (now 8) are not inside the farm area where the non-physical zero effective wind speeds can occur. Hence, we have updated the text to use $\widehat{u}$ relative errors instead.

Please note that the farms considered in this Figure in the revised manuscript have changed from the original submission, as a new set of wind farm examples was selected to improve the visual clarity of Figure 9 (see response to Comment 13 from Reviewer 2).

Comment 64

Figure 9: The plots are too small to be readable on a printed version of the manuscript.

Response: We have made significant changes to improve the visual clarity of Figure 9 (see response to Comment 13 from Reviewer 2).

Comment 65

Figure 9: Comparing the close results of dashed and full lines with different colours (and thus different wind speeds), at first view, one could think that there has been a failure with the allocation of the colours.

Response: We believe the changes to Figure 9 has addressed these issues (see response to Comment 13 from Reviewer 2)

Comment 66

Figure 10: It would be interesting to see the plots for the 3 wind speeds separately, or for the wind speed 12ms-1 only (i.e. wind speed with the highest wake losses), in case of lack of space (or in the appendix).

Response: We have added the requested figure in a new Appendix F now.

Comment 67

Figure 11: It looks as if the orientation of the wake is rotated from the main axes of the farm layouts in Figure 8.

Response: Thank you for pointing this out, you are correct we had incorrectly performed the rotational coordinate transformation for creating the Power rose. This has been corrected and to make the connection between the two plots clearer we have added a North pointing arrow to Figure 8.

Comment 68

Figure 11: Make plots larger to be readable in printed A4 version.

Response: We have updated the layout of Figure 11 to make the power roses more readable. The narrow nature of the figure is in preparation for the two column format of WES.

Comment 69

Line 481: "conditions similar to their training data": Are you suggesting that the optimised layout is less similar to your training data? If that were known, it would be recommendable to add better layouts to your training data.

Response: Thank you for this observation. This is an implicit limitation in our training data distribution. We have added the following explanation and mentioned it as an idea for further work.

"The procedurally generated training layouts do not include simple regular-grid configurations, which may explain the slightly lower accuracy on the regular IEA layout. Including such layouts in future datasets could improve generalization to such farms."

Comment 70

Line 485: In order to validate whether the accuracy of the proposed method is sufficient for optimisation applications (which is the stated motivation for implementing this GNO model), it would be very interesting to compare the AEP predictions made on the basis of PyWake and the GNO, for both wind farm layouts.

Response: Thank you for the suggestion, as discussed in our response to Comment 3, the manuscript has been revised to emphasize neighboring farm wake assessment rather than direct layout optimization as the primary application. However, we have added the requested information on the plot for interested readers.

Comment 71

Line 485: For (farm layout) optimisation problems, the absolute value of the wake losses may be less important than the relative difference between for different layouts. Therefore, consider adding measures about the relative wake power losses predicted by the GNO.

Response: See response to comment 70.

Comment 72

Line 530: I understand your choice to compare computational cost on a one-core CPU. Nevertheless, as currently the GNO can be run on 32 cores simultaneously and on a GPU, in contrast to PyWake, in my opinion, you may also add this comparison in the manuscript. Indeed, that would be the reality for someone who may have to choose between PyWake and the surrogate model.

Response: Thank you for the suggestion. However, we have chosen to retain the current comparison for the following reasons: First, the reported timings are expressed in CPU hours, which inherently accounts for the degree of parallelization, providing a hardware-agnostic measure of efficiency. Second, implementing a fair multi-CPU comparison for PyWake would require careful consideration of parallelization strategies (e.g., parallelizing across wind directions, layouts, or grid points). Without confidence that such an implementation is optimal, we risk misrepresenting the performance of PyWake. We believe it is important to be fair to its authors and avoid any comparison that could inadvertently understate its capabilities. The trained model weights are publicly available, enabling readers to evaluate performance on their own hardware configurations.

4. Conclusion

Comment 73

Line 553: "fresh". Consider alternative wording.

Response: Agreed, changed to "offers a new perspective"

Comment 74

Line 555: ‘afterthought", "established literature": What is meant here?

Response: This was slightly unclear, it was supposed be in relation to single wake surrogates. This has been rephrased to:

"It has been established as a novel approach inspired by classic engineering models, demonstrating how the superposition principle can be integrated directly into the learning process, rather than relying on algebraic wake superposition as single-wake surrogate models require."

Comment 75

Line 559: "In section 3": leave away

Response: Rehrased to "The GNO was evaluated, . . . "

Comment 76

Line 562: "The GNO is less accurate that PyWake." You did not prove that, but it is trivial. See main comments.

Response: We believe we have addressed the concern in the replies to the main comments.

Appendix A

Comment 77

Line 601: "cf. branch input, trunk input": Please clarify these terms.

Response: The context for the GNO has been added and litterature for a broader context has been suggested:

"As with other neural operators, the data comes in triplets: branch input (the wind farm graph encoding turbine positions and inflow conditions), trunk input (the probe graph encoding query locations), and target output. For a broader context, see e.g. (Lu et al., 2021; Seidman et al., 2022)."

Concluding Response

We thank Reviewer 1 for the comprehensive feedback. The review identified several important areas for improvement, and we believe the revised manuscript is stronger as a result. We have endeavored to address every comment and hope the revisions meet your expectations. We remain open to further suggestions if any concerns persist.

Authors' Response to Reviewer 2

General Comments. The authors introduce a Graph Neural Operator architecture for wind farm wake flow prediction, where turbine-turbine interactions are learned through message-passing on a graph and flow velocities can be queried at arbitrary probe locations. Trained on ~21,000 PyWake simulations spanning diverse procedurally-generated layouts, the model generalizes across farm configurations and seems to achieve low RMSE in the predictions while being approximately 10x faster than the engineering baseline (PyWake). In general, I find that the manuscript is well written and comprehensive: all components of the framework, from procedural dataset generation to GNO architecture and training, are described in sufficient detail to enable independent implementation. The breakdown of the results is also very detailed, and I appreciate the thorough reporting on computational aspects. I recommend minor revisions, with specific comments below.

Thank you for your assessment. We believe the comments are fair and constructive and have sought to address them accordingly.

Section 2

Comment 1

A lot of detail is provided for the models that are used by PyWake to generate the training data. I think some of this can be moved to the appendix.

Response: We have moved velocity, TI and Blockage model definitions to the appendix as many WES readers will already be familiar witht these. Addionally an explanation has been added to explain the use of All2AllIterative and its implication for the GNO

Comment 2

The part about the additional coordinate system is a bit confusing and could do with some extra clarification.

Response: We have simplified the explanation and reformulated it as: "Figure 3 illustrates the bounding box of a wind farm. A farm downstream axis, $\widetilde{x}$, is introduced to measure the distance behind the most downstream turbine, making it easier to compare results across different wind farm layouts". In response to Reviewer 1's comment 24, the coordinate has also been changed to $\widetilde{x}$

Comment 3

Why did you choose to use the coordinates of the sender nodes as an edge feature, instead of the relative coordinates (vector coordinates) between the the two connected nodes, as is typically done in other GNN work?

Response: We do use relative coordinates, we have just failed to report it correctly. The following correction has been added: The edges store the relative node positions (x_{ij} and y_{ij}), and Euclidean distance (d_{ij}) between connected nodes:

$$(x_{ij}, y_{ij}, d_{ij}) = \left(x_j - x_i,\, y_j - y_i,\, \sqrt{x_{ij}^2 + y_{ij}^2}\right)$$

Comment 4

I'm not sure I see the advantage of having the decoding stage setup as it is. I get that you want to first freeze the turbine node latents, so that you have the option of adding any amount of probe nodes after, but why not have undirected probe edges between a probe and other probes, as well as between a probe and the turbine nodes. I guess the current setup makes sense when only using a single probe message-passing step, but why not have multiple steps during this phase? It would increase performance (albeit with extra computational costs). Please consider clarifying this.

Response: The current design is inspired by classical engineering wake models, where the flow field is reconstructed by superposing individual turbine wake contributions without propagation between field points. Probe nodes serve an analogous role: they aggregate wake effects from all turbines but do not exchange information among themselves, as they are not wake-generating entities.

We agree that introducing probe-to-probe message passing could potentially capture additional spatial correlations. However, this would likely require higher-fidelity training data (e.g., LES or RANS) to be meaningful. This has been noted as future work in the revised manuscript.

Comment 5

Could you have not resumed training after 72 hours? Or were the models sufficiently trained by this time and it was not needed?

Response: Indeed the model were converged, this is not relevant information and have therefore been removed.

Comment 6

I'm not sure the MAPE metric is correct, as typically we would divide the error by the target value. In some sense, what you show here is more of a MAE that is normalized by the inflow velocity, which may still be a useful metric.

Response: You are correct, the deviation from the standard was actually chosen delibarately but has been the cause of much confusion. Therefore we have chosen to rename our version to MANE and the following has been added to the metrics section explaining the choice.

"While most metrics are standard, MANE is a custom variation of Mean Absolute Percentage Error (MAPE). Standard MAPE normalizes by the target value, which becomes problematic when targets approach zero as can occur near heavily waked turbines where wake superposition overestimates the deficit. Normalizing by the wake deficit instead would cause similar issues in unwaked regions. To avoid this, MANE normalizes by the free-stream velocity, providing a more robust metric that still enables comparison across different inflow conditions."

Comment 7

You mention the cardinality of the graph, is this actually used somewhere?

Response: Yes it is used when the different components of the GNO are introduced and the dimensionality of the inputs and outputs are defined e.g. for the RBF functions

$$\varphi_e \ : \ \mathbb{R}^{|E|\times f_e} \ \rightarrow \ \mathbb{R}^{|V|\times f_e K}$$

Section 3

Comment 8

You could consider moving some of the parameter tables for the hyperparameter study to the appendix.

Response: We agree that the information was slightly overwhelming and have moved Tables (1) and (2) into the Appendix to become Tables (D1) and (D2). Instead, a new Table (1) has been introduced containing just the information of the 5 best models.

Comment 9

No ablation on the number of message-passing steps (M=3 throughout) is performed. This seems like an important hyperparameter to study, given that message-passing propagates information only to immediate neighbors. Would increasing the number of steps increase performance for cluster farm setups, where wake superposition seems to be the biggest hurdle?

Response: We did some intial experimentation prior to our grid searches and the impact of amount of message passing steps was not found to change the results significantly in our setup. This could be due to many factors, and this should be investigated more thoroughly in the future. We have added the following in the manuscript:

"The number of message-passing steps (M) was fixed at 3 throughout the grid search rather than treated as a tunable hyperparameter. Initial testing on a simpler model showed low sensitivity to additional message-passing steps, suggesting diminishing returns beyond $M = 3$. While increasing M could theoretically improve performance in densely clustered farm configurations by propagating wake interactions across more neighbors, preliminary experiments did not support this hypothesis. However, the optimal value of M is suspected to be data-dependent; higher-fidelity datasets capturing more complex wake dynamics or turbulence interactions may benefit from additional message-passing steps to fully resolve inter-turbine dependencies. A systematic sensitivity study on M across diverse farm configurations and data fidelities should be considered for future work."

Comment 10

The training procedure regarding probe node sampling deserves clarification. How are the probe locations selected? Random uniform sampling across the domain, or weighted toward regions with stronger wake effects? This choice likely influences model performance in different flow regions.

Response: For the models documented in the manuscript the probe node sampling during training is strictly random uniform sampling. An explanation has been added to the manuscript in the methodology section about training.

Here included as bonus information, early in the process we experimented with some weighed sampling. However, the amount of parameters became too overwhelming and some simplifications had to be made and this was one of them. However, if someone should be interested in implementing this we were considering this weighting for the domain (although some of the parameters should be updated slightly):

$$p_{\text{linear}}(x) = \begin{cases} 1 + k \cdot (x - 20D) & x < 20D \\ 1 & 20D \leq x \end{cases} \quad , \qquad k = \frac{1}{x_{\max} - x_{\min}}$$

$$p_{\text{cross}}(y) = 1 - \tanh\left(k \cdot y^2\right) \quad , \qquad k = \frac{2}{(y_{\max} - y_{\min})^2}$$

$$p_{\text{circ}}(x, y) = 1 - \frac{1}{1 + k \cdot ((x - x_{\min})^2 + (y - y_{\min})^2)} \quad , \qquad k = \frac{100}{(x_{\max} - x_{\min})^2 + (y_{\max} - y_{\min})^2}$$

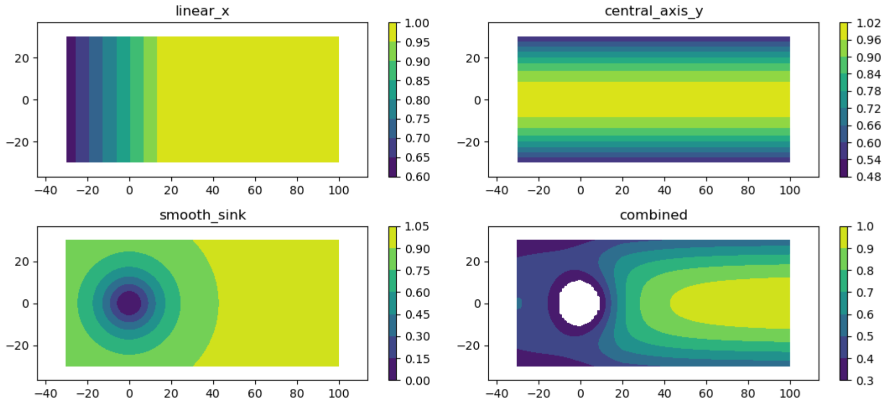

Figure 1: Sampling weights (p)

$$p\,(x,y) = \begin{cases} 0.3 & p_\sigma < 0.3 \\ p_\sigma & 0.3 < p_\sigma < 1 \\ 1 & 1 \leq p_\sigma \end{cases} \quad , \qquad p_\sigma(x,y) = p_{\text{linear}}(x) \cdot p_{\text{cross}}(y) \cdot p_{\text{circ}}(x,y)$$

Comment 11

I would have really liked to see a complete flow map comparison between the GNO's output vs PyWake (along with a difference plot), especially in the near-wake region. It would be extremely useful to understand where the model falls short. You could do this for the IEA Wind 740-10-MW reference reference farms for instance.

Response: We have now added a new Figure 11, that demonstrates the limitations of the model and have added some additional text.

Comment 12

This point is optional, but I was wondering what the learned RBF kernels look like. Showing the learned RBF distributions after training would provide insight into what spatial scales the model finds important.

Response: We have moved the existing RBF figure to the results and extended it with the trained Kernel and overview of the kernel values. And a short explanation of whats in the figure. The figure is numbered 13 now.

Comment 13

Figure 9. I think that you can plot the farm in a nicer way, it is hard to actually see the positions of the turbines.

Response: We have improved Fig. 9 (now Fig. 8) significantly, the graphs has been made more visible by removing empty space in the plots. This has also been applied in the velocity deficit plots. In the process a new set of graphs was chosen randomly. The text and Fig. 10 (Now Fig. 9) has been updated accordingly.

Comment 14

Figure 12. In general I find this plot a bit hard to read, the color scheme can be improved and the bars made a bit wider. For subplot (a): the KDE extends into negative RMSE values, which is not physically meaningful. Consider using a boundary-corrected KDE method, or simply truncating/reflecting at zero.

Response: We have decided to make this a two column format figure now to enable wider bars, we have also changed the order of appearance slightly as to make better use of the new format. With regards to the colors we have made a change for better visual assesment while still avoiding red-green colorblindness issues. Your assesment of the KDE was correct, and we have decided to forego it and instead use a simpler binned error count and show the count instead. The KDE has also been removed from the methodology section.

Section 4

Comment 15

I would suggest more measured language that emphasizes the nice methodological contribution without overclaiming impact ('breakthrough').

Response: Agreed we have toned down the impact and adopted an assessment more in line with the methodological contribution as suggested. We also changed a line in the abstract mentioning this.

Minor Technical Comments

Comment 16

Line 120, line 283: alignment issues

Response: We have fixed the issue.

Comment 17

Line 320: text overflow

Response: We have fixed the issue.

Comment 18

“Windfarm“ vs “Wind farm“ used interchangeably, try to stay consistent. Same for “flowmap“ vs “flow map“.

Response: We have alligned these to be consisitent now choosing two words.

Comment 19

Table 2: typo in caption (’b’ instead of ’bold’?)

Response: We have fixed the issue.

Concluding Response. We are grateful for your positive assessment of the manuscript and your constructive suggestions for improvement. Your recommendations have strengthened the presentation of our results. We also appreciate your suggestion to adopt more measured language regarding our contributions, which has helped better frame the methodological nature of this work. We have been able to accommodate your suggestions and hope the revised manuscript meets your expectations.

Manuscript with Tracked Changes

Graph Neural Operator for ~~windfarm~~ wind farm wake flow

Jens Peter Schøler[1], Frederik Peder Weilmann Rasmussen[1], Julian Quick[1], and Pierre-Elouan Réthoré[1]

[1]DTU Wind and Energy Systems, Frederiksborgvej 399, 4000 Roskilde, Denmark

Correspondence: Jens Peter Schøler (jpsch@dtu.dk)

Abstract. Wind farm flow simulations are computationally expensive. However, numerous simulations are often required ~~in applications such as wind farm layout optimization or when considering~~ to account for wake effects from multiple neighboring farms, which motivates the development of data-driven surrogate models. ~~However, most existing~~ Such surrogates may also enable the consideration of neighboring farm wakes during layout optimization.

Most existing data-driven surrogate approaches rely on ~~classical~~ fixed algebraic superposition principles, which ~~constrain their ability to~~ may limit flexibility when extending to higher-fidelity data sources that capture nonlinear wake interactions. We propose a novel method that embeds a trainable and scalable superposition principle within a Graph Neural Operator (GNO) architecture, enabling the model to learn complex wake combinations directly from data.

The model consists of two sequential Graph Neural Network (GNN) layers: the first encodes turbine–turbine interactions into a latent representation, while the second combines these latent turbine states to predict the wind speed at a desired location. The GNO is trained on a large dataset ~~of simulated wind farms and achieves a low prediction error~~generated with PyWake, a steady-state engineering wake model. More than 2000 unique layouts were procedurally generated for training, with an ~~RMSE of 0.353 ms^{-1} and a MAPE of 0.938% on an unseen test dataset~~additional 998 for testing.

The GNO accurately identifies regions of strong wake interaction, although the spatial extent of wakes is slightly underestimated compared to the simulated values in cases with pronounced wake effects. Overall, the proposed GNO represents a ~~novel contribution~~ methodological advancement in data-driven wind farm flow surrogates, introducing a new conceptual framework inspired by established engineering wake modeling principles.

1 Introduction

Wind farm flow has been studied rigorously since engineers placed multiple turbines together in farms. The phenomenon of wind turbine-induced wakes is one of the most extensively studied subjects in wind energy, see, e.g., (Göçmen et al., 2016; Porté-Agel et al., 2020). Nevertheless, accurately representing the complex interactions among multiple wakes remains an active research challenge. In classical engineering wind farm flow models, the total flow field is obtained by calculating the operating state of each turbine and determining its corresponding wake contribution. The combined wind farm flow is then found through wake superposition. This step forms the core of most engineering models and significantly contributes to their overall performance.

Traditionally, wake superposition is performed using simple algebraic formulations based on the velocity deficit, either linearly (Lissaman, 1979; Niayifar and Porté-Agel, 2016) or quadratically (Katic et al., 1987; Voutsinas et al., 1990). While these formulations are computationally efficient, they rely on strong simplifying assumptions, in particular that wake interactions can be represented as additive. This means that critical physical processes such as wake mixing, entrainment, and interactions with the atmospheric boundary layer (Porté-Agel et al., 2020) are not adequately captured, which can lead to significant errors in the predicted farm flow. Recent studies have proposed modified superposition methods designed to enhance the physical realism of these models. The momentum-conserving superposition model by Zong and Porté-Agel (2020) introduces a weighted sum based on convective wind speeds, while the cumulative wake summation method by Bastankhah et al. (2021) enforces approximate mass and momentum conservation using an altered method of wake addition. These developments represent progress toward more consistent formulations, but they remain limited by their algebraic structure and cannot fully capture the nonlinear nature of wake interactions.

To overcome these limitations, more flexible and expressive operator formulations are needed, ones that can represent the inherently nonlinear and spatially coupled flow interactions occurring within wind farms. Machine learning offers a promising framework for this. Data-driven models can learn such complex wake interactions directly from data, without relying on restrictive analytical assumptions. In this context, graph learning provides a particularly suitable approach. By representing turbines as nodes and their aerodynamic couplings as edges, message-passing Graph Neural Networks (GNNs) can learn to propagate and combine wake information across the wind farm. This effectively generalizes the traditional wake superposition step into a learned, nonlinear operator that can capture the complex flow physics governing wind farm behavior. The graph-based formulation offers distinct advantages over alternative neural network architectures for this application. Convolutional Neural Networks (CNNs) require fixed grid structures and, therefore, struggle to accommodate varying turbine layouts across farms with different sizes and densities. Simultaneously, CNNs need for a fixed grid imposes an upper resolution limit, whereas a GNN-based approach enables inference at the exact positions of interest. Multilayer Perceptron (MLP) based surrogate models for single-wake prediction have been proposed, but these still require a classical superposition scheme to reconstruct the full farm flow, inheriting the limitations of algebraic wake summation. In contrast, the message-passing framework inherent to GNNs naturally represents turbine-turbine interactions through graph connectivity, enabling the model to learn meaningful turbine interactions, including nonlinear interactions, that generalize across diverse layouts without relying on explicit superposition. For a review on data-driven methods in wind farms fluid flow, see e.g. Zehtabiyan-Rezaie et al. (2022) .

GNNs have been successfully applied to similar problems in the past. Park and Park (2019) demonstrated a Physics-induced Graph Neural Network (PGNN) as an accurate and generalizable surrogate model for wind farm power estimation. Their method embeds engineering models into the network to learn physically plausible interactions, which they validated by applying it to a Wind Farm Layout Optimization (WFLO) problem. Bleeg (2020) presented a GNN trained on simulated Reynolds-Averaged Navier–Stokes (RANS) data capable of accounting for wake losses within a wind farm. Yu et al. (2020) trained a GNN using measurement data to superpose temporal states, although its applicability was limited to a single wind farm at a time. Ødegaard Bentsen et al. (2022) employed a Graph Attention Network (GAT) to predict individual turbine power production based on engineering-model data.

Duthé et al. (2023); de Santos et al. (2024) trained GNNs on data generated from engineering models capable of predicting loads and power. Duthé et al. (2024) further developed this model and employed transfer learning to enhance data fidelity using a limited amount of mid-fidelity data from Dynamiks, a further development of HAWC2Farm (Liew et al., 2023), which implements the Dynamic Wake Meandering approach. Li et al. (2024) trained a graph transformer model to predict farm-level power and applied it to a static yaw optimization task.

Li et al. (2022) proposed a unique type of GNN for wake flow prediction, leveraging the GNN framework to predict the flow behind a single turbine. Their configuration resembles that of Convolutional Neural Networks (CNNs) but uses the flexibility of graphs to operate on the unevenly distributed RANS mesh. They employed layered Graph SAmple and aggreGatE (GraphSAGE) blocks to sample neighborhoods and propagate information efficiently throughout the domain. The model by Li et al. (2022) stands out as the only one that attempts to predict the flow field within the domain, rather than solely at the turbine locations. However, since their model predicts the flow only behind a single turbine, it still relies on classical wake superposition methods to reconstruct the overall flow of the wind farm.

In this work, a new Graph Neural Operator (GNO) model is proposed, capable of predicting the flow over an entire wind farm, not only at the individual turbines. The GNO is a special variation of a GNN that utilizes two sequential GNNs: the first processes turbine interactions, while the second enables flow predictions at arbitrary locations throughout the domain.

The model is formulated based on the theoretical foundation introduced by Seidman et al. (2022) within the Nonlinear Manifold Decoders (NOMAD) framework. This approach allows the model to learn continuous flow representations in a physics-consistent manner while retaining the flexibility of graph-based learning. The development of the proposed GNO constitutes the main contribution of this paper. The GNO implementation is inspired by classical engineering models of wind farms, designed to predict spatially continuous flow fields across the entire domain. This extends the predictive capability beyond turbine-level quantities, enabling direct inference of flow fields from graph representations.

Additionally, the GNO is rigorously tested to assess its capability.

Furthermore, a secondary contribution is the development of a data generation pipeline that combines state-of-the-art engineering models with stochastic sampling to produce physically realistic and diverse training data.

The ~~structure of the article is~~ article is structured as follows: Sect. 2 introduces the employed methods and is divided into four subsections~~, concerning data~~: Data generation, graphs, the GNO, and performance metrics. Section 3 consists of two parts: In Sect. 3.1 a summary of the results of a grid search to determine suitable hyperparameters ~~are~~ is introduced, and in Sect. 3.2 the best performing model is rigorously tested and discussed. Finally, in Sect. 4 conclusions and suggestions for future work are summarized.

2 Methodology

~~Here,~~ This section introduces the methods used to construct the GNO, create training data, and evaluate the model~~are presented~~. In Sect. 2.1, the data generation process is described, covering random layout generation, inflow generation, and wind farm flow simulation. A general introduction to graphs is provided in Sect. 2.2, and subsequently, the proposed GNO is introduced in Sect. 2.3, including an overview of the neural network methods used to construct it. Finally, in Sect. 2.4, the methods for training and evaluation are described, along with the introduction of the performance metrics used.

2.1 Data generation

To ensure the dataset includes a representative subset of wind farm layouts and inflow conditions, these scenarios are procedurally generated. In total, 3,570 unique layouts are generated. Ten inflow conditions per layout were ~~chosen~~ selected to improve the neural network's ability to learn the correlation between ~~the~~ inflow conditions and the wind farm wake deficit. 2,072 of the generated layouts are used for training, 500 are used for validation during training, and 998 layouts are set aside for testing. That means that for training 20,720 data points are considered, 5,000 for validation and 9,980 for testing.

Layout generation

Plant Layout Generator (`PlayGen`) by Harrison-Atlas et al. (2024) is used to generate wind farm layouts. `PlayGen` creates four types of layouts: *Cluster*, which uses Poisson disc sampling to iteratively generate the wind farms by selecting an existing turbine, generating a random angle and distance, and place the new turbine if it satisfies the spacing constraints. *Single string* creates ~~a~~ linear turbine arrays with inserted breaks, applies cumulative correlated noise to y-coordinates, and rotates the entire string. *Parallel string* makes multiple linear strings with the same orientation and vertically offsets each string by 1 to 3.5 rotor diameters (D). Then it applies random horizontal shifts and rotates the full layout. *Multi string* distributes turbines across strings, each with a quasi-random independent orientation, and places them in a domain while checking for string spacing. Examples of these are plotted in Fig. 1 (~~a)–(d)~~a-d).

The layout type, number of wind turbines (n_{wt}) and the turbine separation factors (s_{wt}) are sampled according to the probabilities and distributions listed below:

– Farm type: Categorical distribution sampled with probabilities P_{farm}

$$P_{\mathrm{farm}} = \begin{bmatrix} P_{\mathrm{cluster}} \\ P_{\mathrm{single\ string}} \\ P_{\mathrm{parallel\ string}} \\ P_{\mathrm{multiple\ string}} \end{bmatrix} = \begin{bmatrix} 0.40 \\ 0.20 \\ 0.20 \\ 0.20 \end{bmatrix}$$

– Number of wind turbines $n_{\mathrm{wt}} \in \mathbb{Z}$ sampled with a Truncated Normal distribution TN

$$n_{\mathrm{wt}} \sim \mathrm{TN}(\mu = 60,\ \sigma = 60,\ \delta_{\mathrm{low}} = 20,\ \delta_{\mathrm{high}} = 100)$$

– Wind turbine separation factor ~~$s_{\mathrm{wt}} \in \mathbb{R}_{>0}$~~ $s_{\mathrm{wt}} \in \mathbb{R}_0^+$ sampled with a Truncated Normal distribution TN

$$s_{\mathrm{wt}} \sim \mathrm{TN}(\mu = 5D,\ \sigma = 3D,\ \delta_{\mathrm{low}} = 2D,\ \delta_{\mathrm{high}} = 8D)$$

where P are probabilities and subscripts indicate the layout type considered, μ is the mean, σ the standard deviation, δ_{low} the lower cut off and δ_{high} the upper cut off for the truncated normal distribution TN.

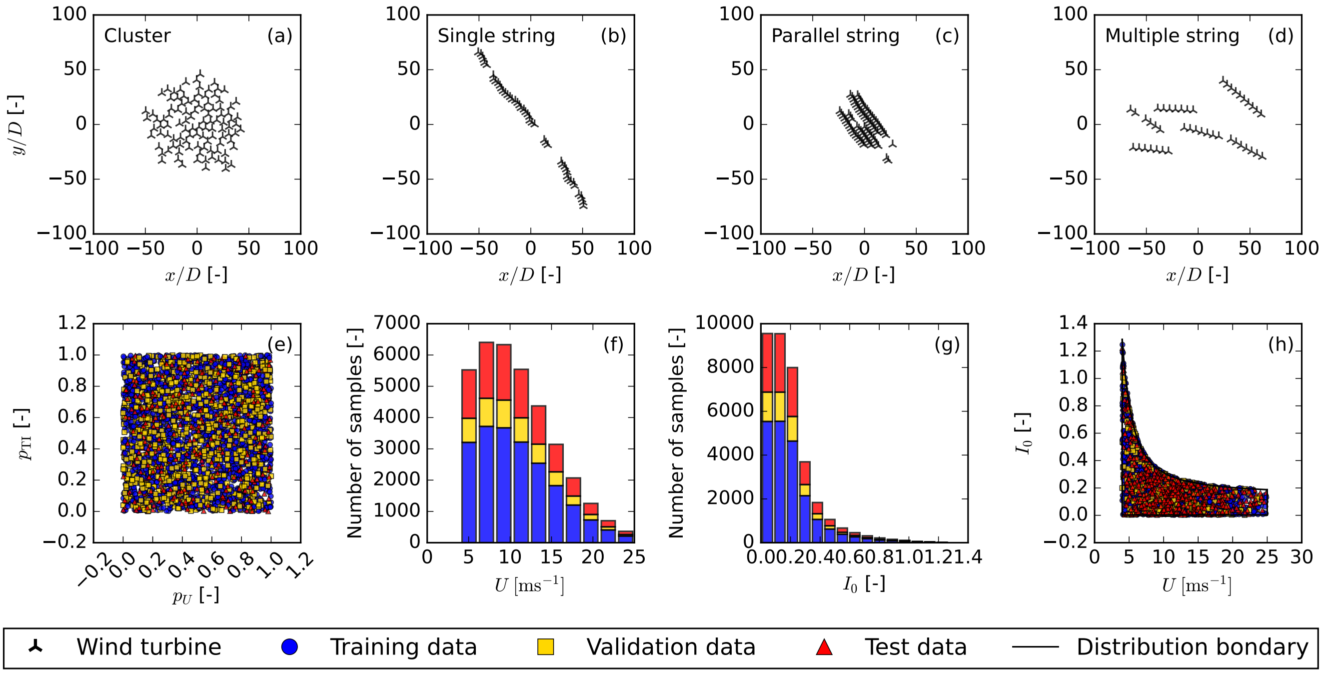

Figure 1. Procedurally generated data, (a-d) wind farm layouts using PLayGen. (e) Quasi random samples generated with the Sobol sequence. (f) U distribution. (g) Ambient TI distribution. (h) Generated U and ~~TI~~ I_0 with boundary.

Inflow generation

The inflow conditions required for the wind farm simulation are the ~~free stream velocity (U)~~ free-stream velocity and ambient Turbulence Intensity~~(TI); as~~ , respectively, in the variables U and I_0. As these are naturally highly correlated, it is necessary to consider ~~when~~ this when generating the flow cases~~are generated~~. The methods of Dimitrov et al. (2018) are used to generate correlated inflow conditions. After Dimitrov et al. (2018) was initially published, ~~the IEC-61400-1 standard (IEC 61400-1:2019 ed. 4)~~ IEC 61400-1:2019 ed. 4 has been updated with a slight change to the classifications of turbulence characteristics. Therefore, the new A$^+$ class is used with reference turbulence intensity (~~$\mathrm{TI}_{\mathrm{ref,A+}} = 0.18$~~$I_{\mathrm{ref,A+}} = 0.18$). Ranges of the ~~free stream~~ free-stream velocity are based on the DTU-10-MW reference wind turbine (Bak et al., 2013) with rotor diameter ($D = 178.3$ m), cut-in ($U_{\mathrm{c,in}} = 4$ ms^{-1}), rated ($U_{\mathrm{rated}} = 11.4$ ms^{-1}) and cut-out ($U_{\mathrm{c,out}} = 25$ ms^{-1}) wind

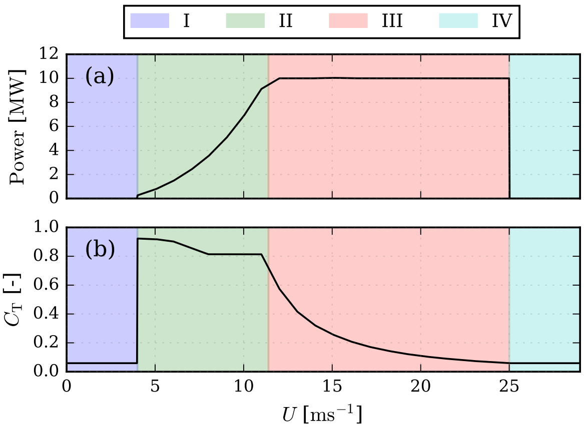

Figure 2. DTU-10-MW reference wind turbine (a) power curve and (b) C_T curve as implemented in `PyWake`. Operational stages I-IV are indicated with colors. I: below cut-in, II: Below rated power III: Rated power and IV: Above cut-out.

speeds. The power curve and coefficient of thrust (C_T) curves of the DTU-10-MW are displayed in Fig. 2. The velocity standard deviation (σ_u) lower and upper bounds follow the expressions in Dimitrov et al. (2018, Tab. 1). The expressions for the inflow bounds are given in Eq. 1.

$$4\ \mathrm{ms}^{-1} \le U \le 25\ \mathrm{ms}^{-1} \tag{1a}$$

$$0.0025 \cdot U \le \sigma_u \le \cancel{\mathrm{TI}}\, I_{\mathrm{ref,A+}} \left(6.8 + \frac{3U}{4} + 3\left(\frac{10}{U}\right)^2\right) \cancel{\mathrm{ms}^{-1}}\, [\mathrm{ms}^{-1}] \tag{1b}$$

~~To generate the coupled flow cases the~~ Correlated inflow conditions are generated using Quasi-Monte-Carlo sampling with the improved Sobol sequence ~~by Joe and Kuo (2008) is used for Quasi-Monte-Carlo with correlated dimension pairs. Dimitrov et al. (2018) uses~~ (Joe and Kuo, 2008). While Dimitrov et al. (2018) employed the Halton sequence ~~, while de Santos et al. (2024) uses the improved Sobol sequence. Both methods were tested , and it was observed~~ for similar purposes, we tested both approaches and found that the Sobol sequence, also used by de Santos et al. (2024), better captures extreme values ~~, so it was chosen for this work. In Fig.~~ in the distribution tails.

Figure 1 (e) ~~the components of~~ illustrates the two-dimensional quasi-random samples from the Sobol sequence ~~are illustrated prior to being projected through the target distribution. p_U corresponding to the component of the sample~~ before transformation. The sampled component used to generate the free-stream velocity U ~~and p_TI~~ is denoted p_U, and p_TI denotes the component used to generate the ambient ~~TI~~ TI. In Fig. 1 (f) the resultant wind speed distribution is shown. The distribution was obtained by projecting the wind speed sample onto a Rayleigh distribution following ~~IEC-61400-1 (IEC 61400-1:2019 ed. 4)~~ IEC 61400-1:2019 ed. 4 and scaled with the range in Eq. 1a. To obtain the resultant TI distribution Eq. 1b is applied, and the turbulence standard deviation is converted to TI by the relation ~~$\mathrm{TI} = \sigma_u/U$~~ $I = \sigma_u/U$, the TI distribution is shown in Fig. 1 (g) as a histogram.

~~*Windfarm simulation*~~*Wind farm simulation*

The dataset is generated with the wind farm simulation tool `PyWake` (Pedersen et al., 2023). PyWake is an open-source steady-state engineering wake modeling framework that computes wake deficits and turbine interactions. The framework supports spatially varying inflow conditions, multiple turbine types, and yaw misalignment via wake deflection models. However, to establish a baseline dataset, several simplifications were adopted: (i) homogeneous inflow with uniform free-stream velocity and turbulence intensity; (ii) a single turbine type (DTU-10-MW) with a fixed C_T-curve; and (iii) all turbines aligned with the inflow direction. These choices reduce the input parameter space but may limit applicability to scenarios involving heterogeneous inflows, mixed turbine fleets, or active wake control.

The wakes are modeled with the updated self-similar Gaussian single wake deficit model `NiayifarGaussianDeficit` by ~~Bastankhah and Porté-Agel (2014); Niayifar and Porté-Agel (2016)~~Niayifar and Porté-Agel (2016), which is a further development of the Gaussian wake model by Bastankhah and Porté-Agel (2014). This model was chosen ~~due to its relative simplicity and its inflow TI dependent wake expansion~~for its TI-dependent wake expansion, which ensures coupling between TI and velocity deficits. The model uses an adaptive wake growth rate (k^*) that is linearly ~~fitted~~ fit to the wind turbine inflow ~~Turbulence Intensity (TI$_{\mathrm{wt}}$~~TI (I_{wt}) immediately upstream of ~~a given turbine. The authors fitted the model to a Large Eddy Simulation (LES), this work uses their default parameters a_1 and a_2.~~

$$\frac{\Delta u}{U} = \left(1-\sqrt{1-\frac{C_{\mathrm{T}}}{8\left(\frac{k^*x}{D+\varepsilon_d}\right)^2}}\right)\exp\left(-\frac{1}{2\left(\frac{k^*x}{D+\varepsilon_d}\right)^2}\left(\frac{y}{D}\right)^2\right)$$

$$\varepsilon_d = 0.2\sqrt{\beta_d}, \quad \beta_d = \frac{1}{2}\frac{1+\sqrt{1-C_{\mathrm{T}}}}{\sqrt{1-C_{\mathrm{T}}}}, \quad C_{\mathrm{T}} < 0.9$$

$$k^* = a_1 \mathrm{TI}_{\mathrm{wt}} + a_2, \quad a_1 = 0.3837, \quad a_2 = 0.003678$$

~~where x is the streamwise direction, Δu is the velocity deficit in the x-direction, C_{T}, is the coefficient of thrust, and ε_d is a shape parameter offset dependent on C_{T} and defined as in .~~

~~To model~~ each turbine. Therefore, it requires an added TI (~~ΔTI~~ I_a) model; here, the `CrespoHernandez` model by Crespo and Hernández (1996) ~~is chosenfor its simplicity. The model depends on the induced velocity factor and the distance behind the turbine, as shown in~~ was chosen.

$$\Delta \mathrm{TI} = 0.73\, a_m^{0.8325}\, \mathrm{TI}^{0.0325} \left(\frac{x}{D}\right)^{-0.32}$$

$$a_m = 0.083\, C_{\mathrm{T}}^3 + 0.0586\, C_{\mathrm{T}}^2 + 0.2460\, C_{\mathrm{T}}$$

~~Where a_m is the induced velocity factor estimated with an empirical polynomial fit of C_{T} to address cases with $a_m \geq 0.5$ as described by Madsen et al. (2020).~~

~~To~~Additionally, to account for the effects of turbine induction, the updated self-similarity blockage model `SelfSimilarityDefici` by Troldborg and Meyer Forsting (2017); Forsting et al. (2023) is used. ~~The blockage model is used to calculate the blockage~~

~~deficit produced by individual wind turbines (Δu_b). It is based on the observation that inductions are radially self-similar for upstream distances greater than one rotor radius (R). It consists of an axial and~~ Finally, to superpose wakes and induction together, a ~~radial-shaped function. The newer variation of the model has an updated linear induction zone half radius ($r_{1/2}$), which corrects the behavior of the turbine inductions in wind farm contexts.~~

$$\frac{\Delta u_b}{U} = a_0(x, C_\mathrm{T})\nu(x)\mathrm{sech}^\alpha\left(\beta_b \frac{r}{r_{1/2}(x)}\right),$$

$$\nu(x) = \left(1 + \frac{x/R}{\sqrt{1+(x/R)^2}}\right),$$

$$a_0(x, C_\mathrm{T}) = \frac{1}{2}\left(1 - \sqrt{1 - \gamma(x, C_\mathrm{T})\cdot C_\mathrm{T}}\right),$$

$$\frac{r_{1/2}(x)}{R} = \lambda\cdot(x/R) + \eta,$$

$$\alpha = 8/9 \quad \beta_b = \sqrt{2} \quad \lambda = -0.672, \quad \eta = 0.4897$$

~~where ν is the centre-line induction and a_0 is the axial induction factor. The `SelfSimilarityDeficit2020` model introduced a $\gamma(x, C_\mathrm{T})$ function that gradually changes from a far-field expression to a near-field expression. Here, this is formulated as a function $\delta(x)$. The near- and far-field γ-functions are parameterized with $c_\mathrm{nf}^{(i)}$ for the near-field γ and $c_\mathrm{ff}^{(i)}$ for the far-field γ.~~

$$\gamma(x, C_\mathrm{T}) = \Bigg\{\delta(x)\cdot\left(c_\mathrm{nf}^{(1)}\cdot\sin\left(\frac{C_\mathrm{T} + c_\mathrm{nf}^{(2)}}{c_\mathrm{nf}^{(3)}}\right) + c_\mathrm{nf}^{(4)}\right)$$

$$\quad + (1-\delta(x))\cdot\left(C_\mathrm{T}^3 c_\mathrm{ff}^{(1)} + C_\mathrm{T}^2 c_\mathrm{ff}^{(2)} + C_\mathrm{T} c_\mathrm{ff}^{(3)} + c_\mathrm{ff}^{(4)}\right)\Bigg\}$$

$$c_\mathrm{nf}^{(1)} = -1.381, \qquad c_\mathrm{nf}^{(2)} = 2.627,$$

$$c_\mathrm{nf}^{(3)} = -1.524, \qquad c_\mathrm{nf}^{(4)} = 1.336,$$

$$c_\mathrm{ff}^{(1)} = -0.06489, \qquad c_\mathrm{ff}^{(2)} = -0.4911,$$

$$c_\mathrm{ff}^{(3)} = -0.1577, \qquad c_\mathrm{ff}^{(4)} = 1.116,$$

$$\delta(x) = \begin{cases} 1 & \text{for } x/R < -6 \\ \dfrac{|\nu(x) - \nu|}{\nu(-6) - \nu(-1)} & \text{for } -6 \le x/R \le -1 \\ 0 & \text{for } -1 < x/R \end{cases}$$

~~A~~ linear sum is used~~for the wake summations. The velocity~~. The effective velocity (u) at turbine number i is computed as

$$u_i = U - \sum_{j=1}^{N_\mathrm{up}} \Delta u_j - \sum_{k=1}^{N_\mathrm{down}} \Delta u_{b,k} \tag{2}$$

where Δu is the velocity deficit caused by wake effects, Δu_b is wake deficit caused by blockage, j sums across all turbines upstream of turbine i and k sums across all turbines downstream of turbine i. For the interested reader, the velocity deficit model, TI model, and blockage model are presented in detail in Appendix B.

~~Finally, to~~ To obtain the state of the wind farm, the `All2AllIterative` wind farm model is used, in which a local reference wind speed u_{ref} is calculated per wind turbine. The inter-turbine effects are iteratively evaluated using fixed-point iteration until a convergence ~~criteria~~ criterion is met. Using ~~`All2AllIterative`~~ All2AllIterative is necessary when a blockage model is included~~because~~, as the turbines interact in both ~~the~~ upstream and downstream directions. After the dataset for this work was generated, a computationally lighter alternative wind farm model, the `PropagateUpDownIterative` model, was implemented in PyWake. This model can significantly accelerate dataset generation, and the authors encourage its adoption in future studies.

~~Once a C_T is determined for all turbines, a flow map can be generated. The GNO is grid invariant, allowing for the strategic choice of a data coordinate system to capture interesting~~ However, the use of All2AllIterative also benefits the GNO application: Because turbine interactions are resolved iteratively in both directions, the resulting input-output mapping constitutes a non-linear operator, providing a more challenging and representative test case for the GNO than a simple `PropagateDownwind` scheme. Although PyWake employs linear wake superposition (Eq. 2), the coupled iterative solution introduces non-linearity that the GNO must learn to approximate. More broadly, the dataset was designed to capture diverse physical effects, including blockage, turbulence-dependent wake expansion, and bidirectional interactions, rather than to maximize fidelity to any single high-accuracy model. Such diversity ensures that the GNO learns a sufficiently complex operator, demonstrating its capacity to generalize across varied flow physics. Future work should investigate the impact of both different superposition methodologies and higher-fidelity training data on GNO performance.

One of the strengths of the GNO is its grid-invariance: once the turbine operating states are determined, flow predictions can be queried at arbitrary locations. This allows the simulated area to be strategically chosen to capture wake dynamics while avoiding redundant data from ~~flow away from turbine wakes~~unwaked regions. Therefore, a bounding box is constructed to fit each farm. For each farm layout, the wind turbines with the minimum and maximum downstream x and cross-stream y coordinates are found and padded as follows:

– **Downstream:** (x-direction) an extension of $100D$ behind the most downstream turbine,

– **Upstream:** (x-direction) an extension of $10D$ in front of the most upstream turbine,

– **Cross-stream:** (y-direction) an extension of $5D$ on each side.

Figure 3 illustrates the bounding box of a wind farm~~, and a~~. A farm downstream axis, ~~$\bar{x}$~~$\tilde{x}$, is introduced to ~~simplify comparisons between different wind farms~~measure the distance behind the most downstream turbine, making it easier to compare results across different wind farm layouts. The flow map is constructed to remain within the bounding box, at a single height, the turbine hub height ($z_{\mathrm{hub}} = 119$ m). The ~~flowmap~~ flow map uses an isotropic grid resolution of 3 points per D. For each layout and sampled inflow case, the computed effective turbine velocities and the velocities at the pre-determined grid locations are recorded. As an artifact of the PlayGen generation methods, the wind farm center and coordinate center do not always coincide. Because the GNO uses relative internal coordinates, this does not affect its performance; more details are given in Sect. 2.2.

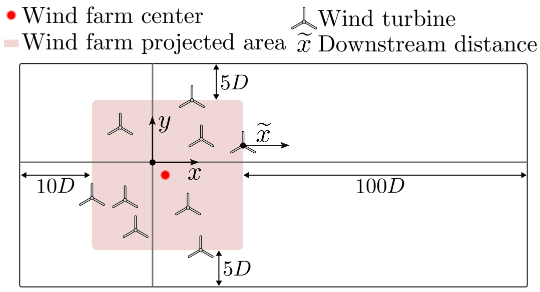

Figure 3. Adaptive computational grid for generating flow maps with `PyWake`.

2.2 Graphs

Graph theory was introduced by Euler (1741) to address the Königsberg bridge problem. Euler modeled land masses as nodes (v) and bridges as edges (e) in a graph. Although simple compared to modern graphs, this representation enabled Euler to mathematically prove the impossibility of solving the Königsberg bridge problem.

Graphs are powerful tools for representing data. They consist of nodes (or vertices), edges, and optionally global features. Nodes hold values representing information such as positions and states, while edges represent relationships or interactions between nodes and may also carry additional attributes, referred to as edge features. Global features, in turn, apply to the entire graph and are often used in graph classification tasks. In this work, global features are not included directly; instead, following the approach of Duthé et al. (2023), they are copied to each node as node features.

The notation in this work uses both set theory and vector notation, depending on which is more appropriate in a given context. In set notation, the nodes are represented by V and the edges by E, with the graph represented as $G = (V, E)$. In vector notation, the nodes are denoted by $\boldsymbol{v}$, with individual vector elements represented as v_i; similarly, the edges are denoted by $\boldsymbol{e}$, with individual elements represented as e_i. Set theory is primarily used to express the cardinality of these sets. That is, the number of elements they contain, written as $|V|$ and $|E|$. The use of cardinality highlights that the numbers of nodes and edges are not fixed, which is a strength of graphs but also makes the notation more complicated.

In our implementation, two types of nodes are considered: wind turbine nodes (V_{wt}) and probe nodes (V_{p}), where

$$V = V_{\mathrm{wt}} \cancel{\in} \cup V \cancel{,\ V}_{\mathrm{p}} \cancel{\in V} \qquad (3)$$

Wind turbine nodes coincide with the physical turbine positions in the farm, while probe nodes correspond to locations where flow predictions are desired, i.e., points in space at which the flow is evaluated. Similarly, the edges are separated into two inter-turbine edges (E_{wt}) and those connecting wind turbines and nodes (E_{p}), both a subset of E.

$$E = E_{\mathrm{wt}} \cup E_{\mathrm{p}} \tag{4}$$

The edges store the relative node positions (x_{ij} and y_{ij}), and Euclidean distance (d_{ij}) between connected nodes:

$$(x_{ij}, y_{ij}, d_{ij}) = \left(x_j - x_i,\, y_j - y_i,\, \sqrt{x_{ij}^2 + y_{ij}^2}\right) \tag{5}$$

The inter-turbine edges are bidirectional, enabling message passing in both directions. This corresponds to solving for turbine thrust coefficients in traditional wake modeling, and the resulting latent space can be thought of as a high-dimensional representation of the turbine thrust states.

By contrast, edges connecting to probe nodes are unidirectional, with each probe node connected to and receiving from all wind turbine nodes. This corresponds to using a wake simulation result to predict a resulting flow map when the turbine thrust states are already set.

Probe node connectivity is straightforward because each probe connects to all wind turbines. Inter-turbine connectivity is more complex because it ~~uses Delaunay triangulation~~ requires an algorithmic approach to determine which turbines should be connected.

Several algorithms exist for constructing ~~turbine graph connectivity~~ the connectivity of the turbine nodes. Duthé et al. (2023) investigated four graph connectivity schemes: Delaunay triangulation (Delaunay, 1934; O'Rourke, 1988; Fey and Lenssen, 2019), K-Nearest Neighbors (KNN), a radius-based method, and a fully connected scheme. They concluded that Delaunay triangulation provides the best balance between accuracy and computational performance. Accordingly, we also adopt Delaunay triangulation in this work to derive inter-turbine connections.

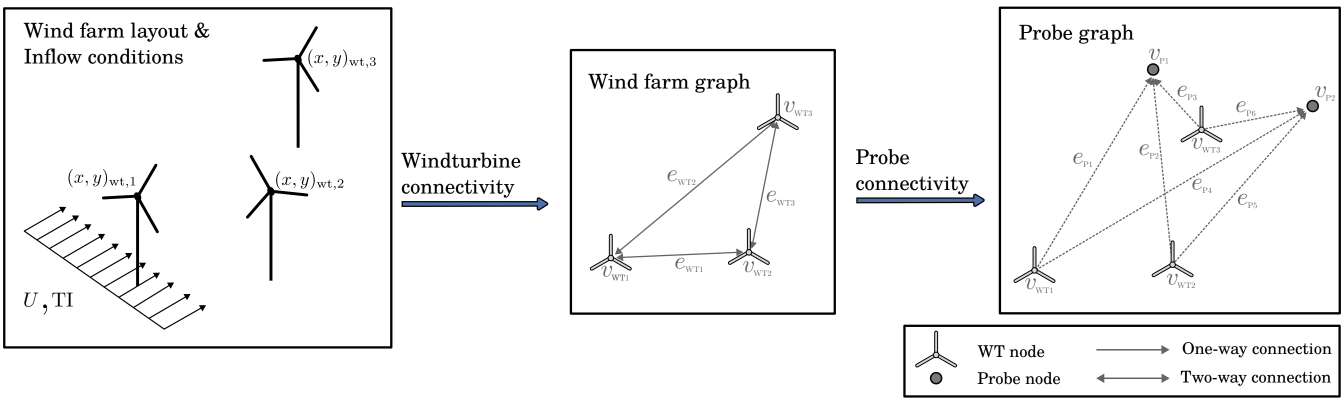

Figure 4. An example of constructing graph connectivity with three wind turbines and two probes, given a layout and inflow.

An example of the graph construction process is illustrated in Fig. 4, where the process of establishing the connectivity of the wind farm graph and the probe graph is separated into two stages. In our formulation, the edges include three features ($f_e = 3$) ~~: the position of the sending node in Cartesian coordinates (the streamwise x and the cross-stream y directions) and~~

~~the Euclidean distance between connected nodes~~as described in Eq. 5. Initially, wind turbine nodes store two features ($f_v = 2$)~~,~~ ~~namely the~~: The global wind speed U and ~~ambient TI~~the ambient I_0. No unique information is stored directly on the nodes; instead, global features are initially seeded and replicated across the nodes as part of the processing. Since the desired output is the wind speed at probe nodes, this constitutes a node task.

2.3 Graph Neural Operator (GNO)

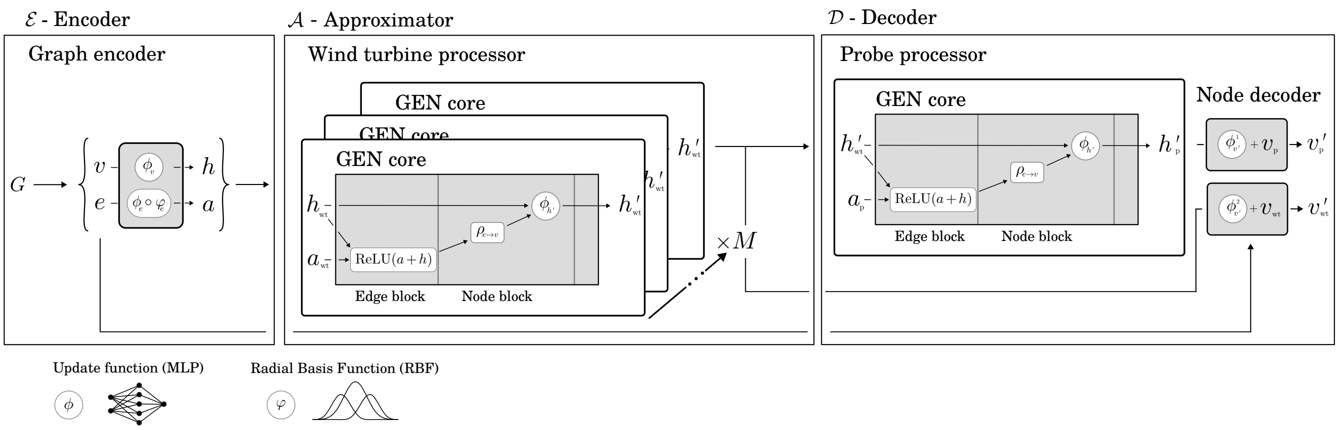

Figure 5. GNO network overview of its three main components: Encoder, Approximator and Decoder.

~~A~~

The Graph Neural Operator (GNO) is a special case of a ~~Graph Neural Network (GNN). The GNO was first proposed by Li et al. (2020b) as a general method to solve Partial Differential Equations (PDEs). Later Sun et al. (2022) proposed~~ GNN designed for learning mappings between function spaces. Li et al. (2020b) first introduced GNOs for this purpose. Building on this foundation, Sun et al. (2022) developed the Deep Graph Operator Network (DeepGraphONet), ~~combining~~ which integrates the Deep Operator Network (DeepONet) ~~by Lu et al. (2021) and Message Passing~~ architecture of Lu et al. (2021) with message-passing GNNs (Scarselli et al., 2009; Gilmer et al., 2017). ~~In this work , the DeepGraphONet is combined with the considerations of Seidman et al. (2022) on~~ The present work adopts the DeepGraphONet framework and extends it using the Nonlinear Manifold Decoders (NOMAD) ~~for operator learning.~~ formulation proposed by Seidman et al. (2022). NOMAD generalizes neural operators to fit into the encoder-processor-decoder abstraction. ~~They~~

Seidman et al. (2022) propose an operator $\mathcal{G}$ can be approximated as $\mathcal{F}$ using an Encoder $\mathcal{E}$, an Approximator $\mathcal{A}$, and a Decoder $\mathcal{D}$. The approximation $\mathcal{F}$ can be written as a composition of the model components using the composition operator $\circ$, where the rightmost function is applied first.

$$\mathcal{G} \approx \mathcal{F} = \mathcal{D} \circ \mathcal{A} \circ \mathcal{E} \tag{6}$$

~~The Encoder, Approximator, Decoder configuration is also a common abstraction~~

The encoder-approximator-decoder configuration is a widely adopted architectural pattern in GNNs, ~~see e.g.~~ providing a modular framework for transforming input graphs into predictions (Battaglia et al., 2018). In Fig. 5 an overview of the considered model is shown, while the ~~initial two~~ encoder and approximator stages conceptually are equivalent to most GNNs the final decoder stage stands out as the decoder includes a variation of a readout that uses a separate set of featured probe edges E_p. By including E_p, the relative position of the turbines to the probe location is included during the readout. Additionally, due to the separation of probe node processing and the simplicity of establishing probe connectivity, probes can be created on the fly without re-establishing wind turbine connections i.e. E_wt, nor re-processing the wind turbine nodes. ~~Which means~~ This means that predicting in a flow field ~~can be sped up compared to~~ incurs lower computational cost than a fully integrated prediction scenario.

As previously discussed, the structure of the GNO mirrors the modeling flow of engineering-based wind farm models. In such models, a connectivity matrix is established based on the spatial layout of the turbines and the chosen wake interaction scheme. For instance, in fully coupled formulations, all turbines influence one another, and the initial state is defined by the undisturbed inflow conditions. Analogously, the GNO constructs a graph representation of the wind farm, which is then encoded into an initial higher-order thrust state. The approximator within the GNO can be interpreted as analogous to the iterative update process used in conventional wake models to adjust individual turbine states. However, in the GNO framework, this computation is performed in a latent space rather than directly in the physical domain. Finally, the decoder evaluates the aggregated flow response of the wind farm, corresponding to the reconstruction of a flow field or flow map in traditional engineering approaches. These parallels are made explicit in Eq. 6, where the constitutive components of the GNO reflect the sequential structure of conventional wind farm modeling frameworks.

~~To train the GNO, a combination of different Python-based frameworks is used. The GNN components are constructed with the Jax (Bradbury et al., 2018) based libraries; Jraph (Godwin et al., 2020) for GNN abstractions, and Flax (Heek et al., 2024) for neural networks. Additionally, as the Jax ecosystem does not yet have a dedicated data pipeline, graph construction and subsequent data loading are handled using PyTorch Geometric (PyG) (Fey and Lenssen, 2019). For additional details on the data pipeline, see .~~

Encoding

The graph encoder layer $\mathcal{E}$ consists of parallel encoders for nodes and edges. The nodes are encoded with a ~~Multilayer Perceptron (MLP)~~MLP, and the edges are encoded in two steps: first, they are pre-processed with Radial Basis Functions (RBFs), and then encoded to the target latent space with an MLP. It is well known that there are different flow regimes at various downstream distances from a turbine. RBFs are used to encourage the network to view different downstream distances as distinct regimes. In initial experimentation, it was found to improve training. The input edge and node data are projected into latent spaces of the same dimensionality, denoted as Q, ensuring dimensional consistency as the chosen approximator core, GEneralized Aggregation Network (GEN), requires equal latent-space dimensions for nodes and edges. This is because the node- and edge features are added together in the latent-space.

The structure of an MLP is described in three stages: the input layer in Eq. 7a, the hidden layers in Eq. 7b, and the output layer in Eq. 7c.

$$\boldsymbol{\xi}^{(0)} = \psi\left(\boldsymbol{W}^{(0)\top}\boldsymbol{x} + \boldsymbol{b}^{(0)}\right) \tag{7a}$$

$$\boldsymbol{\xi}^{(l)} = \psi\left(\boldsymbol{W}^{(l)\top}\boldsymbol{\xi}^{(l-1)} + \boldsymbol{b}^{(l)}\right), \quad l = 1,2,\ldots,L-1 \tag{7b}$$

$$\phi(\boldsymbol{x}) = \boldsymbol{W}^{(L)\top}\boldsymbol{\xi}^{(L-1)} + \boldsymbol{b}^{(L)} \tag{7c}$$

~~Here,~~ where $\phi(\boldsymbol{x})$ represents the overall MLP mapping for input vector $\boldsymbol{x}$, $\boldsymbol{\xi}^{(l)}$ denotes the hidden units at layer l, $\boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$ are the weights and biases of the network, respectively, and ψ is the activation function. In this work, the activation function is the Rectified Linear Unit (ReLU). The MLPs considered in this work operate in real space and are used to map between different dimensions in both the latent and observational spaces, i.e., ~~$\phi : \mathbb{R}^{n\times c} \rightarrow \mathbb{R}^{n\times q} \quad \forall \quad n,c,q \in \mathbb{N}$~~ $\phi : \mathbb{R}^{N\times C} \rightarrow \mathbb{R}^{N\times Q} \quad \forall \quad N,C,$

RBFs are used to map the initial edge features, here distances between wind turbines i and j (d_{ij}), into a higher-dimensional space. We employ $K = 9$ Gaussian basis functions to transform distances such that $\varphi : \mathbb{R}^1 \rightarrow \mathbb{R}^K$. The formulation of the RBF is shown in Eq. 8.

$$\varphi(d_{ij}) = \exp\left(-\beta_{\text{RBF}}^{(k)}\left(d_{ij} - \mu_{\text{RBF}}^{(k)}\right)^2\right)\cdot\delta_c\left(d_{ij}\right), \qquad k = 1,\ldots,K, \tag{8a}$$

$$\delta_c(d_{ij}) = \begin{cases} 0.5\cdot\left[\cos\left(\dfrac{\pi d_{ij}}{d_c}\right)+1\right], & \text{for } d_{ij} \leq d_c, \\ 0, & \text{for } d_{ij} > d_c, \end{cases} \tag{8b}$$

where $\beta_{\text{RBF}}^{(k)}$ and $\mu_{\text{RBF}}^{(k)}$ are trainable parameters defining the k-th basis function, and $\delta_c(d_{ij})$ is a cosine cut-off function with cut-off distance d_c, ensuring a gradual and consistent decay. The initial $\mu_{\text{RBF}}^{(k)}$ are linearly spaced between -1 and 1, while $\beta_{\text{RBF}}^{(k)}$ all are initialized based on the maximum range and amount of basis functions $\beta_{\text{RBF}}^{(k)} = \frac{K}{\max(d_{ij})-\min(d_{ij})}$. The ~~initial RBF functions are illustrated in , where symmetry is emphasized by matching colors but using different line styles. Initial RBF kernels for normalized distance and initial $\beta_{\text{RBF}}^{(k)} = 4.5$.~~ addition of RBF functions to encode the distances is inspired by the work of Jørgensen and Bhowmik (2022) who used it in their GNN for electron density estimation.

Combined, the encoding steps are summarized in Eq. 9,

$$\boldsymbol{h} = \phi_v(\boldsymbol{v}), \tag{9a}$$

$$\boldsymbol{a} = \left(\phi_e \circ \varphi_e\right)(\boldsymbol{e}), \tag{9b}$$

where $\boldsymbol{h}$ and $\boldsymbol{a}$ denote the latent-space representations of nodes and edges, respectively. ~~$\varphi_e : \mathbb{R}^{|E|\times f_e} \rightarrow \mathbb{R}^{|V|\times f_e K}$~~ $\varphi_e : \mathbb{R}^{|E|\times f_e} \rightarrow \mathbb{R}^{|V}$ is the RBF function mapping each edge feature from the initial observational space to a K dimensional RBF space, and $\phi_v : \mathbb{R}^{|V|\times f_v} \rightarrow \mathbb{R}^{|V|\times Q}$ and $\phi_e : \mathbb{R}^{|E|\times f_e K} \rightarrow \mathbb{R}^{|E|\times Q}$ are the node and edge encoding MLPs. For convenience, the key dimensions are restated here: the cardinalities $|V|$ and $|E|$ vary with the wind farm configuration; $f_v = 2$ and $f_e = 3$ are the initial node and edge feature dimensions; $K = 9$ is the number of RBF kernels; and Q is the latent-space dimension.

Approximator: GEneralized aggregation Network (GEN)

The central component of the GNO is the approximator $\mathcal{A}$. It applies the GEN message passing algorithm by Li et al. (2020a) with three message passing steps ($M = 3$) sequentially applied on the encoded latent space wind turbine nodes ($\boldsymbol{h}_{\mathrm{wt}}$) using the latent space inter-turbine edges ($\boldsymbol{a}_{\mathrm{wt}}$) to obtain the processed wind turbine nodes ($\boldsymbol{h'}_{\mathrm{wt}}$).

~~In our work, the GEN message passing core proposed by Li et al. (2020a) is used. The~~ The GEN methodology consists in the construction of the messages ($\boldsymbol{m}_{ij}$), the application of the edge-to-node aggregation function ($\rho_{e\to v}$), and the node update MLP (ϕ_v),

$$\boldsymbol{m}_{ij}^{(m)} = \mathrm{ReLU}\left(\boldsymbol{h}_j^{(m)} + \boldsymbol{a}_{\mathrm{ji}}^{(m)}\right) + \varepsilon, \qquad j \in \mathcal{N}(i) \tag{10a}$$

$$\boldsymbol{h}_i^{(m+1)} = \phi_{\cancel{v}h'}{}^{(m)}\left(\boldsymbol{h}_i^{(m)} + \rho_{e\to v}\left(\boldsymbol{m}_{ij}^{(m)}\right)\right), \qquad m = 1,2\ldots M \tag{10b}$$

where m is the current message passing step, ~~$i \in I$~~ $i \in \mathcal{I}$ is the index of a receiving node, with ~~I~~ $\mathcal{I}$ being an index set relating to all receiving nodes, correspondingly j indicates a sending node, $\mathcal{N}(i)$ is a set of the nodes neighboring the node with index i, and $\varepsilon = 1 \times 10^{-6}$ is a small positive value added for numerical stability. In Fig. 5, the GEN core is visualized schematically. In this work, Softmax aggregation is used as ~~$\rho_{e\to v}$~~ $\rho_{e\to v}$, it uses Softmax to scale the latent space of each feature dimension ~~set ($\hat{\boldsymbol{x}}_j \in \mathbb{R}^{|\mathcal{N}(i)\times 1|}$).~~ independently. The incoming messages to node i from its $|\mathcal{N}(i)|$ neighbors form a matrix of shape $|\mathcal{N}(i)| \times Q$. For each feature dimension, let $\widehat{\boldsymbol{x}}_q \in \mathbb{R}^{|\mathcal{N}(i)|}$ denote the vector containing the q-th feature value across all neighbors.

$$\rho_{\cancel{e\to v}\,e\to v} = \sum\nolimits_{\cancel{\hat{\boldsymbol{x}}_j \in \mathcal{X}\frac{\exp(\hat{\boldsymbol{x}}_j)}{\sum_{\hat{\boldsymbol{x}}_r\in\mathcal{X}}\exp(\hat{\boldsymbol{x}}_r)}}\,\widehat{\boldsymbol{x}}_q \in \mathcal{X}} \frac{\exp(\widehat{\boldsymbol{x}}_q)}{\sum_{\widehat{\boldsymbol{x}}_r\in\mathcal{X}}\exp(\widehat{\boldsymbol{x}}_r)} \cdot \cancel{{}_j}\widehat{\boldsymbol{x}}_q \tag{11}$$

where ~~$\mathcal{X}$~~ $\mathcal{X} = \{\widehat{\boldsymbol{x}}_1, \ldots, \widehat{\boldsymbol{x}}_Q\}$ is a collection of neighborhood features to be aggregated independently.

Decoder

In the first part of the Decoder stage ($\mathcal{D}$), an additional message passing step is performed using the processed wind turbine nodes $\boldsymbol{h'}_{\mathrm{wt}}$ with the latent-space probe edges $\boldsymbol{a}_{\mathrm{p}}$ to obtain processed probe nodes $\boldsymbol{h'}_{\mathrm{p}}$ consisting of aggregated information from all the wind turbine nodes. In the final part of the decoder stage, two separate MLPs map the latent variables back to the observational space. A residual-network (ResNet) formulation is adopted, in which the free-stream velocity U is added to the decoded node values as the final step during inference. Consequently, this residual connection is also taken into account during the evaluation of the loss function during training.

$$\boldsymbol{v'}_{\mathrm{p}} = \phi_{v'}^{(1)}(\boldsymbol{h'}_{\mathrm{p}}) + \boldsymbol{v}_{\mathrm{p}}, \tag{12a}$$

$$\boldsymbol{v'}_{\mathrm{wt}} = \phi_{v'}^{(2)}(\boldsymbol{h'}_{\mathrm{wt}}) + \boldsymbol{v}_{\mathrm{wt}}, \tag{12b}$$

Where $\phi_{v'}^{(1)}$ and $\phi_{v'}^{(2)}$ are the decoding MLPs, both mapping from the latent space to the output space, $\phi_{v'}^{(1)}, \phi_{v'}^{(2)} : \mathbb{R}^{|V_{\mathrm{p}}|\times Q}, \mathbb{R}^{|V_{\mathrm{wt}}|\times Q} \to \mathbb{R}^{|V_{\mathrm{p}}|\times 1}, \mathbb{R}^{|V_{\mathrm{wt}}|\times 1}$. Here, $\boldsymbol{v}_{\mathrm{p}}$, $\boldsymbol{h'}_{\mathrm{p}}$, and $\boldsymbol{v'}_{\mathrm{p}}$ denote the probe input nodes, hidden-state

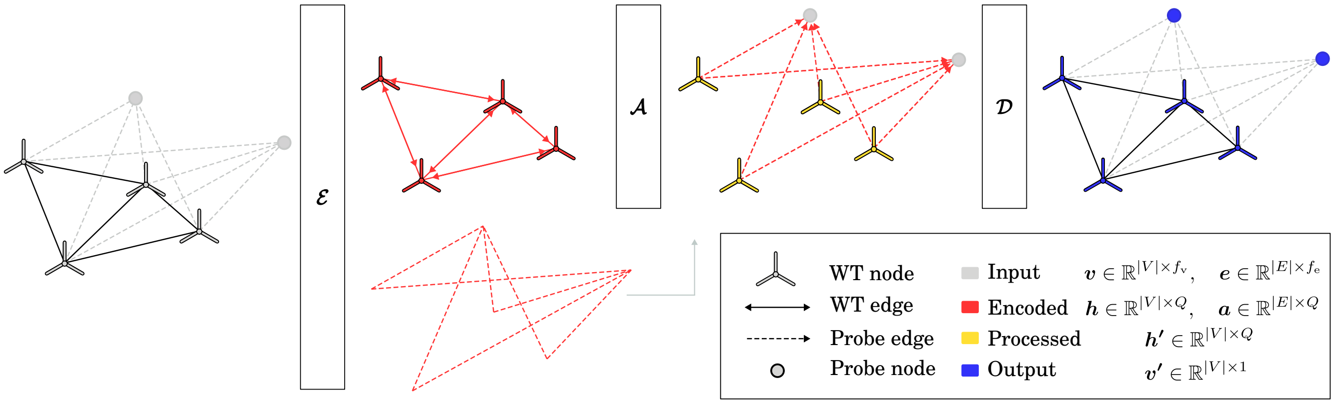

Figure 6. GNO Probe input, processing and output with model data flow from left to right.

nodes, and predictions, respectively. Likewise, $\boldsymbol{v}_{\mathrm{wt}}$, $\boldsymbol{h'}_{\mathrm{wt}}$, and $\boldsymbol{v'}_{\mathrm{wt}}$ correspond to the wind turbine input nodes, hidden-state nodes, and predictions.

The data flow through the GNO is shown in Fig. 6, where the components defined in Eq. 6 and described in this subsection are represented as boxes. The partially processed graph states are distinguished by color, providing a visual overview of the GNO architecture and its internal data transformations.

Regularization & Normalization

With the layered nature of ~~a~~ the GNO model the total amount of layers grow fast, with deep models it is advisable to include regularization and normalization to avoid overfitting. In this work, both the node- and edge features are scaled, meaning wind speeds, TI, and the positional information. Additionally, layer normalization and dropout regularization are used inside the GNO. For feature scaling min-max normalization is used.

$$\vartheta' = \frac{\vartheta}{\max(\vartheta) - \min(\vartheta)} \tag{13}$$

where ϑ is an unscaled feature and ϑ' is the respective scaled feature. The features are only scaled by the feature range as it allows ResNet application inside the scaled feature space, which makes model implementation easier.

To counter overfitting, dropout regularization, as implemented by Srivastava et al. (2014), has been incorporated into our MLP formulation. Dropout works by randomly omitting the output of some neurons during training, making the network more robust and reducing the risk of over-reliance on single neurons. ~~Here, the Dropout method is used, along with the addition of inverse probability scaling, which means the entire Dropout layer can be ignored during inference. During training Dropout is~~

~~applied both after and ,~~

$$\cancel{z_j^{(l)} \sim \mathrm{Bernoulli}(P_{\mathrm{D}}), \quad j = 1, 2, \ldots, N_a}$$
$$\cancel{\tilde{\boldsymbol{\xi}}^{(l)} = \boldsymbol{z}^{(l)} \odot \frac{1}{1 - P_{\mathrm{D}}} \cdot \boldsymbol{\xi}^{(l)}, \quad l = 1, 2, \ldots, L}$$

~~here $\boldsymbol{z}^{(l)}$ is a Dropout mask consisting of zeros and ones, created with $\mathrm{Bernoulli}(P_{\mathrm{D}})$ the Bernoulli distribution given the Dropout probability, which here is chosen to be~~ A dropout probability of $P_{\mathrm{D}} = 0.1$ ~~, N_a is the number of activations in layer l, the operator $\odot$ denotes the Hadamard element-wise product, and $\tilde{\boldsymbol{\xi}}^{(l)}$ is the resultant hidden state with Dropout applied~~has been used in this study; further implementation details can be found in Appendix C.

A common technique to improve training stability and speed up training is batch normalization. However, as the trained GNO can handle graphs of any size, there is no fixed batch size, which makes batch statistics unreliable. Additionally, during training, graphs are batched together, and statistics across graphs are not desirable. For additional information about graph batching with GNOs, see Appendix D. Instead, Layer Normalization, as proposed by Ba et al. (2016), can be used. While often considered less effective than batch normalization, layer normalization is fully compatible with GNNs. Ba et al. (2016) originally introduced it for MLPs and recurrent neural networks, applying it between hidden layers. In this work, since GNNs can be interpreted as compositions of MLP layers, layer normalization is applied only to the final layer of each MLP within the encoder $\mathcal{E}$ and approximator $\mathcal{A}$ stages, as well as during the message-passing step of the decoder $\mathcal{D}$ stage. It is not applied to the final node-decoder MLPs, where full expressive power at the output is desired. ~~The formulation of layer normalization used is shown in .~~

$$\cancel{\mu_{\mathrm{LN}}^{(L)} = \frac{1}{H} \sum_{i=1}^{H} \xi_i^{(L)}, \qquad \sigma_{\mathrm{LN}}^{(L)} = \sqrt{\frac{1}{H} \sum_{i=1}^{H} \left(\xi_i^{(L)} - \mu_{\mathrm{LN}}^{(L)} \right)^2},}$$
$$\cancel{\tilde{\boldsymbol{\xi}}^{(L)} = \boldsymbol{s}_{\mathrm{LN}} \odot \frac{\boldsymbol{\xi}^{(L)} - \mu_{\mathrm{LN}}^{(L)}}{\sigma_{\mathrm{LN}}^{(L)} + \varepsilon} + \boldsymbol{b}_{\mathrm{LN}},}$$

~~Where $\mu_{\mathrm{LN}}^{(L)}$ and $\sigma_{\mathrm{LN}}^{(L)}$ denote the layer mean and standard deviation, $\xi_i^{(L)}$ is the ith activation at the last MLP layer L, and $\tilde{\boldsymbol{\xi}}^{(L)}$ represents the layer-normalized hidden states, while $\boldsymbol{s}_{\mathrm{LN}}$ and $\boldsymbol{b}_{\mathrm{LN}}$ are trainable scale and bias parameters, and $\varepsilon = 1 \times 10^{-6}$ is added for numerical stability.~~ Implementation details can be found in Appendix C.

2.4 Training and evaluation

The GNO is intended as a surrogate model for a regression-type task. Since a true model exists to learn from, the most straightforward training method is offline supervised training. To train the GNO, the High-Performance Computing (HPC) cluster Sophia (Technical University of Denmark, 2019) at DTU has been used. Nodes with `Quadro RTX 4000` GPU acceleration were employed~~; these nodes have a limited runtime of 72 hours~~.

The optimization algorithm employed during training is `Adam` (Kingma and Ba, 2014). Adam is a variant of Stochastic Gradient Descent (SGD) that incorporates past gradients and a momentum term to compute parameter updates. Consequently, multiple hyperparameters control the relative weighting of these terms. In this work, the default momentum parameters are used. During training, the dataset is shuffled and passed through the training algorithm multiple times, one such pass is termed an Epoch. During an epoch when a graph is sampled, the probes are sub-sampled from the flow domain; in practice, a flattened list is sampled uniformly. The probe node sample size (n_p) can be considered a hyperparameter that has been subject of investigation during a hyperparameter search. In our implementation, a maximum of 3000 Epochs has been considered, although this limit has not been reached at any point. ~~Instead, the limiting factor has been the 72-hour walltime.~~

During training, the model is continuously evaluated using the validation dataset. If the validation performance surpasses previous results at any time, the saved model weights and biases are updated. The validation evaluation is performed every fifth Epoch to save computational resources. Unlike during training, the graphs are not shuffled, and the probe nodes remain the same across epochs.

Machine learning framework

To train the GNO, a combination of different Python-based frameworks is used. The GNN components are constructed with the Jax (Bradbury et al., 2018) based libraries; Jraph (Godwin et al., 2020) for GNN abstractions, and Flax (Heek et al., 2024) for neural networks. Additionally, as the Jax ecosystem does not yet have a dedicated data pipeline, graph construction and subsequent data loading are handled using PyTorch Geometric (PyG) (Fey and Lenssen, 2019). For additional details on the data pipeline, see Appendix D.

Performance metrics

Once a set of GNO weights are obtained through training, the resultant GNO model has to be tested. To accurately assess performance, various performance metrics are considered. Four statistical measures are considered: the Mean Absolute Error (MAE), the Mean Absolute ~~Percentage Error (MAPE~~Normalized Error (MANE), the Mean Square Error (MSE), and the Root Mean Square Error (RMSE). They differ in the first two using an l^1-norm and last two using an l^2-norm. MSE and MAE are used to compare models, while ~~MAPE~~ MANE and RMSE is chosen to evaluate the best model as they are more easily interpreted. While most metrics are standard, MANE is a custom variation of Mean Absolute Percentage Error (MAPE). Standard MAPE normalizes by the target value, which becomes problematic when targets approach zero as can occur near heavily waked turbines where wake superposition overestimates the deficit. Normalizing by the wake deficit instead would cause similar issues in unwaked regions. To avoid this, MANE normalizes by the free-stream velocity, providing a more robust metric that still enables comparison across different inflow conditions. The primary reason for including multiple metrics is to facilitate cross-comparison with other works in the field, as there is no consensus on which metrics to use. The metrics considered here are shown in Eq. 14,

$$\mathrm{MAE} = \frac{1}{N}\left\|\boldsymbol{u}-\widehat{\boldsymbol{u}}\right\|_1, \tag{14a}$$

$$\cancel{\mathrm{MAPE}}\mathrm{MANE} = \frac{1}{N}\left\|\cancel{\tfrac{\boldsymbol{u}-\hat{\boldsymbol{u}}}{U}}\frac{\boldsymbol{u}-\widehat{\boldsymbol{u}}}{\boldsymbol{U}}\right\|_1 \cdot 100\% \tag{14b}$$

$$\mathcal{L} = \mathrm{MSE} = \frac{1}{N}\left\|\boldsymbol{u}-\widehat{\boldsymbol{u}}\right\|_2^2, \quad \mathrm{RMSE} = \frac{1}{\sqrt{N}}\left\|\boldsymbol{u}-\widehat{\boldsymbol{u}}\right\|_2 \tag{14c}$$

where $\boldsymbol{u}$ ~~is~~ are the target velocities and ~~$\hat{\boldsymbol{u}}$ is~~ $\widehat{\boldsymbol{u}}$ are the predicted velocities, $\boldsymbol{U}$ ~~is~~ are the corresponding inflow velocities, and N is the number of observations. MSE is used as the loss function ($\mathcal{L}$) during training, as shown in Eq. 14c.

~~For an easy overview of how the errors vary, it is effective to estimate the Probability Density Function (PDF). To estimate the PDF, Kernel Density Estimation (KDE) is applied. The implementation of KDE uses a Gaussian kernel, and Scott's rule (Scott, 1992) to estimate the bandwidth H. H governs the strength of the smoothing effect.~~

$$\cancel{\hat{\varphi}(\zeta) = \frac{1}{NH\sqrt{2\pi}}\sum_{i=1}^{N}\exp\left(-\frac{1}{2}\left(\frac{\zeta-\zeta_i}{H}\right)^2\right)}$$

$$\cancel{H = \sigma_{\mathrm{kde}}N^{-1/5}}$$

~~Here ζ is a random input, σ_{kde} is the standard deviation of the sample data, and $\hat{\varphi}$ is the Gaussian kernel KDE function.~~

In some cases, it is practical to compare graphs by their relative size; a common approach to do so is to add up the size of their constituent parts. This can be done with different scaling factors and norms; however, the most straightforward and common approach is a linear sum accounting for feature sizes. The cardinality of the graph tuple $|G|$ is therefore defined as:

$$|G| \equiv (|V_{\mathrm{wt}}| + |V_{\mathrm{p}}|)\cdot f_v + (|E_{\mathrm{wt}}| + |E_{\mathrm{p}}|)\cdot(1+f_e) \tag{15}$$

Where the ~~addition of 1 to edge set scaling $(1+f_e)$ is added to account for the connectivity list in the practical implementation of a graph tuple~~term $(1+f_e)$ accounts for the fact that each edge stores both its f_e features and an index pair identifying the connected nodes in the adjacency list representation.

IEA Wind 740-10-MW reference wind farm

To evaluate the model ~~under realistic conditions~~on an established benchmark, the IEA Wind 740-10-MW reference wind farm is employed. This site features both a regular and an irregular ~~turbine~~ wind farm layout, corresponding respectively to a simple grid configuration and an optimized configuration designed to improve overall farm performance. Both layouts are tested under the same wind speed conditions as reported in the technical documentation by Kainz et al. (2024). The two layouts are shown in Fig. 7. As the GNO predicts wind speeds not power it is necessary to calculate power using the power curve of the DTU-10-MW illustrated in Fig. 2 a, as this is a static simulation this is a simple operation.

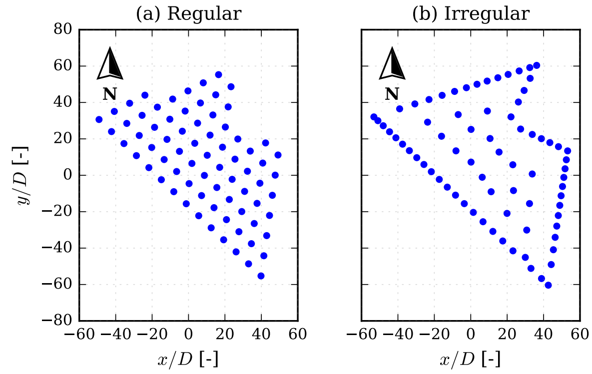

Figure 7. IEA Wind 740-10-MW reference wind farm (Kainz et al., 2024). (a) Regular layout pre-optimization and (b) Irregular optimized layout.

3 Results and Discussion

In this section the results are presented and discussed concurrently. Initially, a summary of our hyperparameter grid search study is presented, followed by the selection of the best model based on the validation data and its subsequent testing. The best performing model is investigated separately, and a series of predictions are made using random layouts of each considered layout type. Both the probe node predictions and the wind turbine node predictions are investigated. Thereafter, a performance analysis is conducted, examining how errors are distributed across different input variables. Finally, the computational cost of the model is investigated and compared to the cost of PyWake.

3.1 Grid search

Neural network models consist of trainable parameters that are updated during training. In addition to these trainable parameters, preset parameters known as *hyperparameters* also define the model. The hyperparameters can be divided into two groups: The ones defining the model architecture and those specifying the optimization configuration. In this work, five consecutive grid searches were performed, with the results from each used to guide the subsequent search.

For the model configuration, the encoder, wind turbine processor, and probe processor MLPs are collectively referred to as the internal MLPs. The tested hyperparameters are: the size of the latent dimension (Q), the number of layers in the internal MLPs (L_{int}), the number of neurons per layer in the internal MLPs (q_{int}), the number of hidden layers in the decoder MLP (L_{dec}), and the number of neurons per layer in the decoder (q_{dec}). ~~Table E1 lists the 11 considered model configurations, each assigned a letter ID.~~

~~Table E1 also includes the optimization configurations. The considered hyperparameters here are related to~~ The considered optimization hyperparameters are the Learning Rate (LR) schedule, which falls into two main categories: *constant* and *piecewise constant*. An additional category, *untriggered piecewise constant*, indicates that the LR remained at its initial value throughout optimization. This occurs because the maximum wall time has been reached, and the training is stopped. This occurs ~~with configurations that have~~ in configurations with large input sizes ~~,~~ because each training step takes longer ~~, resulting in fewer epochs being completed~~and processes more data, requiring fewer epochs to converge.

In the piecewise constant schedule, the current LR is divided by 10 when a trigger step is reached. The trigger steps themselves constitute another hyperparameter investigated in the grid search. Finally, two additional hyperparameters are considered: the number of probes per graph (n_p) and the number of ~~batched graphs~~ graphs per batch (n_G). The batch includes an additional padding graph consisting entirely of zeros; see Appendix D for further details on batching and padding graphs. ~~Model and optimizer configurations used in the grid search. † indicates that the learning rate schedule was not triggered during training.~~ ~~**Model configurations** ID Latent dim.~~ $\cancel{Q}$ $\cancel{L_\mathrm{int}}$ $\cancel{q_\mathrm{int}}$ $\cancel{L_\mathrm{dec}}$ $\cancel{q_\mathrm{dec}}$ ~~a 150 2 100 3 150 b 150 2 100 3 250 c 150 2 200 3 150 d 150 2 200 3 250 e 250 2 100 3 150 f 250 2 100 3 250 g 250 2 200 3 150 h 250 2 200 3 250 i 100 3 75 3 250 j 100 2 250 3 350 k 50 3 350 4 350~~ ~~**Optimizer configurations** ID LR type LR Triggers~~ $\cancel{n_\mathrm{p}}$ $\cancel{n_G}$ ~~1 Piecewise constant† 0.005 500, 1000200 5 2 Piecewise constant 0.005 75, 150200 5 3 Constant 0.001 - 200 5 4 Piecewise constant 0.001 75, 150200 5 5 Piecewise constant 0.005 75, 150300 4 6 Piecewise constant 0.001 75, 150300 4 7 Piecewise constant 0.010 75, 150200 5 8 Piecewise constant† 0.005 75, 150500 2 9 Piecewise constant† 0.010 75, 150500 2~~

~~The different combinations of~~ For the interested reader, details of the parameters considered during the grid search are reported in Appendix E. Table E1 lists the considered model and optimizer configurations; each model configuration is assigned a letter ID and each optimizer configuration is assigned a numeric ID. The different combinations were trained, and the results are ~~listed in Tab. E2. Besides~~ presented in Tab. E2, which includes the model and optimizer IDs, and a grid search number ~~is provided. Three metrics are provided,~~ in roman numerals, the training loss ($\mathrm{MSE}_\mathrm{trn}$), and two validation metrics ($\mathrm{MSE}_\mathrm{val}$ ~~)~~ ~~and (~~and $\mathrm{MAE}_\mathrm{val}$). ~~The entries in have been ranked by the~~ Entries are ranked by $\mathrm{MSE}_\mathrm{val}$~~metric, as it~~ , as this serves as the selection criterion for the best overall model.

The best five models from the hyper-parameter search are presented in Tab. 1; the best combination was Vj8, which featured the largest considered decoder, with the remaining parameters in the middle of the explored ranges. The best performing models in ~~Tab. E2~~ Tab. 1 show a high correlation between ~~$\mathrm{MSE}_\mathrm{val}$ and $\mathrm{MAE}_\mathrm{val}$, which increases the~~ $\mathrm{MSE}_\mathrm{val}$ and $\mathrm{MAE}_\mathrm{val}$. Since these metrics weigh errors differently, this increases confidence that the ~~right model is chosen. The training loss $\mathrm{MSE}_\mathrm{trn}$ on the other hand is not as strongly correlated; this can partially be ascribed to the training dataset nodes being re-sampled during every epoch, which means there is a stochastic component to the $\mathrm{MSE}_\mathrm{trn}$ metric. By contrast, the validation metrics are deterministic and therefore better for comparison.~~ model selection is robust and not overly sensitive to the choice of metric.

~~In~~ Based on the grid search, the five best-performing models by $\mathrm{MSE}_\mathrm{val}$ have been evaluated on the unseen and unscaled test set using the metrics in Eq. 14~~and the unseen test dataset.~~ , the results are reported in Tab. 2. Unlike the training and validation metrics used to compare models, the test metrics also include the more interpretable RMSE and MANE. Additionally, a naive baseline has been created, assuming no wake losses.

All five models perform similarly, indicating that, for the considered parameters, a minimum has been reached and that more substantial model changes are necessary to further improve performance. While the top model Vj8 from the grid search ~~in~~ maintains the best score in terms of MSE and RMSE, it is the third-best model that achieves the best MAE and

~~MAPE~~MANE. Interestingly, these two models share the same model configuration ~~j~~j, but differ in the optimizer configuration, with the variation arising from the input configurations of n_{p} and n_G. This could indicate that the input configuration is ~~of less importance compared to~~ less important than the model configuration. The number of message-passing steps (M) was fixed at 3 throughout the grid search rather than treated as a tunable hyperparameter. Initial testing on a simpler model showed low sensitivity to additional message-passing steps, suggesting diminishing returns beyond $M = 3$. While increasing M could theoretically improve performance in densely clustered farm configurations by propagating wake interactions across more neighbors, preliminary experiments did not support this hypothesis. However, the optimal value of M is suspected to be data-dependent; higher-fidelity datasets capturing more complex wake dynamics or turbulence interactions may benefit from additional message-passing steps to fully resolve inter-turbine dependencies. A systematic sensitivity study on M across diverse farm configurations and data fidelities should be considered for future work.

3.2 Model performance

In this section, the performance of the best model from Tab. E2 is investigated. The testing is conducted in steps initially, where a random selection of each layout type is used to illustrate the model capabilities in the far wake at different downstream distances ~~$\bar{x}$~~$\widetilde{x}$, see Fig. 3 for the definition of ~~$\bar{x}$~~$\widetilde{x}$. In the second step, a more data-centric approach is taken by illustrating the error statistics using the test set and analyzing the error with relation to different input types. Then, the model computational cost in terms of speed and memory is assessed, while varying the number of probe nodes to see the impact of graph size on computational cost.

Predictions

In ~~Fig. 8~~ Fig. 8 (a)-(d), velocity deficit (Δu) predictions made with the GNO are compared to the PyWake test data for a random layout of each layout type. The predictions are illustrated for three downstream distances ~~$\bar{x} = \{25, 50, 100\}$ D~~$\widetilde{x} = \{25, 50, 100\}$ D, at three freestream wind speeds $U = \{6, 12, 18\}$ ms^{-1} and ~~TI = 5%~~$I_0 = 5\%$. Unsurprisingly, the cluster farm with a number of wind turbines, ~~$n_{\mathrm{wt}} = 90$~~$n_{\mathrm{wt}} = 100$, has the most significant wake effect but also the most smeared wake deficit, especially visible for the $U = 6$ ms^{-1} scenario. The remaining layouts have fewer wind turbines, but also more structured layouts. They show clearer peaks and valleys, especially the single string layout. While the performance at $U = 6$ ms^{-1} is satisfactory for all layouts, it becomes less accurate at higher wind speeds. At $U = 12$ ms^{-1}, the model performs the worst, although it captures the shape of the velocity deficit; the scale could be improved. In ~~Fig. 8, the~~ Fig. 8, the maximum velocity deficit error ~~ranges between $\sim 2\%$~~relative to $\widehat{u}$ ranges from 0.23% to ~~$\sim 5\%$ relative to U~~6.13% across the different inflow velocities and layouts.

As a secondary output, the model can predict the velocity at individual turbines. To illustrate this capability, Fig. 9 shows the maximum error observed at each turbine for the same layouts and inflow velocities as those presented in Fig. 8, the inflow direction is from left to right along the downstream direction x. In Fig. 9, it can be seen that ~~heavily waked turbines~~ waked turbines tend to have the largest errors~~in~~. In Fig. 9 (c)~~ and (d) where the turbines are not oriented in a way that creates as deep and dense rows as~~, the most significant error occurs where the density is high and the wake effect is substantial, while in Fig. 9 (a) and (~~b) , the wake effect is~~d) it can be seen that the turbines happening to be downstream of multiple others

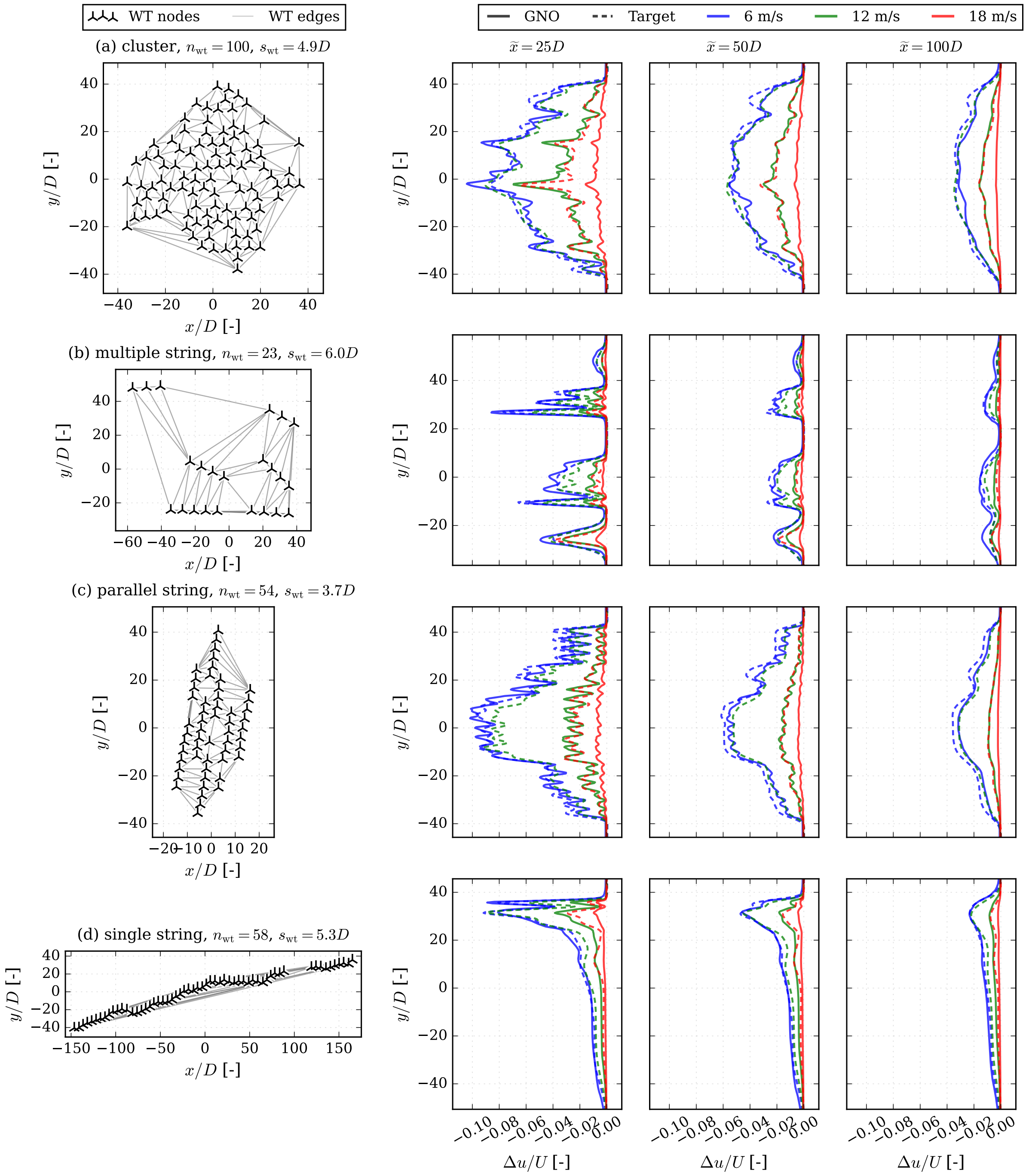

Figure 8. ~~(a-d)~~ Normalized velocity deficit predictions and targets at different wind farm downstream distances ~~$\bar{x} \in \{25D, 50D, 100D\}$~~ $\tilde{x} \in \{25D, 50D, 100D\}$ at $I_0 = 5\%$. (a) cluster, (b) single string, (c) multiple string and (d) parallel string.

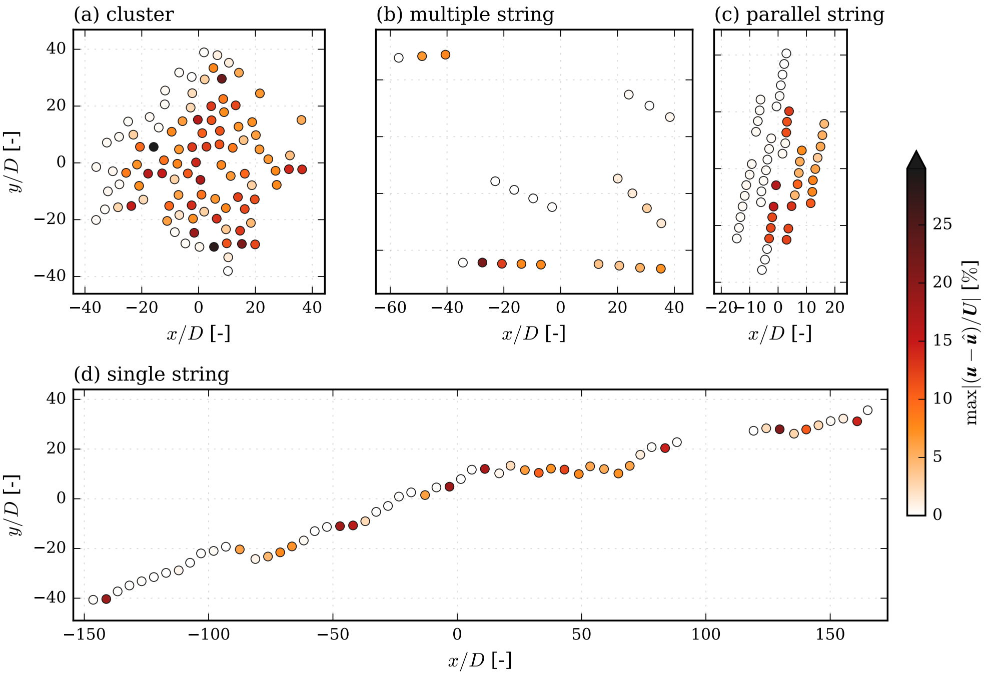

Figure 9. Maximum absolute errors at each wind turbine, for $\boldsymbol{U} = [6,\ 12,\ 18]^\top\ \mathrm{ms}^{-1}$ and $I_0 = 5\%$

produce high errors, although the highest error occurs in the middle of a row. Figure 9 (b) is of the type multiple string and happens to be a good example of the impact of string alignment with respect to the incoming wind having a significant effect on the errors. Here, the largest error occurs at the second turbine in a string aligned with the flow; this is most likely due to the induction effects playing a larger role. In general, where the wake effects are not as pronounced~~and~~, the errors are smaller. For the interested reader, the relative absolute errors are also reported for individual wind speeds in Appendix F.

IEA Wind 740-10-MW reference wind farm

The reference wind farm consists of two layouts a regular and an irregular as described in Sect. 2.4. For each layout, the total farm power output at each wind direction is computed and visualized using a polar grid. As the GNO does not inherently account for wind direction, this is achieved by translating and rotating the wind farm layout. In this way, each wind direction is represented as an equivalent new configuration. Repeating this process across all directions enables the calculation of wind-direction dependent farm power, without requiring the model to explicitly encode directional information. This approach is feasible because the GNO has been trained on a large and diverse dataset comprising numerous wind farm configurations. Consequently, it generalizes well across a wide range of geometric arrangements and inflow conditions. The wind-direction dependent farm power for a turbulence intensity of ~~TI = 5%~~ $I_0 = 5\%$ are presented in Fig. 10.

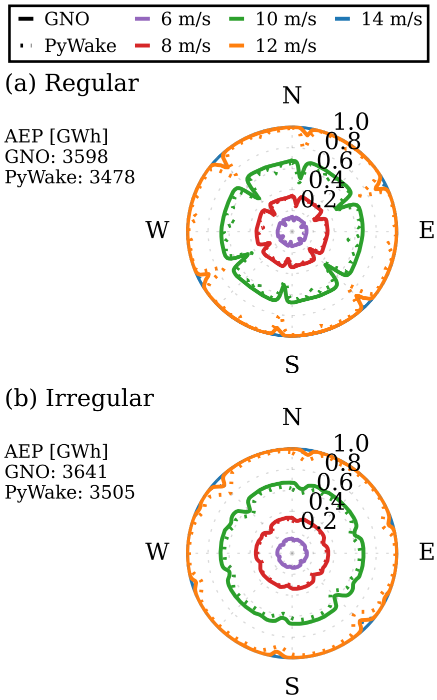

Figure 10. Normalized power production at different wind speeds and ~~TI = 5%~~$I_0 = 5\%$, illustrated as ~~windroses~~ power roses for the IEA Wind 740-10-MW reference wind farm (Kainz et al., 2024). (a) Regular layout pre-optimization and (b) Irregular optimized layout.

As shown in Fig. 10~~a~~ (a), the regular grid layout being non-optimized exhibits more pronounced internal wake effects and therefore produces lower overall power. In contrast, the irregular layout in Fig. 10~~b~~ (b) shows reduced wake interactions and higher power production across most wind directions. This result reflects the optimized nature of the irregular layout, which minimizes wake losses and enhances overall farm performance. The GNO reproduces the overall power trends well for both layouts, with slightly improved accuracy for the irregular configuration. This behaviour is consistent with previous observations that data-driven models tend to perform best under conditions similar to their training data, and where variability is lower. The procedurally generated training layouts do not include simple regular-grid configurations, which may explain the slightly lower accuracy on the regular IEA layout. Including such layouts in future datasets could improve generalization to such farms.

At a wind speed of $14\ \mathrm{ms}^{-1}$, both farms operate close to rated power, leading to small differences in predicted output. Some discrepancies remain at lower wind speeds and for specific wind directions where wake effects are more pronounced. The GNO successfully identifies the regions of strongest wake interaction, although the scale of the effects is not reproduced exactly. The model has been trained on data from within the farm and the far wake, but it does not perform well near the turbines. To demonstrate this model limitation, Fig. 11 shows predictions made with the GNO model, the PyWake targets, and their relative errors. Similar to the previous errors, we have seen that the largest errors are observed for $U = 12\ \mathrm{ms}^{-1}$ while errors are significantly smaller for both $U = 6\ \mathrm{ms}^{-1}$ and $U = 18\ \mathrm{ms}^{-1}$. The largest errors are consequently observed almost on top of the turbines, which is also where the largest and sometimes nonphysical deficits are found. The predictions made in

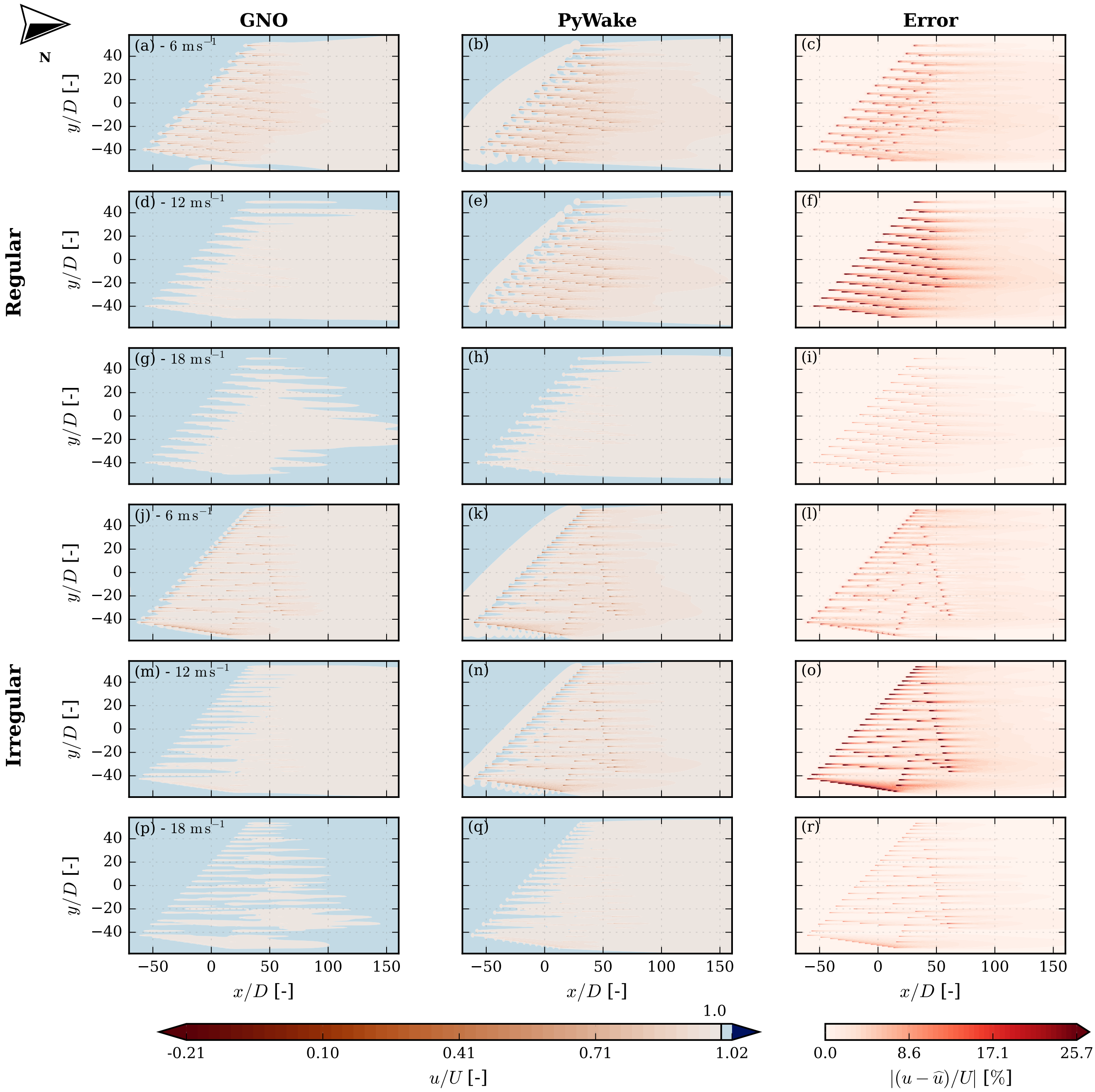

Figure 11. Contour plots depicting the flow around the IEA740 reference wind farm with columns depicting the GNO predicted output, the engineering model target output and the error between the predictions for $\boldsymbol{U} = [6,\ 12,\ 18]^\top$ ms^{-1} and $I_0 = 5\%$ rotated to the equivalency of incoming wind from west.

Fig. 11 shows promise, but demonstrate that the model will need further improvements before it is ready for inter-farm flow simulations.

Performance analysis

To evaluate the performance of the model under different scenarios, predictions are made for all combinations of layouts and inflows in the test dataset. For each layout, 10,000 ~~equally spaced probes are chosen~~probes are selected at equal spacings, and an RMSE is calculated for each flow case. The resultant metrics are displayed in Fig. 12 with separate visualizations for each layout type.

In Fig. 12 (a), ~~a KDE approximation~~ binned counts of the RMSE ~~PDFs~~ metrics is illustrated for each layout type. The ~~PDFs~~ distributions show that the single- and multiple-string layouts have comparable ~~PDFs, and~~ distributions as the two lines lie almost on top of each other, they are simultaneously the best-performing layouts. The worst-performing layout type is the cluster, followed by the parallel string. To further investigate why that is the case, in Fig. 12 (~~b) - (e)~~ b-e) bar plots of the errors are provided to investigate the effect of different aspects of the model inputs. Fig. 12 (~~b~~c) shows a bar plot of the mean RMSE for different inflow wind speeds U. The four layouts share similar distributions with regard to U, all of them exhibiting the largest mean RMSE at $U = 12\ \mathrm{ms}^{-1}$. The largest error occurring at $U = 12\ \mathrm{ms}^{-1}$ is most likely related to the wind turbine C_T-curve, as 12 ms^{-1} is just past the steepest part of the curve in region III, see ~~Fig. 2.~~ Fig. 2. This means that as turbines are affected by wakes inside the farm, the highest variability of C_T occurs at 12 ms^{-1}. Additionally, it is near the discontinuity of C_T when the rated wind speed is reached. To alleviate this, it is suggested that future implementations should rebalance the training dataset to proportionally include more flow cases right above the rated wind speed.

Fig. 12 (~~c~~d) shows a bar plot of the mean RMSE as a function of ~~TI~~ambient TI. As expected, there is an inverse correlation between RMSE and ~~TI. Lower TI~~ I_0. Lower I_0 indicates stronger wake effects, as the wake structure persists for longer, leading to a more complex farm flow and, consequently, higher errors. This occurs because turbulence breaks down wake structures and re-energizes the wind. At higher ~~TI~~ I_0 values, the RMSE increases again, but so do the associated error bars. At the highest ~~TI~~ I_0 levels, a few extreme cases have no error bars, as there is only a single sample and therefore no spread. The sparsity of high ~~TI~~ I_0 values arises because the inflow cases were sampled to reflect realistic operating conditions rather than an even distribution of inflows. This is evident in Fig. 1 (g) and (h), where these extreme ~~TI~~ I_0 values are shown to be rare. Consequently, the limited number of samples for extreme ~~TI~~ I_0 values leads to the model being insufficiently trained for such scenarios, resulting in rising RMSE at the highest ~~TI~~ I_0 values. However, it is worth noting that these high TIs only occur at very low wind speeds and are extremely rare in the real world.

In ~~Fig. 12 (d~~Fig. 12 (b) and (e) the impact of the variables that govern the layout are investigated. These are the separation factor (s_wt) and the number of wind turbines (n_wt)~~and the separation factor (s_wt)~~. All the layouts are strongly inversely correlated with s_wt. The cluster and the parallel string layouts do show a slight correlation to n_wt, but that is neither the case for the single string nor the multiple string layout types.

In summary, there is a strong correlation between the model accuracy, the inflow condition and the separation of the turbines. Farms with fewer turbines and simpler wake interactions exhibit higher model accuracy, whereas larger dense farms, ~~such as~~

~~the cluster example in (a),~~ show greater errors due to increasingly complex and nonlinear inter-turbine interactions. The configuration of the farm influences this behavior: layouts with wider turbine spacings promote simpler flow patterns, while denser configurations, such as cluster and parallel string layouts, intensify wake interactions and adversely affect the GNO's predictive capability.

RBF kernels

One of the more interpretable trained parameters is the RBF kernels used to encode distances. While they do not tell the whole story, because MLPs are also used to encode features further, they do offer an indication of what is important. Figure 13 (a-b) shows both the initial and final RBF kernels for the Vj8 model. Details of the kernel centers and widths are displayed in Fig. 13 (c-d). Overall we can see that the widths are almost kept constant while the centers have been shifted. The RBF kernels have moved closer together in two clusters, and the kernel originally in the middle has joined the cluster to the left of the centre. In our formulation, the distance between a turbine receiving from an upstream sending turbine becomes negative, as shown in Eq. 5. Since the wake effect is generally more important than the blockage, it makes sense that the training has prioritized 5 kernels in the negative direction and 4 in the positive.

Computational cost & Memory consumption

For a surrogate model, it is important to understand the computational cost associated with its execution, as it should be faster than the actual model it replaces. As previously mentioned, the implementation is based on Jax, which enables Just-In-Time (JIT) compilation, making the GNO suitable for scenarios requiring multiple evaluations. For completeness, the GNO is evaluated in three configurations: (~~1~~i) a pre-JIT-compiled state, which most closely reflects the intended real-world application; (~~2~~ii) without JIT compilation; and (~~3~~iii) with JIT compilation, but performing only a single prediction. To encompass a wide range of graph sizes, the whole test set is used with a variable number of probe nodes, $n_\mathrm{p} \in \{1, 10, 100\} \cup \{1000, 2000, \ldots, 10000\}$.

The timings are compared against PyWake, where the number of probes is interpreted as the number of grid points in a flow map. All experiments are conducted using CPU resources for comparability, as PyWake does not currently support GPU acceleration. Timing results are reported in CPU hours (CPUh), as PyWake can be executed on a single CPU core, whereas the surrogate model defaults to utilizing all available 32 cores. The CPU is not saturated for small and moderate graph sizes, putting the surrogate at a slight disadvantage. The results of this analysis are presented in Fig. 14. ~~Fig.~~Figure 14 (a) shows the overall timings, illustrating how costly it is to run the model without compilation and highlighting the significant upfront cost associated with compiling it. Once compiled, the results indicate that the computational cost of running the model increases with the graph size $|G|$, and that a performance improvement of approximately $\sim 10\times$ can be expected compared to PyWake.

To further investigate the GNO, each component described in Sect. 2.3 was timed independently. The timings were measured both (~~1~~i) in a pre-JIT-compiled state and (~~2~~ii) after JIT compilation. The results are shown in Fig. 15. The results show that, at larger graph sizes, prediction time is dominated by the encoding stage. In Fig. 15 (a), the computational cost increases almost linearly on the logarithmic axis, implying exponential growth in computational cost. This behavior is partly due to how the graphs were scaled by simply adding a large number of probes, which causes $|N_\mathrm{p}|$ and $|E_\mathrm{p}|$ to grow rapidly, while $|N_\mathrm{wt}|$

and $|E_{\mathrm{wt}}|$ remain unchanged. Consequently, the wind turbine interactions at the approximator stage are unaffected. Batching graphs is proposed for a more comprehensive analysis, but has not been investigated further.

In Fig. 16, the memory consumption of the GNO is compared to the PyWake examples. As can be seen, there is no significant difference in consumed memory. However, at larger graph sizes, the memory consumption of the GNO starts growing more rapidly and, in some cases, overtakes the memory consumption of PyWake.

In summary, the GNO produces meaningful results that reflect the behavior of the flow around a given wind farm. As seen in Fig. 8, the accuracy increases with the distance behind the farm ($\cancel{\bar{x}}\tilde{x}$). Therefore, a suitable application could be the assessment of long-distance wakes from neighboring farms ~~. Furthermore, the~~ e.g., during WFLO. The GNO is particularly well-suited for this purpose, as the model is highly adaptable with respect to layout configurations. The two-stage structure of the GNO means a neighboring farm only needs to be encoded once, and the resulting latent turbine states can then be reused to evaluate wake deficits at any query location. In practice, this allows candidate turbine positions to be assessed without reprocessing the neighboring wind farm each time, saving computation when exploring many layout alternatives.

As a surrogate for engineering models, the GNO was found to predict approximately $\sim 10\times$ faster than PyWake. While this is an acceptable improvement, the speedup would increase significantly if a higher-fidelity model, such as CFD, were used as the basis. For example, if RANS were considered, a speedup between 10^4 and 10^5 seems realistic given the relative cost difference between RANS and engineering models (van der Laan et al., 2015).

4 Conclusions

The GNO offers a ~~fresh~~ new perspective on data-driven wind farm flow modeling. It has been established as a novel approach inspired by classic engineering models, demonstrating how the superposition principle can be integrated ~~into the modeling processrather than being an afterthought, as is often the case in the established literature~~directly into the learning process, rather than relying on algebraic wake superposition as single-wake surrogate models require. Furthermore, the scalability and versatility of the graph-based approach have been shown to hold across highly varied layouts and inflow conditions, indicating that the model can perform well in a general sense.

~~In , the performance of the~~ The GNO was evaluated~~. The~~ , and the results show that the model compares favorably in terms of computational cost and performs on par with PyWake in terms of memory consumption. The model was found to evaluate with a computational cost between 10^{-6} and 10^{-5} CPUh, equivalent to 3.6–36 ms on a single core, which is well within an acceptable range for practical applications. The GNO is less accurate than PyWake, ~~which is expected, as~~ as expected, since a machine learning surrogate cannot fully replicate the fidelity of its source model. Nevertheless, the accuracy of the GNO remains reasonably high when evaluated on a previously unseen test dataset. An overall RMSE of $0.353\ \mathrm{ms}^{-1}$ and a ~~MAPE~~ MANE of 0.938% were obtained. While these values constitute an acceptable error level, they do not provide the complete picture.

To gain deeper insight into the model performance, a more fine-grained assessment was conducted. First, the prediction error for representative cases was examined. This analysis demonstrated that the model accurately captures the wake shape, although some deviations persist in the predicted magnitude at medium wind speeds. A more comprehensive performance analysis revealed that the GNO performs best in scenarios with limited wake interactions. This trend is observed under inflow conditions with high TI, as well as at both low and high wind speeds. In contrast, performance deteriorates at medium wind speeds within the early part of turbine region III, where the high variability of C_{T} introduces additional complexity. Similarly, layouts with high turbine density, corresponding to a low separation factor, also result in reduced model accuracy. The impact of the total number of turbines was less pronounced, but a slight preference for smaller farms was observed for the *cluster* and *parallel string* layout types.

The GNO presented in this work ~~constitutes a breakthrough in~~ represents a methodological contribution to data-driven wind farm modeling. Nonetheless, several areas for improvement remain. Therefore, some suggestions are made for further work. The current dataset was created using engineering models, which, by design, apply a linear summation of wakes. Even though the ~~`All2AllIterative`~~ All2AllIterative scheme introduces non-linear interactions between turbines, it still merges wakes linearly. The next logical step is to utilize higher-fidelity training data that incorporates turbine interactions directly into the model. The most suitable choice would likely be RANS, as its computational cost is significantly lower than that of other CFD methods. To further reduce ~~the cost~~costs, a transfer-learning scheme could be introduced ~~, combining engineering models and~~ that combines engineering models with RANS, similar to the approach of Duthé et al. (2024).

The current GNO exclusively uses a GEN core, which, ~~in combination with the~~ combined with Softmax aggregation, provides only a rudimentary attention mechanism. However, more powerful attention mechanisms exist in both GAT and full Transformer-based attention layers. Incorporating either of these into the wind turbine processing or probe processing steps could allow the model to approximate a more complex operator. A mixed-modeling approach could also be developed, incorporating PyWake as an additional approximator alongside the GNO to predict a higher-fidelity flow field, thereby forming a multi-fidelity upscaling framework. Integrating this with a more advanced attention mechanism could enable the formation of more physically meaningful graph connections. Although such an approach would likely be more computationally expensive than the model proposed in this work, it would serve as a CFD surrogate and thus remain comparatively affordable. Furthermore, the current decoder architecture processes probe nodes in a single step without probe-to-probe communication. Introducing multiple message-passing steps between probe nodes could enable the model to capture finer spatial correlations in the flow field. However, this extension would require higher-fidelity training data to be meaningful.

Overall, the GNO presented in this work provides a ~~robust~~ baseline for efficient, data-driven flow prediction in complex wind farm environments. Further improvements can be achieved ~~through the incorporation of~~ by incorporating higher-fidelity training data, enhanced attention mechanisms, and multi-fidelity coupling strategies, ~~which have the potential to improve~~ thereby improving its predictive performance. As these developments are implemented, the GNO model ~~may emerge as an essential~~ is well-positioned to become a valuable tool for both research and industrial applications.

Code and data availability. The code used to train the GNO is publicly available at https://github.com/jenspeterschoeler/Wind-Farm-GNO. Owing to the large size of the whole dataset, only the test set is hosted on Zenodo (Schøler et al., 2025). However, the data can be re-generated using the code at https://github.com/jenspeterschoeler/Wind-farm-Graph-flow-data.

Appendix B: Wind farm simulation implementation details

Additional details about the wind farm modeling setup using PyWake are given in this appendix for interested readers.

For estimating the velocity deficit, the `NiayifarGaussianDeficit` by Niayifar and Porté-Agel (2016) is shown in Eq. B1.

$$\frac{\Delta u}{U} = \left(1 - \sqrt{1 - \frac{C_\mathrm{T}}{8\left(\frac{k^* x}{D + \varepsilon_d}\right)^2}}\right) \exp\left(-\frac{1}{2\left(\frac{k^* x}{D + \varepsilon_d}\right)^2}\left(\frac{y}{D}\right)^2\right) \tag{B1a}$$

$$\varepsilon_d = 0.2\sqrt{\beta_d}, \quad \beta_d = \frac{1}{2}\frac{1+\sqrt{1-C_\mathrm{T}}}{\sqrt{1-C_\mathrm{T}}}, \quad C_\mathrm{T} < 0.9 \tag{B1b}$$

$$k^* = a_1 I_\mathrm{wt} + a_2, \quad a_1 = 0.3837, \quad a_2 = 0.003678 \tag{B1c}$$

where x is the streamwise direction, Δu is the velocity deficit in the x-direction, C_T, is the coefficient of thrust, ε_d is a shape parameter offset dependent on C_T defined in Eq. B1b, and a_1 and a_2 are experimentally fitted parameters derived with LES data.

The `CrespoHernandez` added TI model by Crespo and Hernández (1996) depends on the induced velocity factor and the distance behind the turbine, as shown in Eq. B2.

$$I_a = 0.73\, a_m^{0.8325}\, I_0^{0.0325}\left(\frac{x}{D}\right)^{-0.32} \tag{B2a}$$

$$a_m = 0.083\, C_\mathrm{T}^3 + 0.0586\, C_\mathrm{T}^2 + 0.2460\, C_\mathrm{T} \tag{B2b}$$

Where a_m is the induced velocity factor estimated with an empirical polynomial fit of C_T to address cases with $a_m \geq 0.5$ as described by Madsen et al. (2020).

`SelfSimilarityDeficit2020` by Troldborg and Meyer Forsting (2017); Forsting et al. (2023) calculates the blockage deficit produced by individual wind turbines (Δu_b). It is based on the observation that inductions are radially self-similar for upstream distances greater than one rotor radius (R). It consists of an axial and a radial-shaped function. The newer version of the model includes an updated linear induction zone half-radius ($r_{1/2}$), which corrects the behavior of turbine induction in wind farm contexts.

$$\frac{\Delta u_b}{U} = a_0(x, C_\mathrm{T})\nu(x)\mathrm{sech}^\alpha\left(\beta_b \frac{r}{r_{1/2}(x)}\right), \tag{B3a}$$

$$\nu(x) = \left(1 + \frac{x/R}{\sqrt{1+(x/R)^2}}\right), \tag{B3b}$$

$$a_0(x, C_\mathrm{T}) = \frac{1}{2}\left(1 - \sqrt{1-\gamma(x, C_\mathrm{T})\cdot C_\mathrm{T}}\right), \tag{B3c}$$

$$\frac{r_{1/2}(x)}{R} = \lambda\cdot(x/R)+\eta, \tag{B3d}$$

$$\alpha = 8/9 \quad \beta_b = \sqrt{2} \quad \lambda = -0.672, \quad \eta = 0.4897$$

where ν is the centre-line induction and a_0 is the axial induction factor. The `SelfSimilarityDeficit2020` model introduced a $\gamma(x, C_\mathrm{T})$ function that gradually changes from a far-field expression to a near-field expression. Here, this is formulated as a function $\delta(x)$. The near- and far-field γ-functions are parameterized with $\overset{(i)}{c}_\mathrm{nf}$ for the near-field γ and $\overset{(i)}{c}_\mathrm{ff}$ for the far-field γ.

$$\gamma(x, C_\mathrm{T}) = \Bigg\{\delta(x)\cdot\left(\overset{(1)}{c}_\mathrm{nf}\cdot\sin\left(\frac{C_\mathrm{T}+\overset{(2)}{c}_\mathrm{nf}}{\overset{(3)}{c}_\mathrm{nf}}\right)+\overset{(4)}{c}_\mathrm{nf}\right)$$

$$+(1-\delta(x))\cdot\left(C_\mathrm{T}^3\overset{(1)}{c}_\mathrm{ff}+C_\mathrm{T}^2\overset{(2)}{c}_\mathrm{ff}+C_\mathrm{T}\overset{(3)}{c}_\mathrm{ff}+\overset{(4)}{c}_\mathrm{ff}\right)\Bigg\} \tag{B4a}$$

$$\overset{(1)}{c}_\mathrm{nf} = -1.381, \qquad \overset{(2)}{c}_\mathrm{nf} = 2.627,$$

$$\overset{(3)}{c}_\mathrm{nf} = -1.524, \qquad \overset{(4)}{c}_\mathrm{nf} = 1.336,$$

$$\overset{(1)}{c}_\mathrm{ff} = -0.06489, \qquad \overset{(2)}{c}_\mathrm{ff} = -0.4911,$$

$$\overset{(3)}{c}_\mathrm{ff} = -0.1577, \qquad \overset{(4)}{c}_\mathrm{ff} = 1.116,$$

(B4b)

$$\delta(x) = \begin{cases} 1 & \text{for } x/R < -6 \\ \dfrac{|\nu(x)-\nu|}{\nu(-6)-\nu(-1)} & \text{for } -6 \le x/R \le -1 \\ 0 & \text{for } -1 < x/R \end{cases} \tag{B4c}$$

Table 1. ~~Trained~~ Top 5 models from ~~the~~ grid ~~searches ranked by validation MSE$_{\mathrm{val}}$~~ search with full configuration parameters and ~~lowest best~~ training/validation metrics~~marked with b~~.

IDs			Model					Optimizer				Metrics		
Grid search	Model ID	~~Optim~~ ~~MSE$_{\mathrm{trn}}$~~ ~~MSE$_{\mathrm{val}}$~~ ~~10^{-3}~~ ~~MAE$_{\mathrm{val}}$~~ ~~$\times 10^{-3}$~~ ~~8~~ Opt. ID	Q	L_{int}	~~V~~ q_{int}	L_{dec}	q_{dec}	~~0.018~~ LR type	~~0.018~~ LR	~~1.266~~ ~~IV~~ n_{p}	~~k~~ n_G	~~2~~ MSE$_{\mathrm{trn}}$ [m^{2}s^{-2}]	~~0.018~~ MSE$_{\mathrm{val}}$ [m^{2}s^{-2}]	~~0.019~~ ~~1.291~~ MAE$_{\mathrm{val}}$ [ms^{-1}]
~~IV~~	~~j~~	~~2~~	~~0.016~~ La-tent dim.	~~0.021~~ Int. layers	~~1.412~~ ~~I~~ Int. width	~~d~~ Dec. layers	~~1~~ Dec. width	~~0.021~~	~~0.024~~	~~1.538~~ ~~III~~ Probes/ batch	~~a~~ Graphs/ batch	~~6~~	~~0.020~~	~~0.026~~ ~~1.978~~
~~I~~ V	~~h~~ j	~~1~~ 8	~~0.018~~ 100	~~0.026~~ 2	~~1.482~~ ~~I~~ 250	~~a~~ 3	~~1~~ 350	~~0.024~~ PC†	~~0.027~~ 5×10^{-3}	~~1.788~~ ~~I~~ 500	~~g~~ 2	~~1~~ 0.015	~~0.021~~ **0.016**	~~0.027~~ ~~1.692~~ **0.037**
~~HI~~V	~~e~~ k	~~1~~ ~~0.027~~ ~~0.027~~ ~~2.073~~ ~~III~~ ~~a~~ ~~5~~ ~~0.029~~ ~~0.027~~ ~~1.916~~ ~~II~~ ~~h~~ 2	~~0.019~~ 50	~~0.028~~ 3	~~1.707~~ ~~III~~ 350	~~a~~ 4	~~0.021~~ 350	~~0.031~~ PC	~~2.211~~ ~~III~~ 5×10^{-3}	~~i~~ 200	~~6~~ 5	~~0.021~~ 0.015	~~0.032~~ 0.016	~~2.054~~ 0.038
~~I f 1~~ ~~0.026~~ ~~0.032~~ ~~1.899~~ IV	~~f~~ j	2	~~0.030~~ 100	~~0.034~~ 2	~~2.157~~ ~~V~~ 250	~~k~~ 3	~~8~~ 350	~~0.022~~ PC	~~0.035~~ 5×10^{-3}	~~1.947~~ ~~IV~~ 200	~~j~~ 5	~~7~~ **0.014**	~~0.021~~ 0.018	~~0.035~~ ~~1.967~~ 0.041
~~III~~ ~~i~~ ~~4~~ ~~0.022~~ ~~0.040~~ ~~2.804~~ ~~V~~ ~~f~~ ~~9~~	~~b~~ d	1	~~0.025~~ 150	~~0.057~~ ~~2.610~~ ~~V f 8~~ ~~0.028~~ ~~0.062~~ ~~2.201~~ ~~III~~ ~~i~~	~~0.026~~ 200	~~0.062~~ 3	~~3.938~~ ~~V~~ 250	~~j~~ PC†	~~9~~ 5×10^{-3}	~~0.023~~ 200	~~0.063~~ ~~2.208~~ ~~I~~ ~~e~~ ~~1~~ ~~0.020~~ ~~0.075~~ ~~2.582~~ ~~III~~ ~~i~~ 5	~~0.023~~ 0.018	~~0.088~~ 0.02	~~4.807~~ 0.045

Table 2. Test set error metrics for the five best performing models based on the grid search and a no-wake baseline, with the best metrics marked in bold. The naive baseline predicts that the flow everywhere is equal to the free stream velocity (i.e., $u' = U$).

Grid search	Model ID	~~Optimizer~~ Opt. ID	MSE [$\mathrm{m^2 s^{-2}}$]	MAE [$\mathrm{ms^{-1}}$]	RMSE [$\mathrm{ms^{-1}}$]	~~MAPE~~ MANE [%]
V	j	8	**0.124**	0.105	**0.353**	0.938
IV	k	2	0.127	0.106	0.356	0.957
IV	j	2	0.127	**0.105**	0.357	**0.937**
I	d	1	0.133	0.112	0.365	1.010
III	a	6	0.130	0.106	0.361	0.959
Naive free stream baseline			148.6	11.87	12.19	109.7

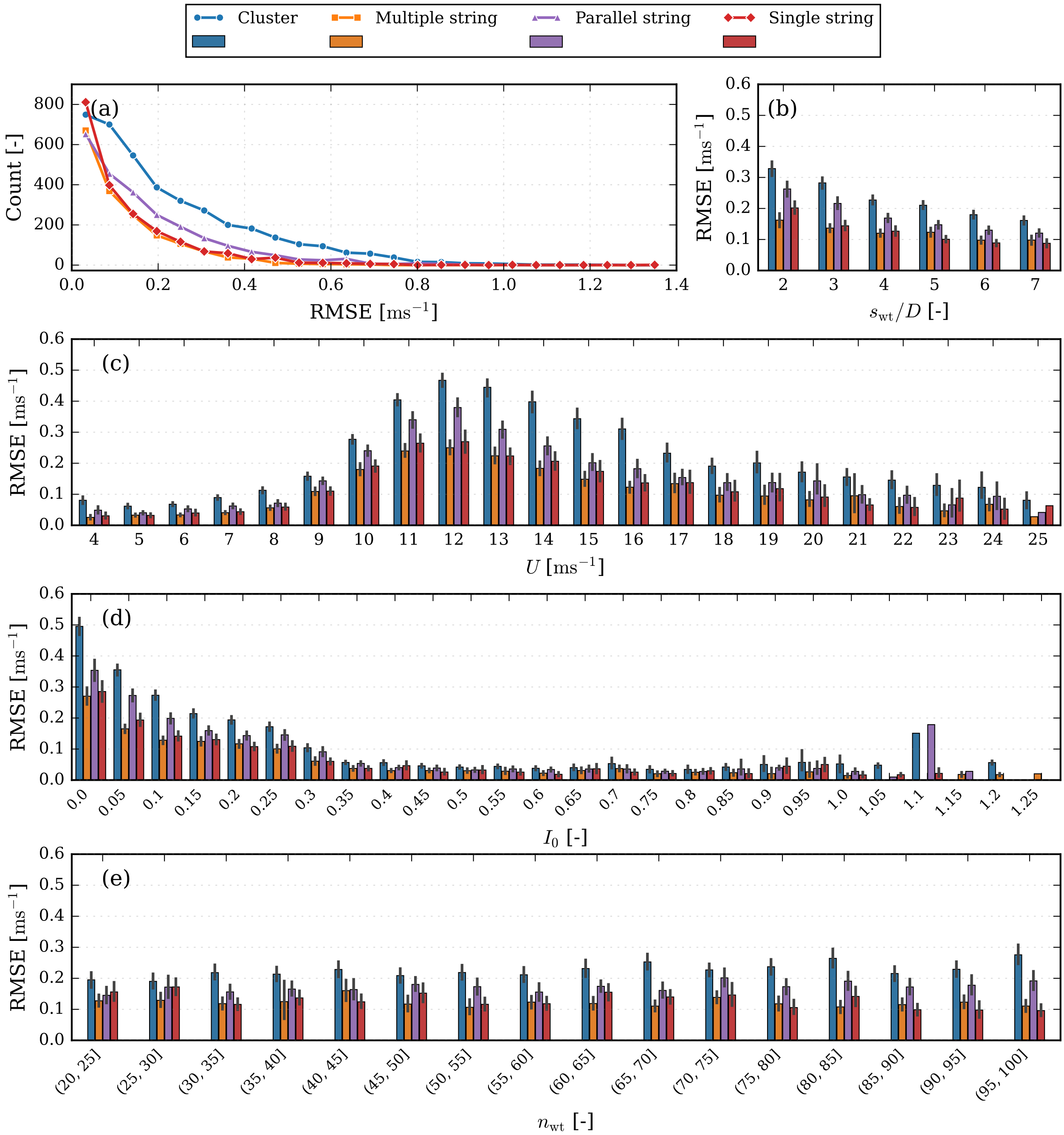

Figure 12. ~~(a)-(e)~~ RMSE metrics for different layout types. (a) RMSE ~~PDFs estimated with KDE~~binned error counts. Binned RMSE (b) with respect to the separation factor s_{wt}, (c) at different free stream velocities U, (~~e~~d) across different TI, and (~~d~~e) against the number of wind turbines n_{wt}~~, and (e) with respect to the separation factor s_{wt}~~

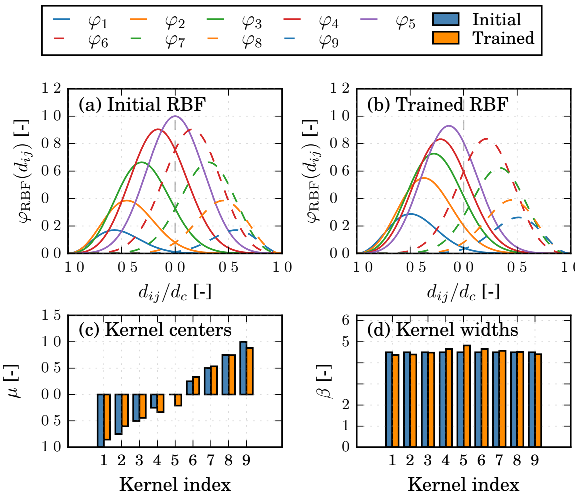

Figure 13. RBF kernels (a) before and (b) after training, with (c) kernel centers μ and (d) kernel widths β shown before and after training.

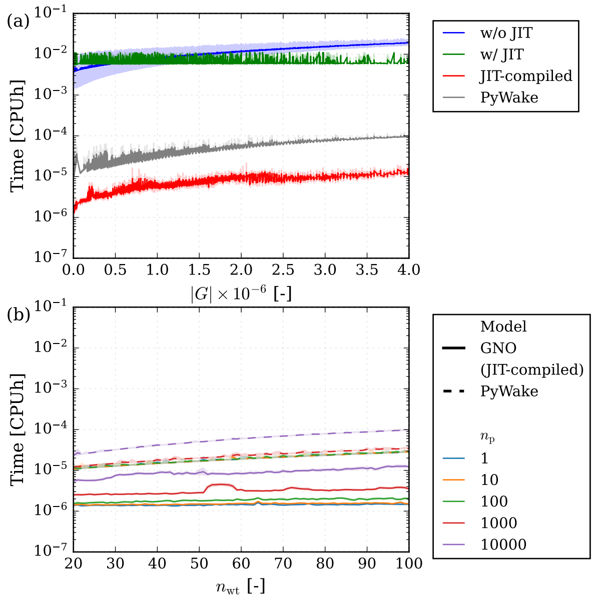

Figure 14. Model computational cost in terms of CPU hours. (a) Model cost of GNO in different JIT states compared to PyWake for varying graph sizes $|G|$. (b) Model computational cost in JIT-compiled state shown with different numbers of probes compared to PyWake for an increasing number of wind turbines.

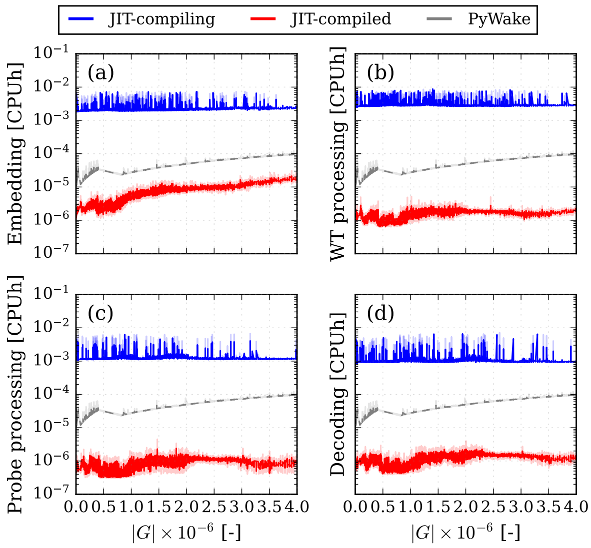

Figure 15. Timings of the GNO components. (a) Encoder $\mathcal{E}$, (b) Approximator $\mathcal{A}$, (c) Decoder $\mathcal{D}$ part 1: probe processing, and (d) Decoder $\mathcal{D}$ part 2: node decoding.

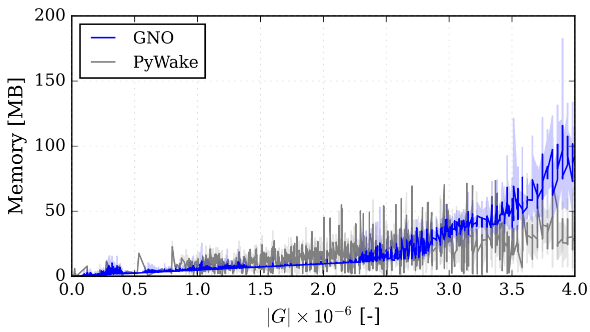

Figure 16. Memory consumption

Appendix C: Neural Network Components

Dropout

Our version of dropout uses inverse probability scaling, so the entire Dropout layer can be ignored during inference. During training Dropout is applied both after Eq. 7a and Eq. 7b,

$$z_j^{(l)} \sim \mathrm{Bernoulli}(P_{\mathrm{D}}), \quad j = 1, 2, \ldots, N_a \tag{C1a}$$

$$\tilde{\boldsymbol{\xi}}^{(l)} = \boldsymbol{z}^{(l)} \odot \frac{1}{1 - P_{\mathrm{D}}} \cdot \boldsymbol{\xi}^{(l)}, \quad l = 1, 2, \ldots, L \tag{C1b}$$

here $\boldsymbol{z}^{(l)}$ is a Dropout mask consisting of zeros and ones, created with $\mathrm{Bernoulli}(P_{\mathrm{D}})$ the Bernoulli distribution given the Dropout probability (P_{D}), N_a is the number of activations in layer l, the operator $\odot$ denotes the Hadamard element-wise product, and $\tilde{\boldsymbol{\xi}}^{(l)}$ is the resultant hidden state with Dropout applied.

Layer normalization

The formulation of layer normalization used is shown in Eq. C2.

$$\mu_{\mathrm{LN}}^{(L)} = \frac{1}{H} \sum_{i=1}^{H} \xi_i^{(L)}, \qquad \sigma_{\mathrm{LN}}^{(L)} = \sqrt{\frac{1}{H} \sum_{i=1}^{H} \left(\xi_i^{(L)} - \mu_{\mathrm{LN}}^{(L)} \right)^2}, \tag{C2a}$$

$$\bar{\boldsymbol{\xi}}^{(L)} = \boldsymbol{s}_{\mathrm{LN}} \odot \frac{\boldsymbol{\xi}^{(L)} - \mu_{\mathrm{LN}}^{(L)}}{\sigma_{\mathrm{LN}}^{(L)} + \varepsilon} + \boldsymbol{b}_{\mathrm{LN}}, \tag{C2b}$$

Where $\mu_{\mathrm{LN}}^{(L)}$ and $\sigma_{\mathrm{LN}}^{(L)}$ denote the layer mean and standard deviation, $\xi_i^{(L)}$ is the ith activation at the last MLP layer L, and $\bar{\boldsymbol{\xi}}^{(L)}$ represents the layer-normalized hidden states, while $\boldsymbol{s}_{\mathrm{LN}}$ and $\boldsymbol{b}_{\mathrm{LN}}$ are trainable scale and bias parameters, and $\varepsilon = 1 \times 10^{-6}$ is added for numerical stability.

Appendix D: GNO Dataloading

As with other neural operators, the data comes in triplets~~, cf. branch input~~: branch input (the wind farm graph encoding turbine positions and inflow conditions), trunk input (the probe graph encoding query locations), and target output. For a broader context, see e.g. (Lu et al., 2021; Seidman et al., 2022). Because both the GNO branch and trunk are GNNs, the branch input is a graph, and the trunk is a second graph with a separate edge configuration. One of the core strengths of GNNs is the ability to process graphs of different sizes. However, to leverage the efficiency of the Jax framework, it is necessary to use JIT compilation, which requires inputs to have fixed sizes and shapes. To create graphs of fixed sizes and shapes, graphs are dynamically batched together until they reach a maximum size, defined by the number of graphs, total nodes, and total edges.

Batching graphs differ from conventional batching. In traditional batching, inputs share identical dimensions, and batching is performed along a new dimension. However, because the number of nodes and edges varies for each graph, batching is achieved by combining multiple graphs into a single larger graph. This approach preserves the sub-graphs individual characteristics by ensuring no edges between sub-graphs, thus preventing interactions during training and inference. Dynamic batching involves adjusting the number of graphs in each batch to fit within the size constraints rather than using a fixed batch size as in conventional batching. When dynamic batching is employed, adding another graph to the batch is prohibited once any maximum size limit is reached, as doing so would violate the constraints. Instead, padding with empty graphs, edges, and nodes is used to meet the target size.

An additional challenge arises because Jax does not currently offer a native data loader. At the time of writing, the most popular and mature Python framework for GNNs is PyG. Consequently, the PyG data loader has been used to load data in parallel from the file system into memory. Afterward, the graphs are converted to the Jraph format. Finally, dynamic batching and padding are performed sequentially on the CPU before the data is transferred to the GPU for processing. The current performance bottleneck lies in the sequential dynamic batching process. Parallelizing this step is challenging due to the variable sizes of the graphs. However, this method is still significantly more efficient than loading data from the file system sequentially. The process is visualized in Fig. D1.

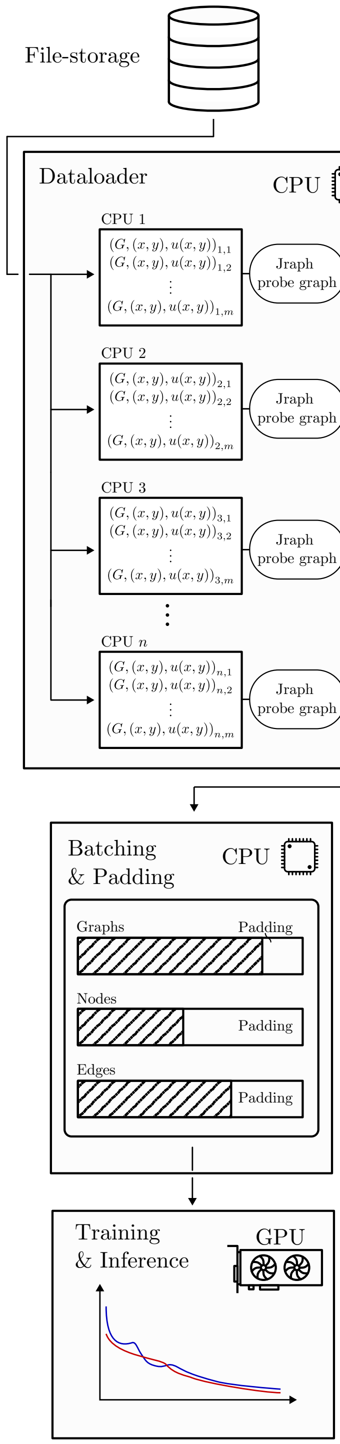

Figure D1. Flowchart diagram illustrating the steps from on-disk memory, constructing the Jraph compatible probe graph in parallel, batching, padding, and moving the resultant batched and padded probe graph to video memory.

Appendix E: Hyper-parameter search detail

Table E1. Model and optimizer configurations used in the grid search.

Model configurations

ID	Latent dim. Q	L_{int}	q_{int}	L_{dec}	q_{dec}
a	150	2	100	3	150
b	150	2	100	3	250
c	150	2	200	3	150
d	150	2	200	3	250
e	250	2	100	3	150
f	250	2	100	3	250
g	250	2	200	3	150
h	250	2	200	3	250
i	100	3	75	3	250
j	100	2	250	3	350
k	50	3	350	4	350

Optimizer configurations

ID	LR type	LR	Triggers	n_{p}	n_G
1	Piecewise constant†	0.005	[500, 1000]	200	5
2	Piecewise constant	0.005	[75, 150]	200	5
3	Constant	0.001	-	200	5
4	Piecewise constant	0.001	[75, 150]	200	5
5	Piecewise constant	0.005	[75, 150]	300	4
6	Piecewise constant	0.001	[75, 150]	300	4
7		0.010	[75, 150]	[illegible]	5

Table E2. Trained models from the grid search ranked by validation MSE_{val}.

Configuration			Metrics		
Grid search	Model ID	Opt. ID	MSE_{trn} [m^2s^{-2}]	MSE_{val} [m^2s^{-2}]	MAE_{val} [ms^{-1}]
V	j	8	0.015	**0.016**	**0.037**
IV	k	2	0.015	**0.016**	0.038
IV	j	2	**0.014**	0.018	0.041
I	d	1	0.018	0.020	0.045
III	a	6	0.017	0.022	0.058
I	h	1	0.016	0.022	0.043
I	a	1	0.021	0.022	0.052
I	g	1	0.018	0.023	0.049
I	e	1	0.023	0.023	0.060
III	a	5	0.024	0.023	0.056
II	h	2	0.016	0.024	0.050
III	a	4	0.017	0.026	0.064
III	i	6	0.018	0.027	0.060
I	f	1	0.022	0.027	0.055
IV	f	2	0.025	0.028	0.063
V	k	8	0.018	0.029	0.057
IV	j	7	0.017	0.030	0.057
III	i	4	0.019	0.034	0.082
V	f	9	0.030	0.039	0.057
II	f	2	0.021	0.039	0.068
I	b	1	0.021	0.049	0.076

Appendix F: Wind turbine predictions visualization

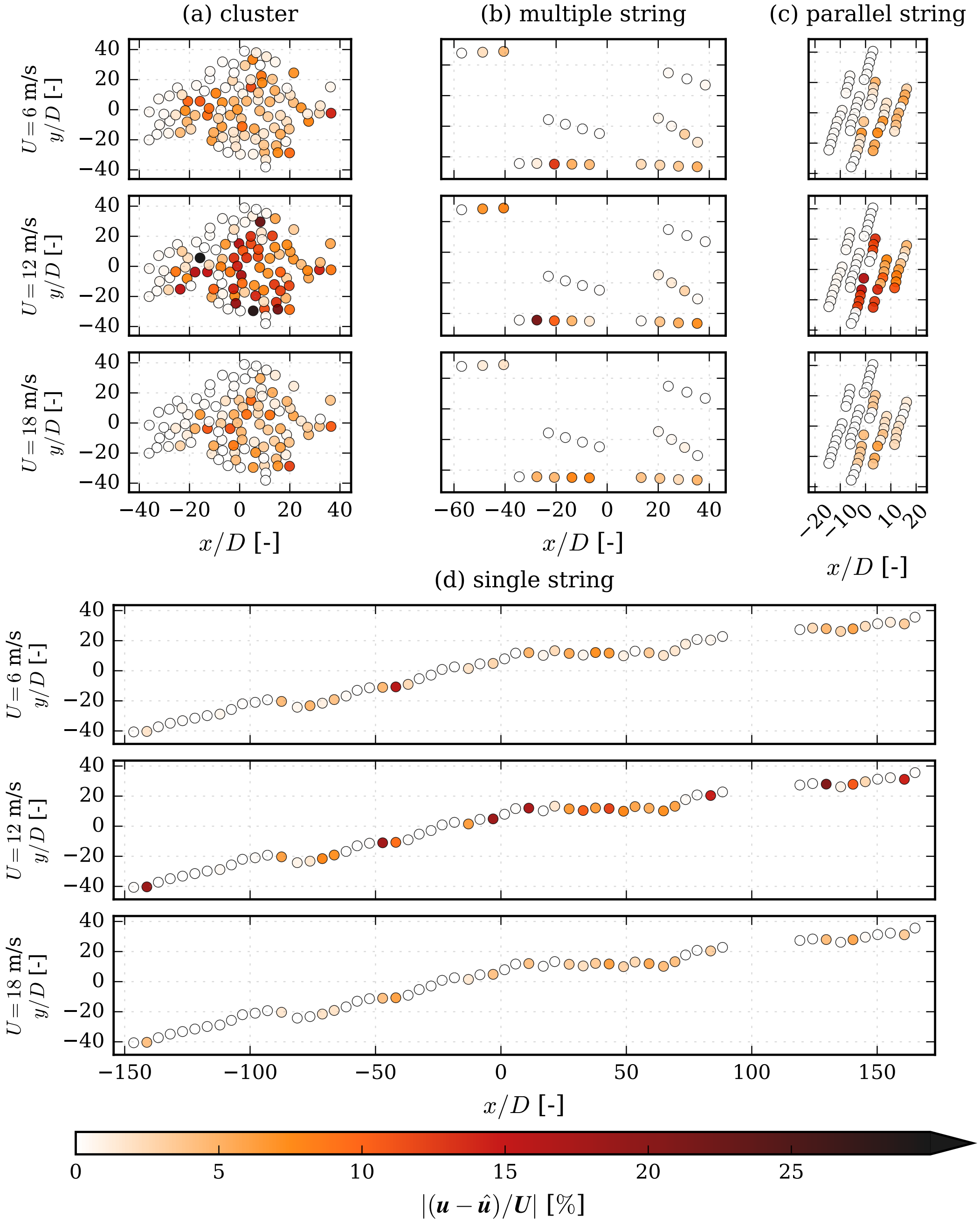

Figure F1. Maximum absolute errors at each wind turbine, for each wind speed in $\boldsymbol{U} = [6, 12, 18]^\top$ ms^{-1} and $I_0 = 5\%$

Author contributions. JPSCH and PIRE conceived the research; JPSCH developed the methodology; JPSCH developed the code for the GNO and generated the required data; JPSCH performed the formal analysis; PIRE and JQ supervised the work; JPSCH, JQ, FPWR wrote the original draft; and JPSCH, JQ, FPWR and PIRE reviewed the draft.

Competing interests. The authors declare that they have no competing interests.

Acknowledgements. This work was partially funded by TotalEnergies under the "Inter-Farm Interactions (IFI)" project. The authors thank Mikkel N. Schmidt at DTU Compute for providing valuable feedback on graph neural networks.

Model training was performed on the Sophia cluster (Technical University of Denmark, 2019), without which the GNO could not have been trained. The authors also acknowledge the use of AI tools in preparing this manuscript and the associated code. In particular, ChatGPT, Grammarly, and GitHub Copilot were used for proofreading and code auto-completion.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E.: Layer Normalization, http://arxiv.org/abs/1607.06450, 2016.

Bak, C. ., Zahle, F. ., Bitsche, R. ., Kim, T. ., Yde, A. ., Henriksen, L. C., Hansen, M. H., Blasques, J. P. A. A., Gaunaa, M. ., and Natarajan, A.: The DTU 10-MW Reference Wind Turbine, in: Danish Wind Power Research, 2013.

Bastankhah, M. and Porté-Agel, F.: A new analytical model for wind-turbine wakes, Renewable Energy, 70, 116–123, https://doi.org/10.1016/j.renene.2014.01.002, 2014.

Bastankhah, M., Welch, B. L., Martínez-Tossas, L. A., King, J., and Fleming, P.: Analytical solution for the cumulative wake of wind turbines in wind farms, Journal of Fluid Mechanics, 911, A53, https://doi.org/10.1017/jfm.2020.1037, 2021.

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

Bleeg, J.: A Graph Neural Network Surrogate Model for the Prediction of Turbine Interaction Loss, in: Journal of Physics: Conference Series, vol. 1618, IOP Publishing Ltd, ISSN 17426596, https://doi.org/10.1088/1742-6596/1618/6/062054, 2020.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q.: JAX: composable transformations of Python+NumPy programs, http://github.com/jax-ml/jax, 2018.

Crespo, A. and Hernández, J.: Turbulence characteristics in wind-turbine wakes, Journal of Wind Engineering and Industrial Aerodynamics, 61, 71–85, 1996.

de Santos, F. N., Duthé, G., Abdallah, I., Élouan Réthoré, P., Weijtjens, W., Chatzi, E., and Devriendt, C.: Multivariate prediction on wake-affected wind turbines using graph neural networks, Journal of Physics: Conference Series, 2647(11), 112 006, https://doi.org/10.3929/ethz-b-000674010, 2024.

Delaunay, B.: Sur la sphère vide, Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na, pp. 793–800, 1934.

Dimitrov, N., Kelly, M. C., Vignaroli, A., and Berg, J.: From wind to loads: wind turbine site-specific load estimation with surrogate models trained on high-fidelity load databases, Wind Energy Science, 3, 767–790, https://doi.org/10.5194/wes-3-767-2018, 2018.

Duthé, G., de Nolasco Santos, F., Abdallah, I., Élouan Réthore, P., Weijtjens, W., Chatzi, E., and Devriendt, C.: Local flow and loads estimation on wake-affected wind turbines using graph neural networks and PyWake, Journal of Physics: Conference Series, 2505, 012 014, https://doi.org/10.1088/1742-6596/2505/1/012014, 2023.

Duthé, G., de N Santos, F., Abdallah, I., Weijtjens, W., Devriendt, C., and Chatzi, E.: Flexible multi-fidelity framework for load estimation of wind farms through graph neural networks and transfer learning, Data-Centric Engineering, 5, e29, https://doi.org/10.1017/dce.2024.35, 2024.

Euler, L.: Commentarii Academiae Scientiarum Petropolitanae, chap. Solutio problematis ad geometriam situs pertinentis, p. 128–140, Published by Michael Behrend at Cantab, English translation available at https://www.cantab.net/users/michael.behrend/repubs/maze_maths/pages/euler.html, 1741.

Fey, M. and Lenssen, J. E.: Fast Graph Representation Learning with PyTorch Geometric, http://arxiv.org/abs/1903.02428, 2019.

Forsting, A. R. M., Diaz, G. P. N., Segalini, A., Andersen, S. J., and Ivanell, S.: On the accuracy of predicting wind-farm blockage, Renewable Energy, 214, 114–129, https://doi.org/10.1016/j.renene.2023.05.129, 2023.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E.: Neural Message Passing for Quantum Chemistry, http://arxiv.org/abs/1704.01212, 2017.

Godwin, J., Keck, T., Battaglia, P., Bapst, V., Kipf, T., Li, Y., Stachenfeld, K., Veličković, P., and Sanchez-Gonzalez, A.: Jraph: A library for graph neural networks in jax., http://github.com/deepmind/jraph, 2020.

Göçmen, T., van der Laan, P., Réthoré, P.-E., Diaz, A. P., Larsen, G. C., and Ott, S.: Wind turbine wake models developed at the technical university of Denmark: A review, Renewable and Sustainable Energy Reviews, 60, 752–769, https://doi.org/10.1016/j.rser.2016.01.113, 2016.

Harrison-Atlas, D., Glaws, A., King, R. N., and Lantz, E.: Artificial intelligence-aided wind plant optimization for nationwide evaluation of land use and economic benefits of wake steering, Nature Energy, 9, 735–749, https://doi.org/10.1038/s41560-024-01516-8, 2024.

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M.: Flax: A neural network library and ecosystem for JAX, http://github.com/google/flax, 2024.

IEC 61400-1:2019 ed. 4: Wind energy generation systems - Part 1: Design requirements (IEC 61400-1:2019 ed. 4), www.ds.dk, 2019.

Joe, S. and Kuo, F. Y.: Constructing Sobol Sequences with Better Two-Dimensional Projections, SIAM Journal on Scientific Computing, 30, 2635–2654, https://doi.org/10.1137/070709359, 2008.

Jørgensen, P. B. and Bhowmik, A.: Equivariant graph neural networks for fast electron density estimation of molecules, liquids, and solids, npj Computational Materials, 8, 183, https://doi.org/10.1038/s41524-022-00863-y, 2022.

Kainz, S., Quick, J., de Alencar, M. S., Moreno, S. S. P., Dykes, K., Bay, C., Zaaijer, M. B., and Bortolotti, P.: The IEA Wind 740-10-MW Reference Offshore Wind Plants, Tech. rep., IEA Task 55, https://github.com/IEAWindTask37/IEA-Wind-740-10-ROWP., 2024.

Katic, I. ., Højstrup, J. ., and Jensen, N. O.: A Simple Model for Cluster Efficiency, Tech. rep., Risø National Laboratory, 1987.

Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, Preprint, http://arxiv.org/abs/1412.6980, 2014.

Li, G., Xiong, C., Thabet, A., and Ghanem, B.: DeeperGCN: All You Need to Train Deeper GCNs, http://arxiv.org/abs/2006.07739, 2020a.

Li, S., Zhang, M., and Piggott, M. D.: End-to-end Wind Turbine Wake Modelling with Deep Graph Representation Learning, Applied Energy, 339, https://doi.org/10.1016/j.apenergy.2023.120928, 2022.

Li, S., Robert, A., Faisal, A. A., and Piggott, M. D.: Learning to optimise wind farms with graph transformers, Applied Energy, 359, 122 758, https://doi.org/10.1016/j.apenergy.2024.122758, 2024.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A.: Neural Operator: Graph Kernel Network for Partial Differential Equations, http://arxiv.org/abs/2003.03485, 2020b.

Liew, J., Göçmen, T., Lio, A. W. H., and Larsen, G. C.: Extending the dynamic wake meandering model in HAWC2Farm: a comparison with field measurements at the Lillgrund wind farm, Wind Energy Science, 8, 1387–1402, https://doi.org/10.5194/wes-8-1387-2023, 2023.

Lissaman, P. B. S.: Energy Effectiveness of Arbitrary Arrays of Wind Turbines, Journal of Energy, 3, 323–328, https://doi.org/10.2514/3.62441, 1979.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E.: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nature Machine Intelligence, 3, 218–229, https://doi.org/10.1038/s42256-021-00302-5, 2021.

Madsen, H. A., Larsen, T. J., Pirrung, G. R., Li, A., and Zahle, F.: Implementation of the blade element momentum model on a polar grid and its aeroelastic load impact, Wind Energy Science, 5, 1–27, https://doi.org/10.5194/wes-5-1-2020, 2020.

Niayifar, A. and Porté-Agel, F.: Analytical modeling of wind farms: A new approach for power prediction, Energies, 9, https://doi.org/10.3390/en9090741, 2016.

O’Rourke, J.: Computational Geometry, Annual Review of Computer Science, 3, 389–411, https://doi.org/10.1146/annurev.cs.03.060188.002133, 1988.

Park, J. and Park, J.: Physics-induced graph neural network: An application to wind-farm power estimation, Energy, 187, 115 883, https://doi.org/10.1016/j.energy.2019.115883, 2019.

Pedersen, M. M., Forsting, A. M., Paul van der Laan, R. R., Romàn, L. A. A., Risco, J. C., Friis-Møller, M., Quick, J., Schøler, J. P., Rodrigues, R. V., Olsen, B. T., and Réthoré, P.-E.: PyWake 2.5.0: An open-source wind farm simulation tool, https://gitlab.windenergy.dtu.dk/TOPFARM/PyWake, 2023.

Porté-Agel, F., Bastankhah, M., and Shamsoddin, S.: Wind-Turbine and Wind-Farm Flows: A Review, Boundary-Layer Meteorology, 174, 1–59, https://doi.org/10.1007/s10546-019-00473-0, 2020.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G.: The Graph Neural Network Model, IEEE Transactions on Neural Networks, 20, 61–80, https://doi.org/10.1109/TNN.2008.2005605, 2009.

Schøler, J. P., Quick, J., Réthoré, P.-E., and Rasmussen, F. P. W.: Wind farm: Graph flow test data, https://doi.org/10.5281/zenodo.17671257, 2025.

Scott, D.: Multivariate Density Estimation: Theory, Practice, and Visualization, Wiley Series in Probability and Statistics, Wiley, ISBN 9780471697558, 1992.

Seidman, J. H., Kissas, G., Perdikaris, P., and Pappas, G. J.: NOMAD: Nonlinear Manifold Decoders for Operator Learning, http://arxiv.org/abs/2206.03551, 2022.

Srivastava, N., Hinton, G., Krizhevsky, A., and Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Tech. rep., Department of Computer Science University of Toronto, 2014.

Sun, Y., Moya, C., Lin, G., and Yue, M.: DeepGraphONet: A Deep Graph Operator Network to Learn and Zero-shot Transfer the Dynamic Response of Networked Systems, http://arxiv.org/abs/2209.10622, 2022.

Technical University of Denmark: Sophia HPC Cluster, https://doi.org/10.57940/FAFC-6M81, 2019.

Troldborg, N. and Meyer Forsting, A. R.: A simple model of the wind turbine induction zone derived from numerical simulations, Wind Energy, 20, 2011–2020, https://doi.org/10.1002/we.2137, 2017.

van der Laan, M. P., Sørensen, N. N., Réthoré, P. E., Mann, J., Kelly, M. C., and Troldborg, N.: The k-ε-f_p model applied to double wind turbine wakes using different actuator disk force methods, Wind Energy, 18, 2223–2240, https://doi.org/10.1002/we.1816, 2015.

Voutsinas, S., Rados, K., and Zervos, A.: On the Analysis of Wake Effects in Wind Parks, Tech. rep., National Technical University of Athens, Laboratory of Aerodynamics, 1990.

Yu, M., Zhang, Z., Li, X., Yu, J., Gao, J., Liu, Z., You, B., Zheng, X., and Yu, R.: Superposition Graph Neural Network for offshore wind power prediction, Future Generation Computer Systems, 113, 145–157, https://doi.org/10.1016/j.future.2020.06.024, 2020.

Zehtabiyan-Rezaie, N., Iosifidis, A., and Abkar, M.: Data-driven fluid mechanics of wind farms: A review, Journal of Renewable and Sustainable Energy, 14, https://doi.org/10.1063/5.0091980, 2022.

Zong, H. and Porté-Agel, F.: A momentum-conserving wake superposition method for wind farm power prediction, Journal of Fluid Mechanics, 889, A8, https://doi.org/10.1017/jfm.2020.77, 2020.

Ødegaard Bentsen, L., Warakagoda, N. D., Stenbro, R., and Engelstad, P.: Wind Park Power Prediction: Attention-Based Graph Networks and Deep Learning to Capture Wake Losses, Journal of Physics: Conference Series, 2265, 022 035, https://doi.org/10.1088/1742-6596/2265/2/022035, 2022.