https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

Open Access WIND ENERGY SCIENCE DISCUSSIONS

Convolutional versus graph-based surrogate models for inter-farm wake prediction using multi-fidelity transfer learning

Jens Peter Schøler[†1], Frederik Peder Weilmann Rasmussen[†1], M. Paul van der Laan[1], Alfredo Peña[1], and Pierre-Elouan Réthoré[1]

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

Correspondence: Jens Peter Schøler[†] (jpsch@dtu.dk) and Frederik Peder Weilmann Rasmussen[†] (fpwra@dtu.dk)

Abstract. Accurate prediction of wind farm wake interactions is important for energy yield assessment, as offshore wind farms increasingly operate in close proximity to one another. This work presents a systematic comparison of two neural network surrogate models for inter-farm wake deficit prediction: a convolutional neural network (CNN) based attention residual U-Net (ARU-Net) and a graph neural network (GNN) based graph neural operator (GNO). Both architectures are trained using multi-fidelity transfer learning. First, pre-trained on low-fidelity engineering model simulations, and second fine-tuned on high-fidelity Reynolds-averaged Navier-Stokes actuator wind farm (RANS-AWF) data. The models are evaluated on procedurally generated wind farm layouts spanning diverse farm sizes, turbine spacings, wind speeds, and ambient turbulence intensities. Both architectures achieve high prediction accuracy but exhibit complementary strengths: evaluated over the wake region ($\delta_w \geq 10^{-3}\,\mathrm{m\,s^{-1}}$) and averaged across both evaluation grids, the GNO achieves a lower RMSE (0.024 vs. 0.028 $\mathrm{m\,s^{-1}}$), while the ARU-Net attains a higher F_1 score (0.98 vs. 0.91), reflecting its superior wake boundary capture. Transfer learning substantially benefits the ARU-Net, while the GNO shows only marginal improvement.

[†]These authors contributed equally to this work.

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

1 Introduction

As global energy demand increases and a number of areas suitable for offshore wind development become increasingly densely populated, the complexity of accurately modelling these environments has increased significantly. Multiple wind farms now operate in close proximity, creating complex flow interactions that extend far beyond single-farm boundaries. However, wake interactions between neighbouring farms remain poorly predicted by current engineering models, posing an important challenge for accurate energy yield assessment and wind farm layout optimisation (WFLO).

Wind farm wakes, defined as downstream regions of a wind farm with reduced wind speed and increased turbulence intensity (TI), have been observed to persist over remarkably long distances. Multiple independent measurement methodologies, including ground-based radar (Nygaard and Newcombe, 2018), scanning lidar systems (Schneemann et al., 2020; Cañadillas et al., 2022), synthetic aperture radar (SAR) satellite imagery (Hasager et al., 2015; Djath et al., 2018), and airborne measurement campaigns (Platis et al., 2018), have confirmed that wind farm wakes can extend far outside the wind farm boundaries and, under certain atmospheric conditions, beyond tens of kilometers, particularly for large wind farm clusters operating in stable atmospheric boundary layers. These observations challenge the fundamental assumptions of conventional engineering models, which are primarily developed and calibrated for intra-farm wake effects at distances of tens of rotor diameters.

Engineering wake models have served as the workhorses of wind farm design and energy yield assessment for the last few decades, offering computational efficiency and enabling the rapid evaluation of thousands of layout configurations during WFLO. Models such as the Park model by Jensen (1983), the Gaussian wake model of Bastankhah and Porté-Agel (2014), and the Turbulence Optimised Park (TurbOPark) model developed by Nygaard et al. (2020, 2022) are popular representative examples of analytical wake models.

The atmospheric conditions play an important role in wake persistence. Low turbulence, for instance, suppresses wake mixing, limiting downstream wake recovery, while high TI increases turbulent mixing, accelerating wake recovery. The interplay among atmospheric conditions, farm size, turbine spacing, and wake superposition creates a multidimensional parameter space that engineering models struggle to predict accurately.

Engineering wake models have fundamental limitations that become increasingly problematic at inter-farm scales. First, they assume steady-state, axisymmetric wake structures, neglecting the inherent unsteadiness and three-dimensional complexity of turbulent wakes. Second, they rely on algebraic superposition methods, typically linear or squared-sum approaches, to combine multiple single-turbine wakes. While computationally efficient, superposition does not capture the non-linear wake interactions that occur when multiple wakes merge and interact with the atmospheric boundary layer. Third, these models neglect large-scale atmospheric effects, including Coriolis forces, atmospheric stability, and mesoscale flow phenomena that become relevant at distances of tens of kilometres. Even when calibrated to specific sites or conditions, engineering models lack the physical fidelity to generalise across the diverse operating conditions met in practice. When designing a wind farm, consideration must be given to neighbouring wind farms, as wake losses from neighbouring projects may affect profitability. Multiple neighbours may be present simultaneously, and at the time of freezing the design, their layouts might not yet be finalised and subsequently change. Accounting for these inter-farm effects is therefore challenging due to the inherent uncer-

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

tainty surrounding the configuration of neighbouring projects, and a fast, accurate model is needed to efficiently evaluate many candidate designs.

Data-driven surrogate models have gained popularity as they bridge the accuracy-efficiency gap in wind farm flow modelling. By learning mappings from input parameters, like wind farm layout and inflow conditions, to output flow fields directly from simulation data, these models can achieve prediction speeds comparable to engineering models while capturing more of the complex physics of the training data. However, the accuracy of data-driven surrogates is fundamentally constrained by the fidelity of the training data. Therefore, a model trained exclusively on engineering model outputs cannot exceed the engineering accuracy of the model, regardless of how sophisticated the fully data-driven neural network architecture is.

Reynolds averaged Navier-Stokes (RANS) or large eddy simulations (LES) are more physically accurate, but computationally much more expensive than low-fidelity engineering models. On the other hand, idealised RANS and LES cannot account for the large and mesoscale scales of the atmospheric flow. Therefore, a common method for simulating inter-farm wake effects is to use wind-farm parameterisations implemented in numerical weather prediction models. Although such models realistically simulate atmospheric processes across large spatial domains, their spatial resolutions are too coarse to account for individual wind turbine wakes. Therefore, the accuracy of these parameterisations is currently a matter of debate (Peña et al., 2022; García-Santiago et al., 2024). This observation motivates multi-fidelity transfer learning approaches that utilise both the abundant low-fidelity data and sparse high-fidelity data. The strategy involves pre-training neural network models on large datasets generated by computationally inexpensive engineering models, thereby establishing robust representations of fundamental wake physics and layout-to-flow mappings. These pre-trained models can then be fine-tuned on smaller datasets of high-fidelity simulations, enabling them to learn corrections to engineering model predictions while avoiding overfitting to limited data.

In this work, two principal families of neural network architectures have been applied to wind farm flow prediction: convolutional neural networks (CNNs) and graph neural networks (GNNs). These architectures differ in how they interpret spatial information and how they handle wind farm geometry. CNNs operate with flow fields on a regular grid, treating the spatial domain as a regular, image-like structure, while the GNNs operate on the more adaptive structure of a graph.

Both neural network architectures can be trained to predict the flow field around different layouts, but the two methods use different methods of encoding turbine placement. In most CNN-based approaches, it is necessary to rasterise locations onto a fixed spatial domain, whereas turbine positions can be placed exactly at graph nodes, as the graph is adaptive. However, this depends on how the graph is constructed. Graphs can broadly be divided into two categories: *extrinsic* graphs, where the nodes and edges form a spatial mesh, and *intrinsic* graphs, where the nodes correspond directly to the physical entities of interest, in the context of wind farms, typically the turbines themselves.

Multiple previous studies have applied CNN-based architectures to aerodynamic and wind flow prediction tasks. Thuerey et al. (2019) demonstrated that a U-Net encoder-decoder architecture could serve as an effective RANS surrogate for predicting velocity and pressure fields around airfoils, establishing the viability of convolutional auto-encoders (CAEs) for learning complex flow patterns from simulation data. Hasanpoor et al. (2025) developed a physics-informed deep convolutional hierarchical encoder-decoder neural network for predicting flow fields within wind farms, incorporating physical constraints to improve generalisation. Anagnostopoulos et al. (2023) employed CNN-based wake models within a multi-fidelity deep transfer learning framework to accelerate wind farm yaw and layout optimisation, demonstrating that features learned from low-fidelity wake simulations could be transferred to improve predictions at higher fidelity levels.

Similarly, GNNs have previously been explored in literature: Park and Park (2019) trained a physics-induced graph neural network (PGNN) for power estimation in wind farms. Bleeg (2020) trained a GNN as an RANS surrogate to predict turbine wind speeds. Ødegaard Bentsen et al. (2022) employed a graph attention network (GAT) to predict individual turbine power production. Duthé et al. (2023, 2024) created a GNN using intrinsic graphs for load estimation of the turbines. Li et al. (2024) demonstrated a graph transformer for power estimation with yaw angle variations. Daenens et al. (2025) implemented a spatio-temporal GNN capable of predicting the power of individual turbines by processing node input signals with a long short-term memory (LSTM) model rather than a conventional multi-layer perceptron (MLP). Wang et al. (2026) demonstrated a graph transformer with a hybrid approach, which they termed a knowledge-fusion graph transformer (KFGT), capable of predicting the turbine-level flow and loads. Hou et al. (2026) demonstrated a two-stage GNN with a novel approach to turbine connectivity using a hybrid approach.

1.1 Our contribution

In our own previous work, we have explored both a CNN-based architecture, the attention residual U-Net (ARU-Net) (Rasmussen et al., 2026) and a GNN-based approach, the graph neural operator (Schøler et al., 2025). In Rasmussen et al. (2026), a two-stage transfer learning strategy was applied, pre-training on the Gaussian engineering model by Zong and Porté-Agel (2020) and fine-tuning on Reynolds-averaged Navier-Stokes actuator wind farm (RANS-AWF) (van der Laan et al., 2023) data. The present work builds on this foundation by extending the ARU-Net architecture with feature-wise linear modulation conditioning, using a variant of parameter-efficient fine-tuning (PEFT), namely low rank adaptation (LoRA), and by directly comparing its performance with that of a graph-based alternative.

In Schøler et al. (2025), we demonstrated a methodology to predict a flow field using an intrinsic graph approach, where each node represented a physical wind turbine. However, to predict at a random location, we introduced probe nodes in addition to the intrinsic wind farm nodes. This probe node formulation shares some similarities with the extrinsic approach, as they can be placed at arbitrary spatial locations rather than at physical entities.

In this work, we present a systematic comparison of the ARU-Net and the GNO for multi-fidelity transfer learning in inter-farm wake modelling. Both architectures are pre-trained on a large dataset generated using the TurbOPark engineering model (Nygaard et al., 2022) implemented in PyWake (Pedersen et al., 2023), then fine-tuned on RANS-AWF data using PyWakeEllipSys (DTU Wind and Energy Systems, 2026) simulations with LoRA-based PEFT. We evaluate wind speed prediction accuracy across diverse wind farm layouts and inflow conditions, analyse spatial performance, including wind speed error distributions, and assess sensitivity to farm size and atmospheric inflow conditions.

The remainder of this paper is organised as follows: Section 2 describes the methodology, including wind farm flow modelling approaches, procedural data generation, neural network architectures, and training procedures. Section 3 presents the results and discussion for both models, covering qualitative and quantitative analyses alongside an assessment of the implica-

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

tions for practical wind farm modelling applications. Section 4 concludes with a summary of the results, an overview of the qualitative differences between the models, and directions for future research.

2 Methodology

In this section, the simulation tools and the neural network architectures are described. However, first, a broader comparison of CNNs and GNNs is presented to clarify their similarities and differences.

2.1 On the connection between CNNs and GNNs

In fields such as computer vision and medical science, CNNs have shown usefulness across many diverse applications. Most users of machine learning (ML) in wind energy have worked with CNNs. Although GNNs are gaining wider adoption within the wind energy community, they remain under-investigated. In this section, an analogy between CNNs and GNNs is presented.

A grid can be viewed as a specific type of graph where the connections of the nodes are fixed on a regular lattice, see, e.g., Bronstein et al. (2021, Sect. 3.4 and 4.2) for a thorough introduction to the concepts and Sanchez-Lengeling et al. (2021) for an interactive introduction. This abstraction carries important implications for the relationship between CNNs and GNNs. Graphs used in Scientific ML (SciML) can be broadly placed on a spectrum between two poles. At one end lie *extrinsic* (discretised) graphs: the nodes and edges form a mesh that approximates an underlying continuous field. The graph is a computational artefact. One could remesh at a different resolution or with a different triangulation and still represent the same physical domain. At the other end lie *intrinsic* (native) graphs, where the nodes correspond to the actual entities of interest and the topology is physically meaningful. A graph whose nodes are wind turbines and whose edges encode relative positional information between them is an example: the topology directly reflects the physical layout of the farm, and there is no finer-resolution version of that layout to recover. CNNs operate exclusively on discretised, regular grids of pixel positions, e.g., on an image or, in the present context, cell centres in the flow field grid.

The motivation for comparing CNNs and GNNs in this work is that the standard discrete convolution used in CNNs can be viewed as a special case of the message-passing operation in GNNs. This is illustrated in Fig. 1, where the physical interpretation of the convolution and message-passing is visualised on a gridded data structure as a raster field (Fig. 1(a)) and as a graph (Fig. 1(b)). For a deeper mathematical analysis, a more detailed overview is provided in Appendix A. As Fig. 1 illustrates, on a regular grid, a convolutional kernel can apply the same learned weights at every position because the uniform cell spacing guarantees that neighbouring cells always bear the same spatial relationship to one another. GNNs achieve analogous local aggregation via edge connections during message passing, without requiring a regular grid.

On a regular pixel grid, a single convolutional layer can only aggregate information from a small spatial neighbourhood defined by the kernel size. For wind farm wake modelling, where upstream turbines are far away from downstream conditions, this local view is insufficient. Deeper architectures address this limitation: in a plain CNN, stacking n_{depth} convolutional layers linearly increase the active domain with depth, while convolutional encoder-decoder architectures, such as the U-Net,

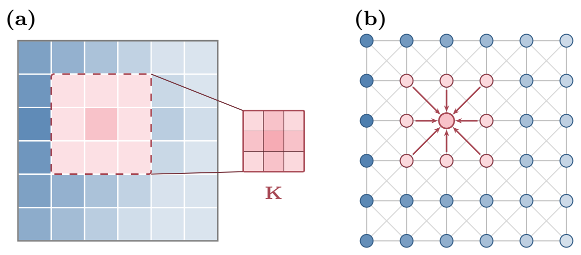

Figure 1. Comparison of local aggregation operations: (a) convolution applies a learned kernel **K** over a regular grid, aggregating values from a fixed spatial neighbourhood; (b) message passing aggregates features from neighbouring nodes on a graph with a fixed grid domain.

go further by compressing the spatial domain using downsampling, allowing deeper layers to see increasingly larger portions of the domain. The methodology is depicted in Fig. 2 (a).

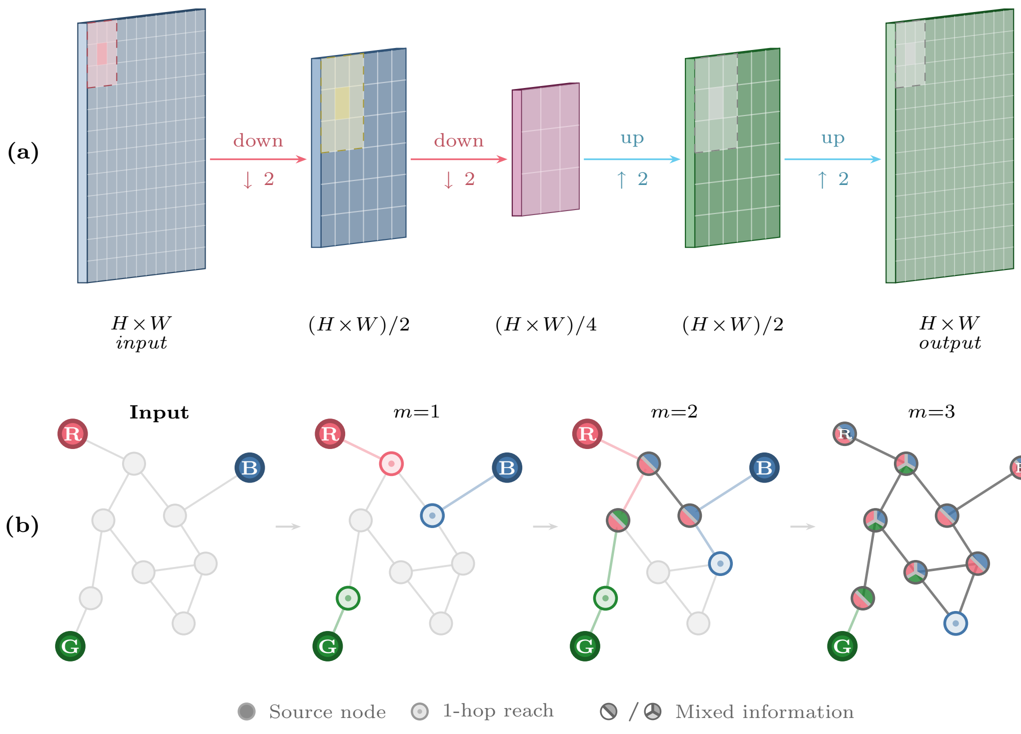

Figure 2. Depth as a mechanism for multi-scale information aggregation. (a) A CNN operating on an input domain of size $H \times W$ growing its effective domain through successive downsampling layers: a local 3×3 kernel covers the full spatial domain after two downsampling stages. (b) A GNN spreads information progressively via message-passing steps (m).

Unlike CNNs, which are inherently defined on regular grids, GNNs operate on arbitrary graph structures, where the choice of intrinsic or extrinsic topology is specified as part of the model architecture prior to training. When the graph is a discretised mesh, similar to grids regularly used for computational fluid dynamics (CFD), as in the MeshGraphNets framework of Pfaff et al. (2021), the message-passing mirrors the spatial discretisation of the computational domain. Stacking message-passing layers then serves a purpose similar to stacking convolutional layers: each additional step propagates information one step further across the mesh, gradually extending the spatial reach of the model. Two examples relevant to wind energy flow modelling include Li et al. (2022), who created a single turbine RANS-based GNN wake surrogate model using a discredited mesh, and Duthé et al. (2025), who demonstrated a mesh-type GNN that is used to create realistic realisations of the flow around an airfoil given knowledge of the airfoil boundary.

In contrast, when the graph is intrinsic, as in this work, where each node represents a wind turbine and the edges encode positional relations, a single message-passing step already exchanges information directly between the turbines. Stacking additional layers, therefore, takes on a different role. Rather than extending the spatial reach in a grid, it enables the model to consider turbine interactions outside the direct neighbourhood of each local turbine, thereby connecting the turbines into a farm. A visualisation of GNN message passing in connection with depth is illustrated in Fig. 2 (b), where three source nodes are not directly connected, but information is exchanged between them after successive message passing steps (m).

2.2 Wind farm engineering model

To generate the low-fidelity dataset, we use the open-source tool PyWake (Pedersen et al., 2023). PyWake is a modular implementation of various engineering modelling tools, including wake deficit models, added TI models, blockage, wake deflection, and mirroring. We employ the turbulence optimised park model (TurbOPark) with the Gaussian wake profile developed by Nygaard et al. (2022). TurbOPark was selected because it was calibrated for far-wake and cluster-wake applications and showed the closest agreement with RANS-AWF among the engineering models we tested.

The model incorporates the added TI formulation of Frandsen (2007) directly into the wake expansion, eliminating the need for a standalone TI model. For combining multiple wakes, it uses squared-sum superposition, motivated by energy-deficit arguments (Porté-Agel et al., 2020), a ground-mirroring model to implement the surface boundary condition, and a Gaussian overlap model for rotor averaging. The wake deficit model is given by

$$\frac{\Delta u}{U_\infty} = C \exp\left(-\frac{r^2}{2\sigma_w^2}\right), \quad C = 1 - \sqrt{1 - \frac{C_\mathrm{T}}{8(\sigma_w/D)^2}}, \tag{1a}$$

$$\frac{\sigma_w}{D} = \varepsilon + \frac{0.04\, I_0}{\beta}\left(\sqrt{(\alpha+\beta\xi)^2+1} - \sqrt{1+\alpha^2} - \ln\left[\frac{\left(\sqrt{(\alpha+\beta\xi)^2+1}+1\right)\alpha}{\left(\sqrt{1+\alpha^2}+1\right)(\alpha+\beta\xi)}\right]\right), \tag{1b}$$

$$\alpha = 1.5\, I_0, \quad \beta = \frac{0.8\, I_0}{\sqrt{C_\mathrm{T}}}, \quad \varepsilon = 0.25\sqrt{\frac{1+\sqrt{1-\widehat{C}_\mathrm{T}}}{2\sqrt{1-\widehat{C}_\mathrm{T}}}}, \quad \widehat{C}_\mathrm{T} = \min(C_\mathrm{T}, 0.96), \tag{1c}$$

where Δu is the velocity deficit, U_∞ the free-stream velocity, r the radial distance from the wake centreline, σ_w the characteristic wake width, D the rotor diameter, $\xi = x/D$ the dimensionless downstream distance, C_T the thrust coefficient, and I_0

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

the ambient TI. The coefficient C represents the centreline maximum velocity deficit. The parameters α and β, which govern the turbulence-dependent wake expansion rate, are adopted from Frandsen (2007); the coefficient 0.04 was calibrated by Nygaard et al. (2022) against operational data from 19 offshore wind farms. The initial wake width ε follows the formulation of Bastankhah and Porté-Agel (2014).

Despite its calibration including far-wake physics, TurbOPark shares limitations common to engineering wake models: it assumes steady, axisymmetric wakes, neglects large-scale atmospheric effects, and relies on algebraic superposition, which does not fully capture wake interactions.

2.3 Reynolds averaged Navier–Stokes with actuator wind farm (RANS-AWF)

RANS-AWF simulations offer higher physical fidelity than engineering models by resolving the mean flow field around wind farms, including pressure gradients and wake boundary layer interactions, while representing turbines through distributed momentum sink terms.

The high-fidelity dataset is produced with EllipSys3D (Michelsen, 1992; Sørensen, 1995), coupled to the AWF model of van der Laan et al. (2023). In this formulation, the RANS equations are closed by the k-ε-f_P turbulence closure (van der Laan et al., 2015), and each wind farm enters the momentum equations as a distributed drag force analogous to a forest canopy. A surrogate trained on higher-resolution actuator-disk simulations (van der Laan et al., 2026) provides the wind farm thrust coefficient, enabling accurate long distance wake prediction at moderate computational cost. A single RANS-AWF simulation takes only a few minutes using multiple nodes on a high performance computing (HPC) cluster (Technical University of Denmark, 2019).

A neutral atmospheric surface layer is employed by prescribing a logarithmic streamwise velocity profile, a constant turbulence kinetic energy (TKE) and a profile of the TKE dissipation, at both the inlet and top boundaries. Inflow TI is governed by the surface roughness length z_0, which we relate to a prescribed streamwise I_0 via

$$z_0 = z_H \exp\left(-\frac{1}{I_0}\right). \tag{2}$$

Because of the k-ε-f_P turbulence closure is formulated in terms of a turbulence kinetic energy (TKE)-based turbulence intensity $I_{\mathrm{TKE}} = \sqrt{2k/3}\,/\,U_\infty$ rather than the streamwise component, a conversion is needed to map to the ambient TI, I_0. In the atmospheric surface layer, turbulence is anisotropic, with streamwise fluctuations exceeding those in the vertical and lateral directions (van der Laan et al., 2015). We therefore adopt $I_{\mathrm{TKE}} \approx 0.8\,I_0$. The sampled range for the RANS-AWF dataset is $I_0 \in [0.03, 0.25]$ corresponds to roughness lengths $z_0 \in [\sim 10^{-12}, \sim 2]$ m.

The numerical domain represents a Cartesian grid with dimensions $1522D \times 1266D \times 25D$ in the streamwise, x, lateral, y, and vertical, z, directions, respectively. The wind farm is placed at the horizontal origin of the domain, $\{x, y\} = \{0, 0\}$, which is located $569D$ from the inlet boundary and $633D$ from the lateral boundaries. A region around the wind farm, with dimensions of $-69 < x/D < 453$ and $-133 < y/D < 133$, is defined at which a cell spacing of $2D$ is applied, based on a grid refinement study (van der Laan et al., 2023). The horizontal cell sizes are stretched towards the lateral, inlet, and outlet boundaries with a maximum growth ratio of 1.2, covering a distance of $500D$. The vertical cell distribution consists of a first cell height of $D/200$ that is initially stretched towards a cell height of $D/6$ at $z/D = 3$, and then further stretched towards the top boundary. In total, 3.9 million cells are used, distributed as $320 \times 192 \times 64$ in the horizontal, lateral, and vertical directions, respectively. The logarithmic inflow is prescribed at the inlet and top boundary, resulting in a lid-driven flow setup. A fully developed flow is assumed at the outlet, at which all gradients normal to the outlet plane are assumed to be zero. The bottom boundary is a rough wall boundary (Sørensen et al., 2007) and the lateral boundaries are periodic. The hub-height plane at 119 m is extracted from each simulation and interpolated onto a 256×128 Cartesian grid consistent with the low-fidelity domain, with 64 vertical levels. The resulting dataset comprises 1000 samples from 250 distinct layouts, each evaluated at four uniformly sampled inflow conditions, and is partitioned into 700 for training, 100 for validation and 200 for test cases.

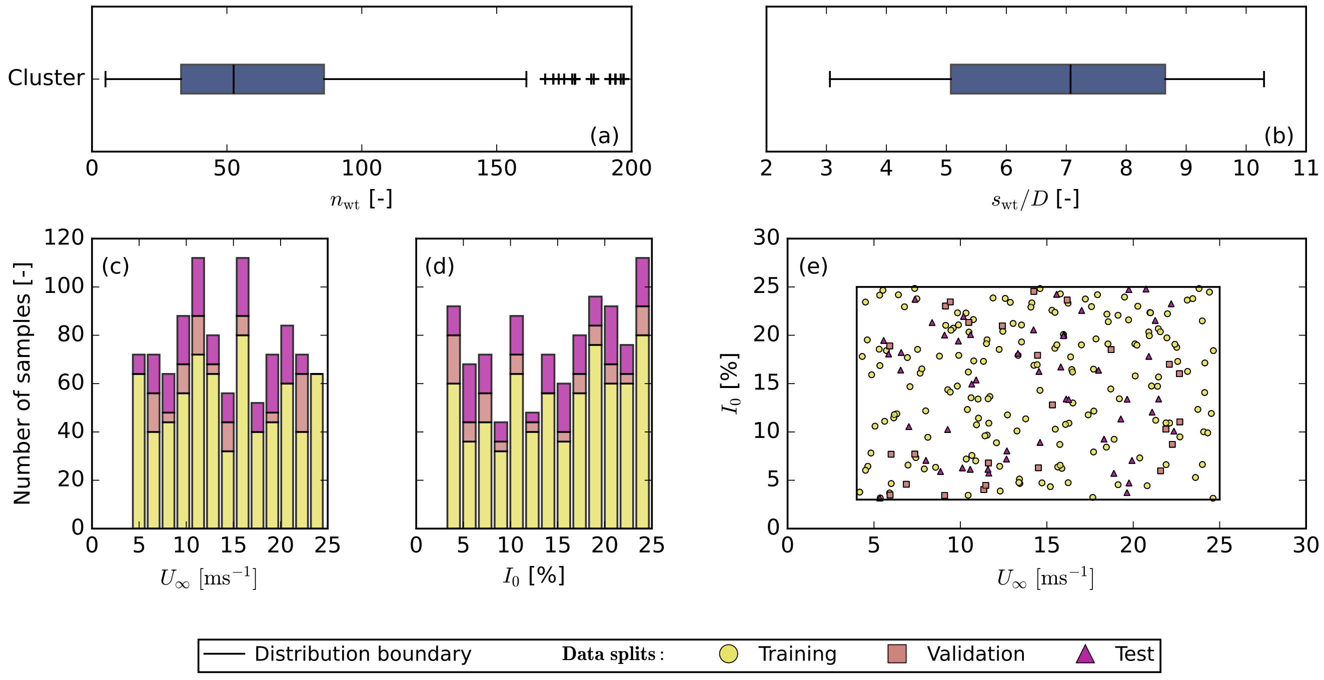

Figure 3. Procedurally generated data for the RANS-AWF dataset, (a–b) statistics of generated layouts, (c) U distribution, (d) Ambient TI distribution, and (e) generated U and I_0 samples with boundary.

2.4 Procedural data generation

To ensure the models are trained on a broad range of layouts and inflow conditions, a procedural data-generation strategy is adopted for the engineering model dataset. The method is adopted from Schøler et al. (2025), but with two improvements: (i) layout-specific number of wind turbines (n_{wt}) and turbine separation factors (s_{wt}) defining the smallest allowed distance between turbines, and (ii) capping I_0 at 80% to limit the number of extreme cases. An overview of the method as applied when generating the low-fidelity dataset, is shown in Fig. 4.

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

The RANS-AWF dataset is repurposed from Rasmussen et al. (2026) and therefore does not utilise the same sampling strategy. In Rasmussen et al. (2026), wind farm layouts were generated by placing turbines within random polygons using Poisson disk sampling with minimum spacing constraints between $3D$ and $10D$, similar to the cluster generation mode in PLayGen. Random turbine removal, with a probability of 0%–50%, was applied to increase layout diversity. Inflow conditions were sampled uniformly with $U_\infty \in [4, 25]$ m s^{-1} and $I_0 \in [0.03, 0.25]$, treating wind speed and turbulence intensity as independent variables. This contrasts with the engineering model dataset, in which U and σ_u are coupled through the correlated IEC-based sampling framework. The RANS-AWF sampling domain can largely be considered a subset of the engineering model inflow range. However, because the IEC-based upper bound on σ_u (Eq. 3b) decays with increasing wind speed, the correlated Sobol sequence does not populate the corner of the (U_∞, I_0) space where both quantities are simultaneously high. A small number of uniformly sampled RANS-AWF points occupy this region and therefore fall outside the engineering model's inflow coverage.

Wind farm layout generation

To generate the wind farm layouts, the Plant Layout Generator (PLayGen) by Harrison-Atlas et al. (2024) is used. PLayGen generates four types of layouts. *Cluster*, which consists of irregularly distributed turbines across the domain. *Single string* layouts, that arrange turbines along a single curved string with periodic gaps. *Parallel string* layouts, which feature multiple strings sharing an orientation, similar to conventional offshore wind farms. *Multiple string* layouts, that comprise several strings with independent orientations, resulting in configurations with strings at multiple angles. Examples are shown in Fig. 4 (a–d).

Inflow sampling

For each layout we consider four different combinations of inflow conditions, meaning four combinations of the free-stream velocity U and ambient turbulence intensity I_0. These parameters are coupled in the atmospheric boundary layer, so we employ the correlated sampling framework of Dimitrov et al. (2018) to generate physically realistic combinations. Following updates to IEC 61400-1 (2019 ed. 4), we adopt the A$^+$ turbulence class with reference intensity $I_{\mathrm{ref,A^+}} = 0.18$. An upper bound of $I_{0,max} = 0.8$ was set to avoid overly extreme flow cases.

The operational velocity range is set by the DTU-10-MW reference turbine (Bak et al., 2013), which has a rotor diameter of $D = 178.3\,\mathrm{m}$, cut-in wind speed $U_{\mathrm{c,in}} = 4\,\mathrm{m\,s^{-1}}$, rated wind speed $U_{\mathrm{rated}} = 11.4\,\mathrm{m\,s^{-1}}$, and cut-out wind speed $U_{\mathrm{c,out}} = 25\,\mathrm{m\,s^{-1}}$. Bounds on the velocity standard deviation σ_u follow Dimitrov et al. (2018, Tab. 1):

$$4\,\mathrm{m\,s^{-1}} \leq U_\infty \leq 25\,\mathrm{m\,s^{-1}}, \tag{3a}$$

$$0.0025\,U_\infty \leq \sigma_u \leq I_{\mathrm{ref,A^+}} \left(6.8 + \frac{3}{4}U_\infty + 3\left(\frac{10}{U_\infty}\right)^2\right)\ [\mathrm{m\,s^{-1}}]. \tag{3b}$$

We populate the inflow space using quasi-Monte-Carlo sampling with the improved Sobol sequence of Joe and Kuo (2008). Although Dimitrov et al. (2018) used the Halton sequence, we found that the Sobol sequence achieves better coverage of the distribution tails (Schøler et al., 2025). The velocity samples are mapped through a Rayleigh distribution following IEC 61400-

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

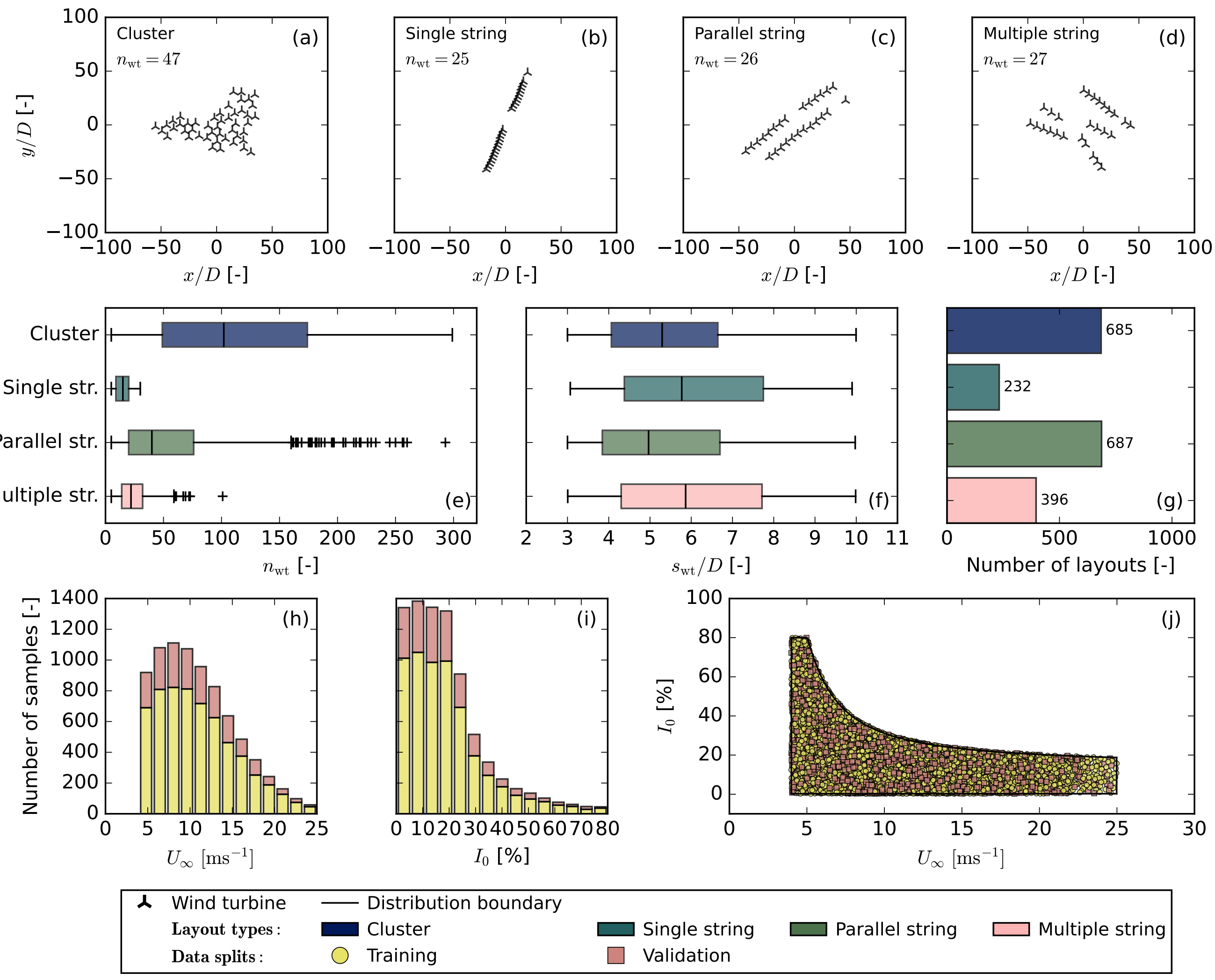

Figure 4. Procedurally generated data for the engineering model dataset, (a–d) examples of wind farm layout types in PLayGen, (e-g) statistics of generated layouts, (h) U distribution, (i) Ambient TI distribution, and (j) generated U and I_0 samples with boundary.

1 (2019 ed. 4) and scaled to the bounds in Eq. (3a). The turbulence samples are transformed via Eq. (3b) to obtain σ_u, which is then converted to ambient TI through $I_0 = \sigma_u/U_\infty$. The resulting TI distribution is shown in Fig. 4 (j).

2.4.1 Dataset Configuration

Table 1 summarises the dataset configuration for both fidelity levels. Both architectures are trained on the same layouts and inflow conditions to permit a fair comparison. However, the data representations differ. The ARU-Net operates on gridded flow fields of shape $256 \times 128 \times 1$, whereas the GNO represents each layout as a graph whose turbine nodes carry positional and inflow features and whose probe nodes are placed at evaluation locations. The two models are evaluated on two distinct grids, the CNN grid (256×128) and the GNN grid (257×129), to assess the sensitivity of each model to the grid changes (Sect. 3.4).

Table 1. Dataset configuration for low-fidelity and high-fidelity simulations.

Parameter	Low-Fidelity (TurbOPark)	High-Fidelity (RANS-AWF)
Wind speed (U_∞)	4–25 $\mathrm{m\,s^{-1}}$ (Rayleigh/Sobol)	4–25 $\mathrm{m\,s^{-1}}$ (uniform)
Turbulence intensity (I_0)	0.25–80% (Dimitrov et al., 2018)	3–25% (uniform)
Wind direction	270° (fixed)	270° (fixed)
Conditions per layout	4	4
Unique layouts	2 000	250
Total samples	8 000	1 000
Train / Val / Test split	75% / 25% / –	70% / 10% / 20%

265 The low-fidelity dataset comprises 8 000 TurbOPark simulations generated from 2 000 unique layouts, each evaluated at four inflow conditions sampled using the correlated IEC-based framework of Dimitrov et al. (2018), as described in Sect. 2.4. The high-fidelity dataset consists of 1 000 RANS-AWF simulations derived from 250 unique layouts, each evaluated at four uniformly sampled inflow conditions with $U_\infty \in [4, 25]\ \mathrm{m\,s^{-1}}$ and $I_0 \in [0.03, 0.25]$, partitioned into 700 training, 100 validation, and 200 test samples, an overview of the RANS-AWF dataset is presented in Fig. 3. In both datasets, layouts are assigned to a
270 single data split before inflow sampling, ensuring that no identical layout appears in both the training and evaluation sets. A key difference between the two datasets is the distribution of inflows. The low-fidelity data spans a substantially wider turbulence intensity range (I_0 up to 80%, Fig. 4(i-j)) using the correlated sampling of Dimitrov et al. (2018), whereas the RANS-AWF data is restricted to $I_0 \in [0.03, 0.25]$ (see Fig. 3(e)). Figure 5 shows the domain in which the data is generated. For data gener-

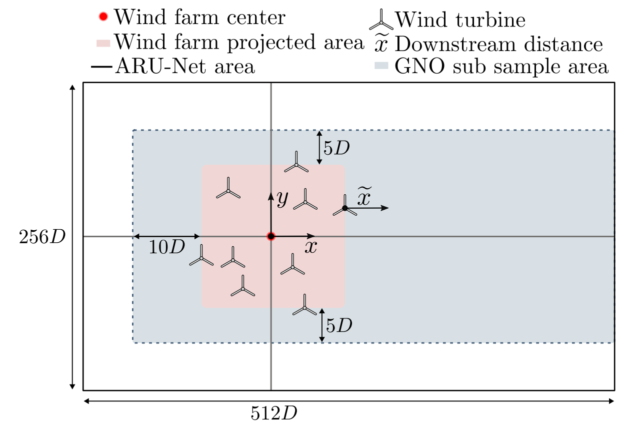

Figure 5. Modelling domain shared by the U-Net and GNO, indicating the spatial subset of probe locations used when training the GNO.

ation, the same fixed-resolution domain for all layouts, as required by the CNN-based approach. For the GNO, no fixed grid is necessary, owing to its grid-invariance, only a spatial subset of the domain is queried during training to improve computational efficiency. This concept is shown in Fig. 5.

2.5 Attention residual U-Net (ARU-Net)

The attention residual U-Net (ARU-Net) is a convolutional surrogate model that predicts wind farm wake fields from turbine layout rasters and inflow specifications. Its architecture follows the encoder-decoder paradigm established by U-Net (Ronneberger et al., 2015), with several enhancements to improve feature extraction and gradient flow. The encoder extracts hierarchical spatial features through successive downsampling stages. The bottleneck integrates inflow conditions, and the decoder reconstructs the flow field at the original resolution via skip connections that preserve fine-grained spatial information. A similar architecture was previously used in a multi-fidelity transfer learning context in Rasmussen et al. (2026). The remainder of this section describes the configuration used in the present work.

The encoder comprises six resolution levels connected by five downsampling stages, with channel counts increasing at each stage and a fixed maximum channel count. Each encoder block contains two 3×3 convolutional layers with residual connections,

$$\boldsymbol{h}^{(l)} = \psi_{\mathrm{m}}\left(\mathcal{F}(\boldsymbol{h}^{(l-1)}; \boldsymbol{\theta}^{(l)}) + \mathcal{P}(\boldsymbol{h}^{(l-1)})\right), \tag{4}$$

where $\mathcal{F}$ denotes the stacked convolution–normalisation transformation with learnable parameters $\boldsymbol{\theta}^{(l)}$, $\mathcal{P}$ is a 1×1 projection that matches channel dimensions when necessary, and $\psi_{\mathrm{m}} = \mathrm{Mish}$ (Misra, 2020) is the activation function. Group normalisation is applied after each convolution. Downsampling employs a learned Gaussian blur filter to avoid spatial aliasing artefacts, while the decoder uses transposed convolutions for upsampling. Squeeze-and-excitation (SE) blocks (Hu et al., 2018) provide channel-wise feature recalibration within each encoder and decoder block, as seen in Eq. (5):

$$\boldsymbol{z} = \frac{1}{HW}\sum_{i,j} \boldsymbol{h}_{i,j}, \tag{5a}$$

$$\boldsymbol{s} = \psi_{\mathrm{s}}(\mathbf{W}_2\,\psi_{\mathrm{r}}(\mathbf{W}_1 \boldsymbol{z})), \tag{5b}$$

$$\tilde{\boldsymbol{h}} = \boldsymbol{s} \odot \boldsymbol{h}, \tag{5c}$$

where $\boldsymbol{z}$ is the spatially-pooled feature vector, $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable weights with a reduction factor of 16, $\psi_{\mathrm{r}} = \mathrm{ReLU}$ (Agarap, 2019), ψ_{s} is the sigmoid function, and $\odot$ denotes the index-wise product. Attention gates (Oktay et al., 2018) are applied to skip connections to suppress irrelevant features before concatenation with decoder activations, allowing the network to focus on spatially relevant wake structures, denoted by AG in Fig. 6.

Conditioning on inflow variables $\boldsymbol{\Omega} \in \{U_\infty, I_0\}$ is achieved through two mechanisms. At the spatial bottleneck, the scalar inputs are projected to match the bottleneck channel dimension, broadcast across the spatial dimensions, and added to the feature map. This is followed by channel mixing using an expansion, followed by a compression pathway, such that the channels are expanded by $c \rightarrow 2c \rightarrow c$, with a residual connection and depthwise-separable spatial refinement. Additionally, feature-wise linear modulation (FiLM) layers modulate intermediate feature maps throughout the encoder (from the second

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

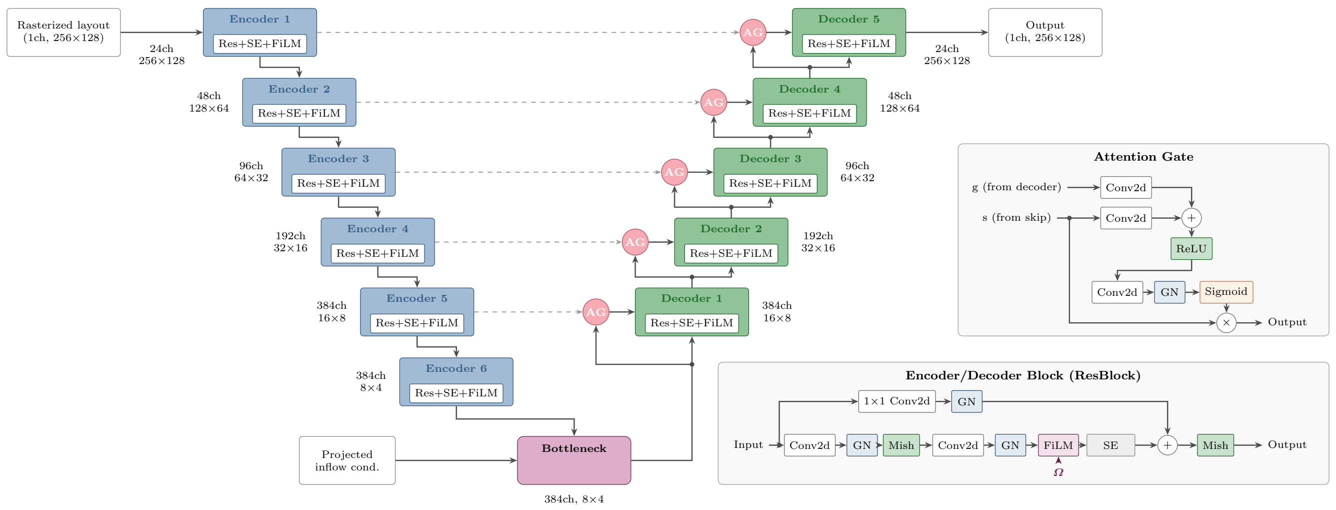

Figure 6. Overview of the attention residual U-Net (ARU-Net) architecture. The model follows an encoder-decoder structure with skip connections, incorporating squeeze-and-excitation (SE) attention in each block and attention gates (AG) on skip connections. Inflow conditions are injected at the spatial bottleneck and through feature-wise linear modulation (FiLM) layers.

block onward) and all decoder blocks:

$$\mathrm{FiLM}(\boldsymbol{h},\,\boldsymbol{\Omega}) = \boldsymbol{h}\odot\big(1+\gamma(\boldsymbol{\Omega})\big)+\beta(\boldsymbol{\Omega}), \tag{6}$$

where γ and β are learned scale and MLP scalers for the inflow conditions, with outputs clamped to ± 0.3 and ± 0.1 respectively to ensure stable training. Regularisation is applied through dropout (10% in the bottleneck and decoder) and stochastic depth with a linear schedule that increases from 0% at the shallowest encoder block to a peak, determined during hyperparameter tuning, at the deepest encoder blocks, and is mirrored symmetrically in the decoder.

To select the ARU-Net architecture and training hyperparameters, a Bayesian hyperparameter optimisation was conducted on the TurbOPark engineering dataset using the Optuna framework (Akiba et al., 2019) with the tree-structured Parzen estimator (TPE) sampler. A total of 200 trials were evaluated, each trained for a fixed budget of 50 epochs. The final selected model was subsequently trained to convergence on the full TurbOPark dataset. The search space for the hyperparameter tuning can be seen in Tab. 2. Weight decay of 0.01 applied through the AdamW optimiser, and gradient clipping with a maximum norm of 1.0.

For the ARU-Net, the inflow conditioning variables are scaled to the unit interval via min-max normalisation: $U_\infty \in [4.0, 25.0]\ \mathrm{m\,s^{-1}}$ and $I_0 \in [0.03, 0.25]$ are each mapped to $[0, 1]$. The target flow field undergoes a log-deficit transformation. Because RANS-AWF simulations can produce local speed-ups from blockage effects, cell velocities exceeding U_∞ are first clamped to U_∞. The relative wake deficit, $\delta = 1 - u/U_\infty$, then quantifies the fractional velocity reduction at each grid cell. Because the logarithm is singular at zero, a floor value $\varepsilon_0 = 10^{-5}$ is applied, giving the clamped deficit $\delta_c = \max(\delta, \varepsilon_0)$. The normalised target $\tilde{\delta} \in [0, 1]$ is then obtained as

$$\tilde{\delta} = \frac{\ln \delta_c - \ln \varepsilon_0}{-\ln \varepsilon_0}. \tag{7}$$

325 This log transformation compresses the wide dynamic range of wake deficits, improving the network's ability to resolve both strong and near-wake losses and subtle far-wake recovery.

Table 2. Hyperparameter search space for the ARU-Net architecture (15 dimensions). The search was conducted using the Optuna framework. Categorical choices are denoted by braces $\{\ldots\}$ and continuous ranges by brackets $[\ldots]$.

Parameter	Search Space
Architecture	
Initial channels	$\{16, 24, 32, 48\}$
Encoder depth	$\{3, 4, 5\}$
Channel multiplier	$[1.3, 2.5]$
Max channels	$\{256, 384, 512\}$
Kernel size	$\{3, 5\}$
Convolutions per block	$\{1, 2, 3\}$
Flags (SE, Attn, L.Down, L.Up)	{on, off}
Regularization	
Dropout rate	$[0.0, 0.25]$
Stochastic depth (p_{sd})	$[0.0, 0.15]$
Optimization	
Learning Rate (log)	$[10^{-5}, 10^{-3}]$
Weight decay (log)	$[10^{-4}, 10^{-1}]$
Gradient clip norm	$[0.5, 2.0]$

2.6 Graph neural operator (GNO)

The GNN approach uses the GNO model, first introduced in Schøler et al. (2025). The GNO is a graph-based surrogate model that enables the prediction of flow fields. An overview of the model is presented in Fig. 7, the architecture is organised into 330 three stages: an encoder ($\mathcal{E}$) that projects turbine features and their spatial relationships into a latent space, an approximator ($\mathcal{A}$) that iteratively refines turbine states through learned message passing, and a decoder ($\mathcal{D}$) that evaluates the resulting flow at arbitrary locations.

The architecture has one addition: a global context that supplies inflow conditions to the decoder, as illustrated in Fig. 7, following the path of $\boldsymbol{\Omega}$. Rather than concatenating the raw inflow values $\boldsymbol{\Omega} \in \{U, I_0\}$ directly to the process node, which

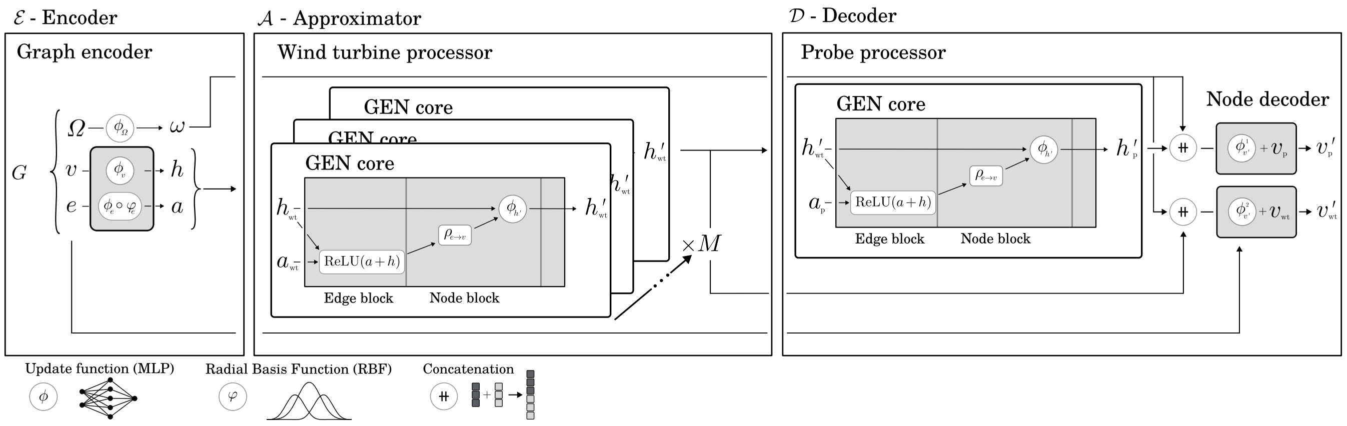

Figure 7. Overview of the GNO model with the added global context. The three main components remain: Encoder ($\mathcal{E}$), Approximator ($\mathcal{A}$), and Decoder ($\mathcal{D}$).

would mix observational and latent spaces, a small encoder ϕ_Ω first projects $\boldsymbol{\Omega}$ into a latent representation $\boldsymbol{\omega}$ with the same dimensionality as the latent space node features. Because its sole purpose is to bridge this dimensional mismatch, ϕ_Ω is implemented as a single-hidden-layer neural network,

$$\boldsymbol{h}_\Omega^{(1)} = \psi_\mathrm{r}\left(\mathbf{W}^{(0)\top}\boldsymbol{\Omega} + \boldsymbol{b}^{(0)}\right), \tag{8a}$$

$$\phi_\Omega(\boldsymbol{\Omega}) = \boldsymbol{\omega} = \mathbf{W}^{(1)\top}\boldsymbol{h}_\Omega^{(1)} + \boldsymbol{b}^{(1)}, \tag{8b}$$

where $\boldsymbol{h}_\Omega^{(1)}$ is the hidden-layer activation, and $\mathbf{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$ are the learnable weights and biases of layer l. The resulting $\boldsymbol{\omega} = \phi_\Omega(\boldsymbol{\Omega})$ is concatenated to each process node prior to decoding. The remaining hyperparameter names are adopted directly Schøler et al. (2025), with the latent space dimension Q, hidden layer size of the internal MLP (q_int) and hidden layers (L_int), the number of message passing steps in the wind turbine processor (M_wt) as seen in Fig. 7, and hidden layer size of the decoder (q_dec) and the decoder hidden layers (L_dec). To ensure that changes in both the data type and the model architecture are adequately addressed, a smaller grid search has been conducted with variations of these hyperparameters.

The GNO uses the normalisation and regularisation techniques from Schøler et al. (2025), normalising inputs and targets to the min-max range, applying dropout with a rate of 10%, and using layer normalisation. In addition, during transfer learning, gradient clipping normalisation has been added as an optional addition, with a max l^2-norm of 1.0.

In Fig. 8 an overview of the data flow is provided, demonstrating how the model-stages processes the graph and a prediction is made. This design decouples the turbine interaction model from the spatial prediction, allowing new locations to be evaluated without recomputing the upstream turbine graph. A full description of the base architecture is given in Schøler et al. (2025). The remainder of this section describes an improvement introduced in the present work.

To distinguish between the ARU-Net and the GNO, it is at times convenient to refer to the GNO with the operator notation, $\mathcal{G}$. All considered model variations are based on the best original model, designated as $\mathcal{G}_\mathrm{org}$. In addition, three smaller and three larger models have been considered. These variations are reported here in order of size: $\mathcal{G}_\mathrm{S3}$, $\mathcal{G}_\mathrm{S2}$, $\mathcal{G}_\mathrm{S1}$, $\mathcal{G}_\mathrm{org}$, $\mathcal{G}_\mathrm{L1}$, $\mathcal{G}_\mathrm{L2}$, and

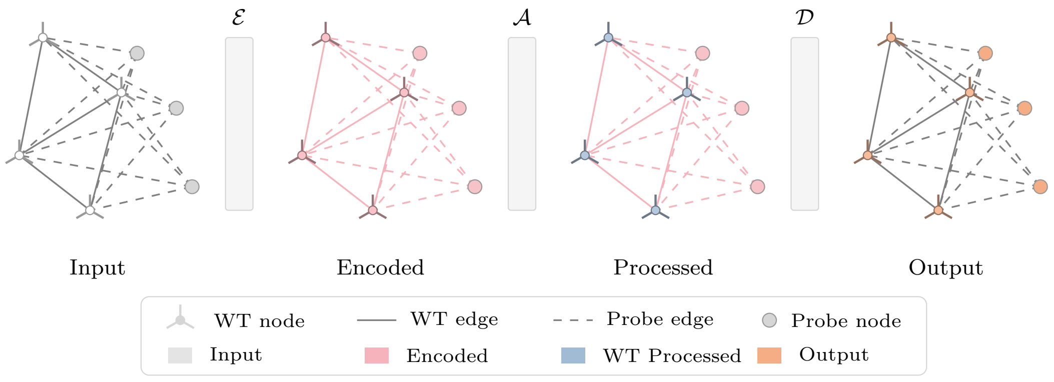

Figure 8. GNO model input and output augmentation. The model stages, encoder $\mathcal{E}$, approximator $\mathcal{A}$, and decoder $\mathcal{D}$, are displayed as simple blocks; see Fig. 7 for the expanded model stages.

$\mathcal{G}_{L3}$. The actual model configurations are reported in Tab. 3. In addition, for the step-based learning rate schedule, two initial learning rates, 5×10^{-3} and 1×10^{-3}, have been considered, each divided by 10 at 200 and 350 epochs, respectively.

2.7 Training and validation

For training the respective models, different stochastic gradient-based optimisers have been used: the GNO has been trained with Adam (Kingma and Ba, 2014) using default parameters, while the ARU-Net has been trained using AdamW (Loshchilov and Hutter, 2019), which decouples weight decay from the adaptive learning rate by applying it directly to the parameters, ensuring that regularisation is applied uniformly regardless of gradient history. For the GNO, training is monitored using the validation set, which is evaluated every 5 epochs. If an improvement of less than 1×10^{-7} is observed over 1000 consecutive epochs, training is stopped. During transfer learning to the RANS-AWF dataset, the limit is lowered to 200 epochs. The ARU-Net and GNO use slightly different training stacks. The ARU-Net is built on PyTorch (Paszke et al., 2019), while the GNO is built on a JAX (Bradbury et al., 2018) based framework, using Jraph (Godwin et al., 2020) for GNN abstractions, Flax (Heek et al., 2024) for neural network layers, and PyTorch Geometric (Fey and Lenssen, 2019) for graph construction and data loading. All training was done on a system featuring two AMD EPYC 7351 16-core CPUs (32 cores, 2.4 GHz), 256 GB of DDR4 RAM (2.666 GHz), and an 8 GB NVIDIA Quadro RTX 4000 GPU (Technical University of Denmark, 2019).

Performance metrics and loss function

During training and validation, two metrics are considered: the l^1-norm-based mean absolute error (MAE) and the l^2-norm-based mean squared error (MSE). MSE is used as the loss function ($\mathcal{L}$) for both the ARU-Net and the GNO. In addition, during testing, the root mean squared error (RMSE) and an F_1-score are also reported. MAE and MSE are used to compare models during development, while RMSE is chosen for final evaluation as it is more readily interpretable than MSE. While most of these metrics are standard for flow surrogates, the F_1-score is less common. The F_1-score is typically used in classification

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

tasks, where it quantifies performance by comparing true positives (t_p), false positives (f_p), and false negatives (f_n). Here, the F_1-score is used to evaluate how well the wake boundary is captured. A binary mask is created using a threshold (δ_w): grid points falling below the threshold are classified as unwaked and those above as waked. Equation 9 shows the definitions of the metrics applied in this paper:

$$\mathrm{MAE} = \frac{1}{N}\left\|\boldsymbol{u}-\boldsymbol{u}'\right\|_1, \tag{9a}$$

$$\mathcal{L} = \mathrm{MSE} = \frac{1}{N}\left\|\boldsymbol{u}-\boldsymbol{u}'\right\|_2^2, \quad \mathrm{RMSE} = \frac{1}{\sqrt{N}}\left\|\boldsymbol{u}-\boldsymbol{u}'\right\|_2, \tag{9b}$$

$$F_1 = \frac{2\cdot\mathrm{precision}\cdot\mathrm{recall}}{\mathrm{precision}+\mathrm{recall}} = \frac{2\,t_\mathrm{p}}{2\,t_\mathrm{p}+f_\mathrm{p}+f_\mathrm{n}}, \tag{9c}$$

where $\boldsymbol{u}$ are the target velocities, $\boldsymbol{u}'$ are the predicted velocities, and N is the number of observations. The F_1-score, defined as the harmonic mean of precision and recall, is shown in two equivalent forms: one using precision and recall, and one using the classification outcomes t_p, f_p, and f_n, to emphasise its interpretation as an overall classification accuracy metric.

Multi-fidelity transfer learning

A range of different strategies has been tested to perform the transfer learning. The most straightforward approach is to update all weights during transfer learning; however, in some cases, this has been reported to be less efficient than freezing parts of the network during the transfer learning stage (Hu et al., 2021). Updating all parameters during transfer learning is termed fine-tuning, whereas freezing parts of the network is collectively referred to as weight freezing. Finally, the last considered approach is a variation of parameter-efficient fine-tuning (PEFT) called low-rank adaptation (LoRA) by Hu et al. (2021), which is referred to as LoRA fine-tuning. During fine-tuning with LoRA, the pre-trained weights $\mathbf{W}_0^{(l)} \in \mathbb{R}^{d\times p}$ are frozen across the entire network. LoRA adapters are introduced in all layers of the targeted MLPs, while the remaining MLPs are kept fully frozen. For each adapted layer, the forward pass becomes

$$\boldsymbol{\xi}^{(l)} = \left(\mathbf{W}_0^{(l)} + \Delta\mathbf{W}^{(l)}\right)\boldsymbol{\xi}^{(l-1)}, \tag{10a}$$

$$\Delta\mathbf{W}^{(l)} = \mathbf{B}^{(l)}\mathbf{A}^{(l)}, \tag{10b}$$

$$\mathbf{A}_0^{(l)} \sim \mathcal{N}(0, 1/r), \quad \mathbf{B}_0^{(l)} = \mathbf{0}, \tag{10c}$$

where $\mathbf{A}^{(l)} \in \mathbb{R}^{r\times d}$ and $\mathbf{B}^{(l)} \in \mathbb{R}^{p\times r}$, with a pre-determined rank r that controls the degree of parameter reduction. The initialisation in Eq. (10c) ensures $\Delta\mathbf{W}^{(l)} = \mathbf{0}$ at the start of fine-tuning, so the model initially reproduces the pre-trained predictions exactly.

2.8 Testing and validation

To test the performance of the neural networks, a separate test set is used to avoid data leakage. While it is a moderate dataset of 200 inflow-layout combinations, it is still more than what is practical for creating qualitative plots, such as streamwise velocity deficit profiles and flow field plots. To ensure a representative sub-sample, the test set is grouped by layout size into three categories based on the number of turbines. Within each size category, three inflow conditions are selected targeting the diagonal of the (U_∞, I_0) space. The low-, medium-, and high-regions are defined by dividing the wind speed and ambient TI ranges into three equal bins across all test cases. For each size-inflow combination, the case closest to the centroid of each category in the (U_∞, I_0) space is selected, yielding nine representative test cases in total. The three size categories are split into the following sizes: n_{wt}: small (S, $n_{\mathrm{wt}} \leq 47$), medium (M, $47 < n_{\mathrm{wt}} < 110$), and large (L, $n_{\mathrm{wt}} \geq 110$), where the S/M boundary is the median of sub-large layouts and the M/L boundary is set at 110 turbines. These thresholds yield 21 small, 21 medium, and 8 large layouts. The selection of these regions is shown visually in Fig. 9.

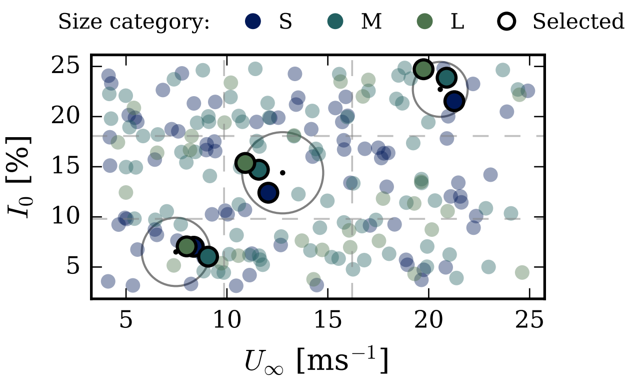

Figure 9. Selection of the RANS-AWF test subset for qualitative evaluation. Background scatter points represent the layouts in the test set, grouped by turbine count n_{wt} into S, M, and L. The parameter space of (U_∞, I_0) is divided into low, medium, and high regions along the diagonal. For each size category, the case closest to the centroid of each (U_∞, I_0) region is selected. The centroids of each background region are marked with a black dot, surrounded by a circle encompassing the three size-selected test cases.

For the transfer learning approach to be feasible, it is a prerequisite that the source and target domains share sufficient similarities for the learned representations to generalise, while the high-fidelity data remains sufficiently different that training on both provides meaningful improvements in accuracy, justifying the additional cost. Figure 10 examines that balance for the two fidelity levels used in this work, with Fig. 10 (a)–(d) displaying the flow case from the RANS-AWF test dataset with the largest overall RMSE between TurbOPark and RANS-AWF ($n_{\mathrm{wt}} = 185$, $U_\infty = 5.90\,\mathrm{m\,s^{-1}}$, $I_0 = 3.69\%$).

Figure 10 (a) shows the cross-stream velocity profiles, meaning the relative streamwise velocity deficit $\Delta u / U_\infty$ across the lateral y-direction, at specific downstream x-locations. It shows that, even in this worst case scenario, the two models share broad structural similarities. Overall, the wind farm wake expansion and extent of the wake region are consistent between TurbOPark and RANS-AWF. However, TurbOPark generally overestimates the magnitude of the wake deficit compared to RANS-AWF (Fig. 10 (a, f)), consistent with earlier findings that TurbOPark predicts stronger wind farm wake effects than RANS (van der Laan et al., 2023). The individual turbine wakes in TurbOPark also remain distinctly separated, especially for low TI, where the turbulent mixing is at a minimum, producing a jagged cross-stream profile, whereas the RANS-AWF solution shows significantly greater lateral mixing that merges neighbouring wakes into a smooth deficit profile. The two-

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

dimensional velocity maps (Fig. 10 (b–c)) confirm these observations. TurbOPark retains a jagged wake structure for all downstream distances, while the RANS-AWF field is laterally diffused a few rotor diameters downstream. From Fig. 10 (d), it can further be seen that the RANS-AWF dataset contains blockage upstream of the wind farm and speed ups on the side of the wind farm.

The quantile-to-quantile density scatter of the wake-region (Fig. 10(e)) quantifies this difference, giving an RMSE of 1.31%, an MAE of 0.48%, and a coefficient of determination of $R^2 = 0.47$. The low R^2 indicates that, despite the broad similarities between the wakes, the two models differ significantly in the point-wise magnitudes of the predicted wake deficits. The mean streamwise wake recovery curves (Fig. 10 (f)) further illustrate this difference. The lateral average wake deficit calculated by TurbOPark is consistently larger than the RANS-AWF reference downstream of the wind farm. From Fig. 10 (f), it can also be observed that the RANS-AWF model includes decreased wind speed upstream from the wind farm due to blockage.

Finally, the RMSE between the two models, binned by the free-stream velocity and ambient TI (Fig. 10(g)), shows a clear dependency on the inflow conditions. The largest difference between the two models occurs at low wind speeds and low I_0, where high thrust coefficients increase the individual wake deficits and reduce turbulent mixing, thereby limiting both turbulent mixing and streamwise recovery. Especially at wind speeds below $13\,\mathrm{m\,s^{-1}}$, the difference between TurbOPark and RANS-AWF is most significant.

3 Results and discussion

This section first presents the results of the architecture search for engineering-model-based source models, followed by the transfer learning results. Based on these, the best-performing ARU-Net and GNO models are selected and analysed together in terms of prediction accuracy and ability to resolve the wake region of the flow. This analysis comprises both a qualitative comparison, using cross-stream velocity deficit plots and flow field plots of velocity predictions from both models, and a quantitative comparison of performance metrics.

3.1 Engineering source model – architecture search

To investigate the effect of model architecture, a small hyperparameter search was performed for both the ARU-Net and the GNO. The best configuration from the ARU-Net hyperparameter search (Table 2) uses five encoder levels with channel counts of $(24, 48, 96, 192, 384)$, yielding 16,113,456 parameters. The optimiser enabled SE blocks, attention gates, and learnable downsampling, but preferred transposed convolutions (2×2 kernel, stride 2) over the bilinear-plus-learned-convolution alternative for upsampling. Training uses an effective batch size of 32 (physical batch size 8 with 4 gradient accumulation steps), 16-bit floating-point mixed-precision arithmetic, and no learning rate schedule. When trained to converge on TurbOPark, the selected model achieves a full-domain RMSE of $0.1050\,\mathrm{m\,s^{-1}}$ and MAE of $0.0225\,\mathrm{m\,s^{-1}}$ on the validation set. Table 3 shows the results of this search.

For the GNO, a simple grid search was conducted, as described in Sect. 2.6, and seven model variations were considered. The two best-performing GNO configurations are the variations closest to the originally best $\mathcal{G}_{\mathrm{org}}$ configuration described

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

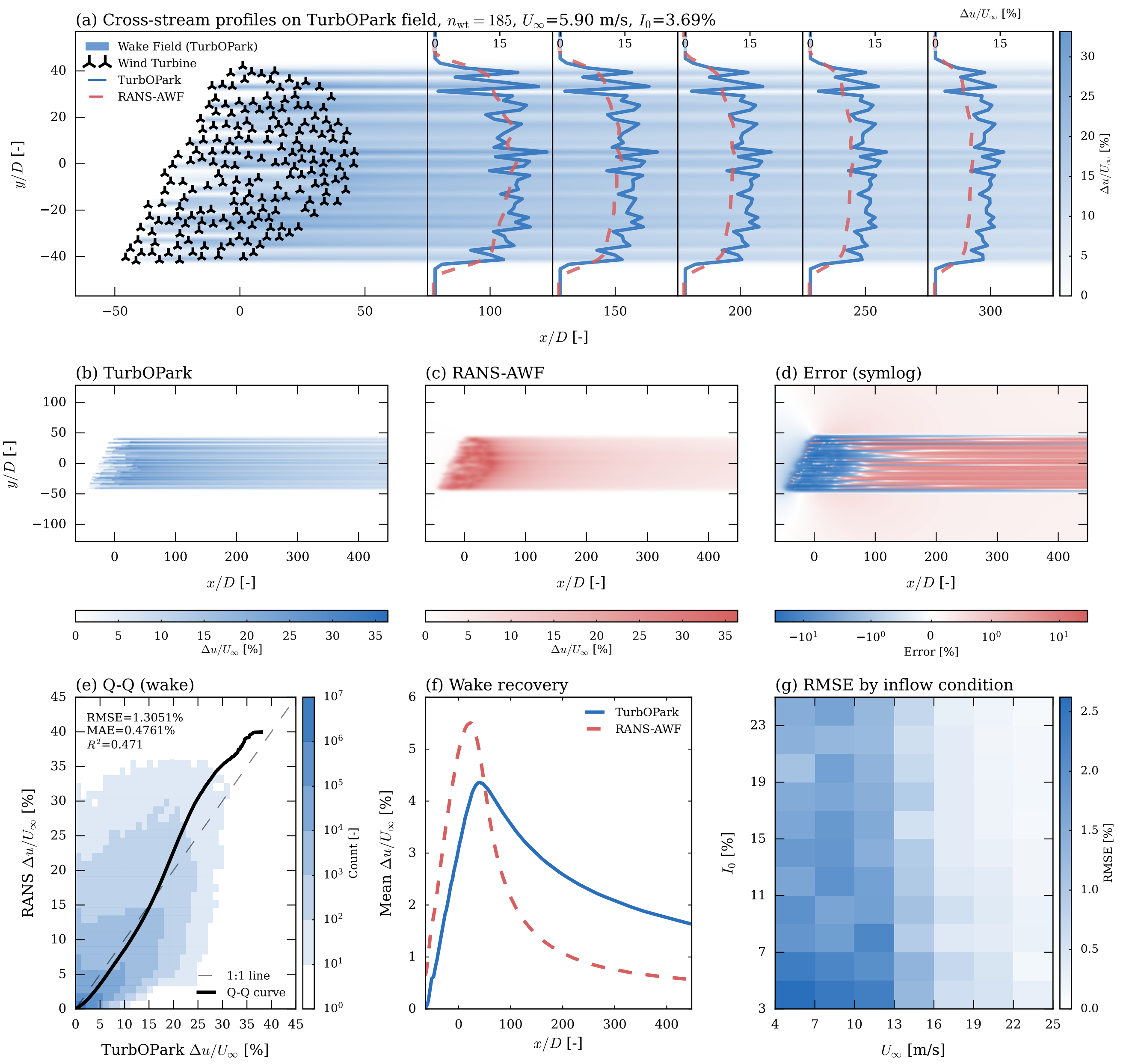

Figure 10. Comparison between TurbOPark and RANS-AWF. (a) Cross-stream velocity profiles ($\Delta u/U_\infty$). (b–c) Two-dimensional maps of $\Delta u/U_\infty$ for an arbitrary case.(d) Spatial difference map ($\Delta u/U_\infty$) on a symmetric logarithmic scale. (e) Quantile-to-quantile density scatter of effective velocity in the wake region, with RMSE, MAE, and R^2. (f) Streamwise wake recovery as laterally averaged $\Delta u/U_\infty$ versus x/D. (g) RMSE between models binned by freestream wind speed and turbulence intensity (I_0).

Table 3. Source model architecture search on TurbOPark dataset, evaluated on validation set. The ARU-Net architecture was selected via a 200-trial Optuna search (Table 2); the GNO results show the best result per architecture across 14 configurations (7 architectures × 2 initial learning rates). Best value per metric column in bold.

Model	Architecture						$\mathrm{RMSE_{val}}$ $[\mathrm{m\,s^{-1}}]$	$\mathrm{MAE_{val}}$ $[\mathrm{m\,s^{-1}}]$
Best ARU-Net	c_0 init. ch.	D_enc enc. depth	μ_c ch. mult.	c_max max ch.	k kernel	p_drop dropout		
	24	5	2.0	384	3×3	10%	**0.1050**	**0.0225**
GNO grid-search	Q latent dim.	q_int hidden width	L_int MLP layers	M_wt MP steps	q_dec dec. width	L_dec dec. layers		
$\mathcal{G}_\mathrm{L1}$	128	320	2	4	448	4	**0.2031**	**0.0755**
$\mathcal{G}_\mathrm{S1}$	64	160	2	3	224	2	0.2036	0.0811
$\mathcal{G}_\mathrm{org}$	100	250	2	3	350	3	0.2056	0.0779
$\mathcal{G}_\mathrm{L2}$	160	400	3	4	560	4	0.2058	0.0796
$\mathcal{G}_\mathrm{S2}$	50	125	2	2	175	2	0.2202	0.0888
$\mathcal{G}_\mathrm{S3}$	32	80	2	2	112	2	0.2340	0.1003
$\mathcal{G}_\mathrm{L3}$*	180	450	3	4	630	4	0.2720	0.1332

*Best result obtained with initial learning rate 10^{-3}.

in Schøler et al. (2025), namely the $\mathcal{G}_\mathrm{L1}$ and $\mathcal{G}_\mathrm{S1}$ models, followed by the $\mathcal{G}_\mathrm{org}$ setup itself. The best-performing $\mathcal{G}_\mathrm{L1}$ model uses slightly larger internal and decoder MLPs and employs four wind-turbine message-passing steps (M_wt) instead of the original three. The hyperparameter tuning done during the creation of the TurbOPark pretrained model is not exhaustive. A more thorough tuning could yield baseline models that yield better metrics. However, the authors believe that improvement would be minor.

3.2 Transfer learning – RANS-AWF hyperparameter search

Based on the results of the initial source-model training stage, a transfer learning experiment is conducted using combinations of the different approaches described in Sect. 2.7 with the chosen model architectures. The ARU-Net model architecture was selected as the best-performing model from the hyperparameter tuning on the TurbOPark dataset, as described in Sect. 2.5. The resulting hyperparameters for the ARU-Net are shown in Tab. 3. Before the model was used as the initial weights for the fine-tuning procedures, it was further trained to convergence, beyond the initial 50 epochs. This model is denoted as ARU-Net Best in Tab. 3. During fine-tuning, the exact same model architecture and hyperparameters were used as during the pretraining, except for the learning rate, which was tested for lr $\in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. For the LoRA-fine-tuned models, each learning rate was tested with ranks $r \in \{4, 8\}$. The GNO was tested across a range of transfer learning combinations, using

$\mathcal{G}_{\text{L1}}$ as the source model. As a significant number of variations were included, the models have been grouped into categories, and only the best in each group are shown in Tab. 4.

Table 4. Results of RANS-AWF transfer learning for both the ARU-Net and the GNO. The GNO results are based on the best L1 model weights from the TurbOPark training. Results are grouped by architecture and fine-tuning strategy, showing only the best configuration per category. N is the number of models in each category. Best model options lists the learning rate and LoRA r and α where applicable for the ARU-Net, and the LoRA r and α where applicable, and the setting of gradient clipping for the GNO. All models are trained on an Nvidia Quadro P4000 GPU. GNO metrics are reported to five decimal places to distinguish otherwise tied values.

Group	N	Best model options	Total Tunable Parameters	RMSE$_{\text{val}}$ $[\text{m}\,\text{s}^{-1}]$	MAE$_{\text{val}}$ $[\text{m}\,\text{s}^{-1}]$	Final epoch	Time
ARU-Net							
Full fine-tune	4	$\text{lr} = 10^{-5}$	16,113,456	**0.0169**	**0.0087**	1823	5h 5m
Freeze encoder	4	$\text{lr} = 10^{-5}$	7,662,696	0.0185	0.0092	3474	8h 21m
LoRA	8	LoRA ($r = 8$, $\alpha = 1$), $\text{lr} = 10^{-4}$	460,944	0.0275	0.0129	9467	28h 58m
From scratch	4	$\text{lr} = 10^{-3}$	16,113,456	0.0380	0.0153	548	1h 32m
GNO							
LoRA	12	LoRA ($r = 16$, $\alpha = 32$), no clip	291,312	**0.00558**	0.00234	417	7h 11m
Full fine-tune	2	no clip	2,704,916	0.00563	**0.00233**	360	5h 49m
Freeze $\mathcal{E}$	16	clip 1.0	2,406,850	0.00565	0.00245	309	3h 57m
Freeze $\mathcal{E}$+$\mathcal{A}$	16	LoRA ($r = 16$, $\alpha = 32$), no clip	245,792	0.00576	0.00255	255	3h 14m
From scratch	1	no clip	2,704,916	0.00696	0.00285	2712	43h 24m

As shown in Tab. 4, for both the ARU-Net and GNO, the LoRA-based approach yields no significant computational advantage over full fine-tuning or training from randomly initialised weights, despite the reduction in tunable parameters. This is likely because the full models are relatively small, leading the training process to be dominated by overhead, or because the ratio of tunable parameters in the LoRA configuration relative to the full model is not as distinct as in large language models (LLMs) with billions of parameters. For the ARU-Net, the full-finetuned model and the frozen-encoder variant perform comparably, whereas LoRA and training from scratch perform worse. This indicates that initialising the model weights and leaving them fully or partially tuneable is the most effective strategy, as it leverages the pre-learned wake physics while retaining sufficient flexibility to adapt to the high-fidelity data, unlike the restricted parameter space of LoRA or the uninformative starting point of random initialisation.

Table 4 shows that for the majority of the fine-tuning techniques used for fine-tuning the GNO, the results fall within an acceptable range. Even for the model, which is trained from scratch using randomly initialised parameters, the results are mostly acceptable. Interestingly, the full fine-tune group, which applies neither LoRA nor freezing, performs remarkably well, achieving the lowest MAE. Ultimately, the best model was a LoRA variant, but not by a wide margin.

Another intriguing insight is that the time column in Tab. 4 shows that using LoRA did not significantly reduce training time compared to simply using the full fine-tuning variations. The low effect of LoRA is surprising: the overall time per epoch is similar across groups, suggesting the cost reduction does not justify the added complexity. The LoRA group coming out ahead is therefore ascribed to the fact that twelve configurations were considered, and one of the initialisations managed to escape a local minimum that the full fine-tuning runs became trapped in before early stopping was triggered. This interpretation is supported by the fact that only one LoRA model outperformed the best full fine-tuning model.

3.3 Qualitative study – predictions with representative layout-inflow cases

The streamwise coordinate x is measured from the centre of the wind farm, as seen in Fig. 5. The distance from the most downstream turbine to a given wake location varies with wind farm size. To account for this, we introduce $\widetilde{x}$, defined as the streamwise distance from the most downstream turbine to that location. In Fig. 11, the nine selected combinations of farm layouts and inflow conditions described in Sect. 2.8 are used to create cross-stream deficit profiles at $\widetilde{x} = 6\,\mathrm{D}$ downstream distances for both predictions using the ground truth RANS-AWF data and the ARU-Net and GNO.

Figure 11 shows that both models produce good results and reproduce the ground truth with only minor inaccuracies. The areas where the ARU-Net has its most significant errors are dominated by a degree of over-smoothing, which is most notable in Fig. 11 (a) at $\widetilde{x} = 50D$, where the most significant relative discrepancy occurs. For the GNO, the error characteristics are not over-smoothing but rather a failure to capture the peak values accurately, primarily manifested as overestimation of peaks, e.g., as in Fig. 11 (e, i). In Fig. 12, the model predictions and ground truths are displayed as velocity field plots, with inclusions of model errors (ε). It is clear that both models perform worst in or near the farm. Simultaneously, there is a negative correlation between error and both free-stream velocity and ambient TI. This inverse correlation is due to wake physics: when the velocity is low, the turbines operate at high thrust coefficients, meaning they are in the state where they most affect the flow. At the same time, a low ambient TI means wake recovery is slow, as insufficient turbulence-induced mixing makes the wakes more persistent. These factors, when combined, produce a pronounced wake effect and a complex flow, both of which negatively affect model performance. That said, all investigated cases remain well within an acceptable error range for both models.

For the ARU-Net, another behaviour has been identified. It is clearest in the case of the small wind farm at low wind speed: when the turbines are placed far apart, and the farm is small, there is a low degree of wake mixing, and the convolutional kernels are not sufficient to resolve the finer details of the flow. This is observed mainly in areas where a neighbouring wind farm would not be placed, but it remains an interesting qualitative difference between the considered models. For the GNO, there are indications of a similar behaviour, though it is less pronounced. When the farm is dense, as in Fig. 11 (b) and (e), the GNO tends to overestimate the wake effect slightly, although the effect is not clear enough to draw a decisive conclusion.

To facilitate a direct visual comparison between the two models, their absolute errors are subtracted for each of the representation layout-inflow combinations. The resulting difference fields are shown in Fig. 13, where blue indicates that the ARU-Net has lower error and green indicates that the GNO has lower error. As seen in Fig. 13, the relative difference between the models is low, and both models have areas where they outperform the other. Most notably, the ARU-Net performs slightly better in the L, High case ($U_\infty = 19.7\,\mathrm{m\,s^{-1}}$, $I_0 = 25\,\%$) Fig. 13 (i), while the GNO performs better in the S, Low case ($U_\infty = 8.4\,\mathrm{m\,s^{-1}}$, $I_0 = 7\,\%$)Fig. 13 (a). Looking at the overall picture, the GNO appears to have a slight advantage over the ARU-Net, with more areas coloured green than blue. Also, the case where the ARU-Net performs best is with a high inflow velocity and a small

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

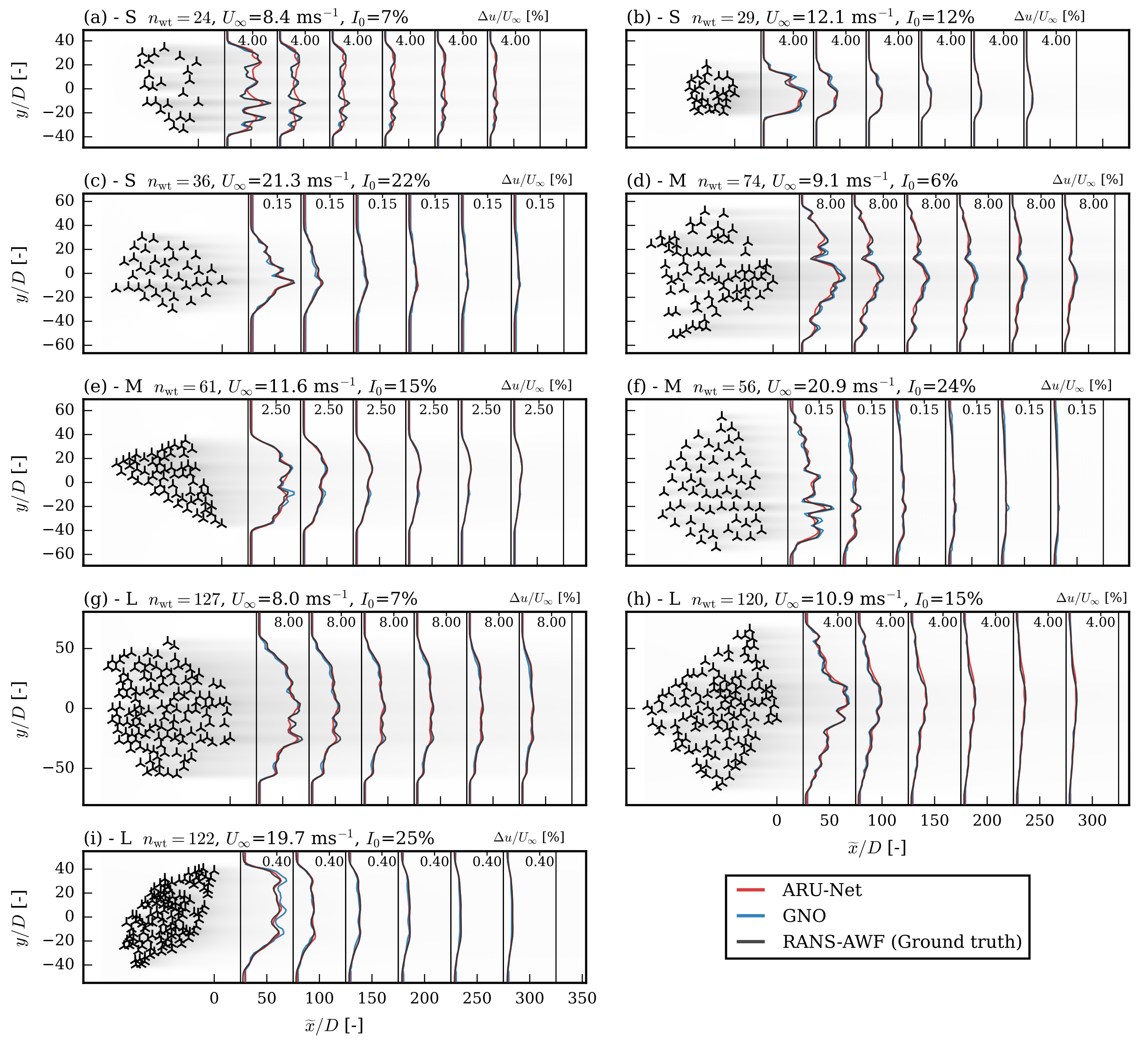

Figure 11. Cross-stream velocity deficits at selected distances behind the wind farm, comparing ARU-Net and GNO predictions against the RANS-AWF ground truth for a number of (a–c) small (S), (d–f) medium (M), and (g–i) large (L) farm layouts. Transverse profiles are extracted at downstream distances $\widetilde{x} \in \{50, 100, 150, 200, 250, 300\}\,D$ where $\widetilde{x}$ is measured from the turbine furthest downstream. The grey hue indicates the RANS-AWF flow field data.

nominal error, while the case where the GNO performs best is with a low velocity and a larger nominal error. However, because this advantage is primarily within the farm, it is not expected to yield a meaningful benefit in most practical applications.

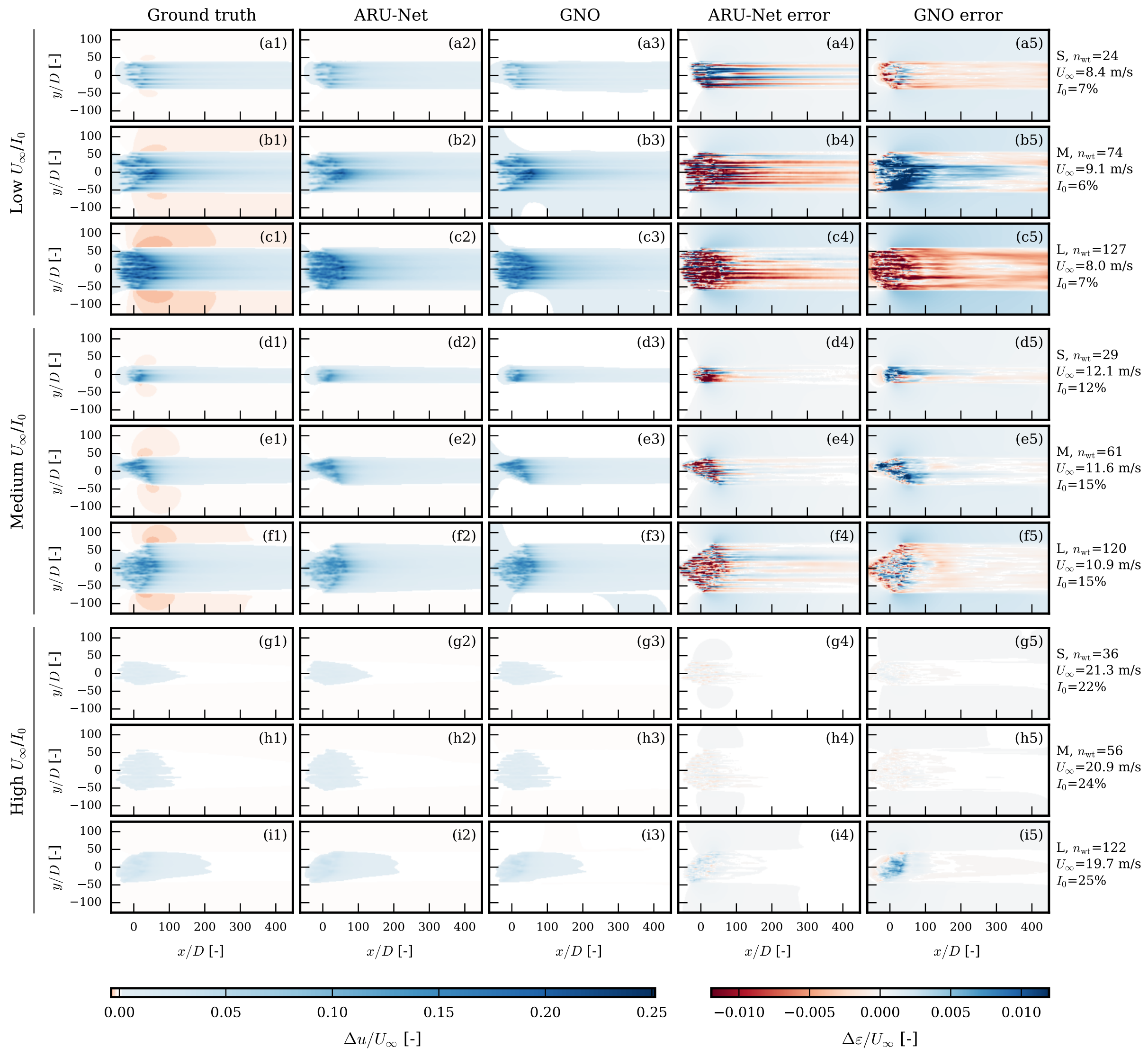

Figure 12. Flow field plots of the representative farm layout-inflow combinations. Columns indicate (a) RANS-AWF ground truth, (b) ARU-Net predictions, (c) GNO predictions, (d–e) Free-stream normalised model errors for the ARU-Net and GNO, respectively.

3.4 Quantitative study – performance metrics & sensitivity analysis

525 To supplement the more qualitative investigation of the selected representative layout inflow combinations, a quantitative comparison has been performed using the performance metrics introduced in Sect. 2.7. As a CNN-based approach requires a fixed, gridded domain, there will naturally be a lot of data with uninteresting flow, with no wake effect. While this area is partially of interest as a model should not predict strong wake effects where none are present, having it dominate the

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

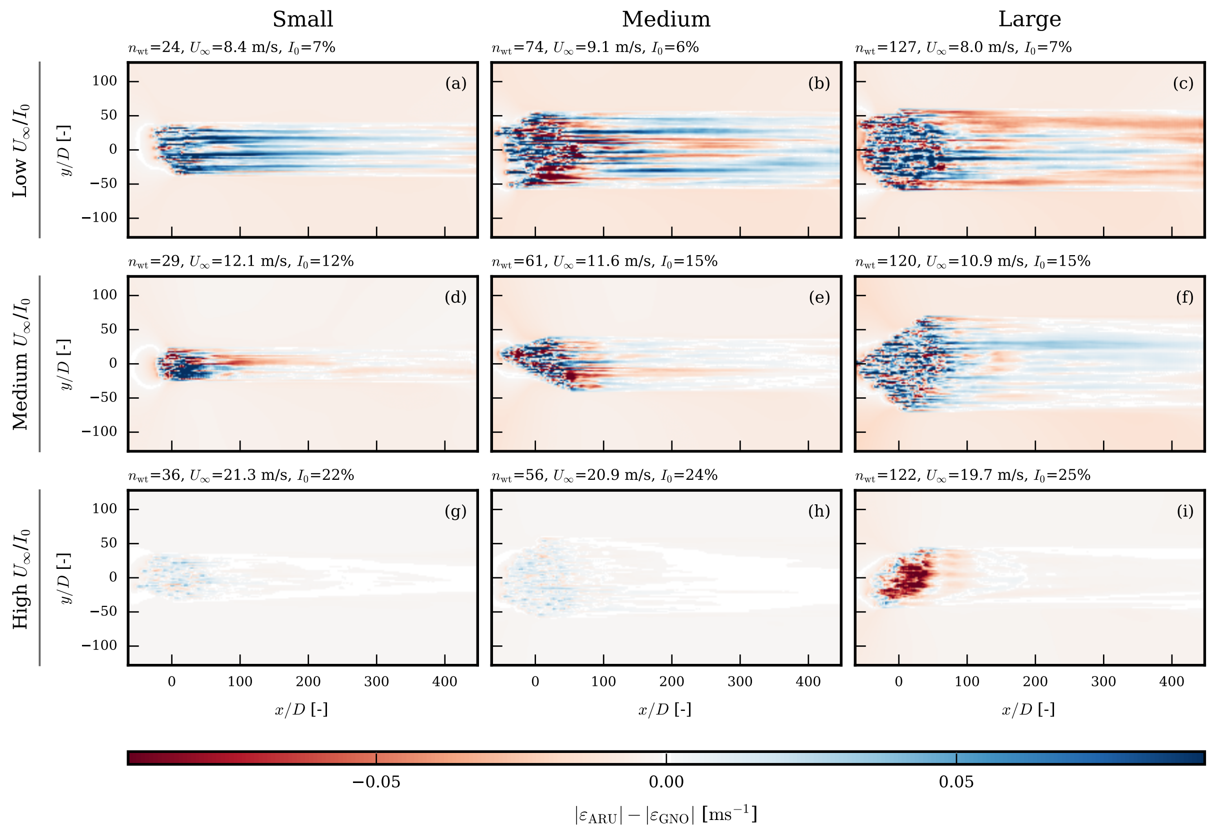

Figure 13. Model comparison using flow field error plots of the selected layout-inflow combinations.

performance metrics is suboptimal. Therefore, the performance metrics are also reported using a range of wake masks ($\delta_w \in$
530 $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}\,\mathrm{m\,s^{-1}}$) that have been applied on the RANS-AWF and only the data above the threshold has been considered in the metric evaluation. The resulting performance metrics are reported in Tab. 5.

Table 5 shows a mixed picture of both the ARU-Net and the GNO performing well in different categories. To summarise these results with a single pair of metrics, the wake region ($\delta_w \geq 10^{-3}\,\mathrm{m\,s^{-1}}$) grid-averaged RMSE is 0.024 $\mathrm{m\,s^{-1}}$ for the GNO and 0.028 $\mathrm{m\,s^{-1}}$ for the ARU-Net, while the corresponding F_1 scores are 0.91 and 0.98, respectively. However, both models
535 achieve better performance when evaluated on grids matching the resolution of their respective training data. The GNO has the lowest RMSE in most scenarios, while the ARU-Net has the best F_1-score most of the time. This underpins the observations from the qualitative study, which found that the GNO performs slightly better in overall accuracy, while the ARU-Net captures the wake boundary more accurately. The ARU-Net's ability to capture a well-defined wake boundary may be due to logarithmic scaling of the target data, which emphasises smaller values and rewards the network for capturing these changes.

540 To better understand how the models perform as input variables change, metrics are computed for each layout-inflow combination in the full test set. This is explored in Fig. 14, where the RMSE is binned by the different input variables, illustrating

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026
© Author(s) 2026. CC BY 4.0 License.

Table 5. CNN (ARU-Net) vs GNN (GNO) comparison across grids and wake deficit thresholds. Where interpolation is used on the ARU-Net predictions to evaluate on the GNN grid.

		CNN grid (256 × 128)		GNN grid (257 × 129)	
Threshold	Metric (test set)	ARU-Net	GNO	ARU-Net	GNO
No mask	RMSE [$\mathrm{m\,s^{-1}}$]	**0.0140 ± 0.0125**	0.0151 ± 0.0128	0.0176 ± 0.0135	**0.0159 ± 0.0131**
	MAE [$\mathrm{m\,s^{-1}}$]	**0.0073 ± 0.0067**	0.0103 ± 0.0079	**0.0083 ± 0.0071**	0.0105 ± 0.0080
	F1 [–]	–	–	–	–
$\delta_w \geq 10^{-5}$	RMSE [$\mathrm{m\,s^{-1}}$]	0.0209 ± 0.0188	**0.0205 ± 0.0203**	0.0269 ± 0.0201	**0.0221 ± 0.0208**
	MAE [$\mathrm{m\,s^{-1}}$]	**0.0108 ± 0.0108**	0.0122 ± 0.0127	0.0133 ± 0.0116	**0.0128 ± 0.0130**
	F1 [–]	0.5971 ± 0.1277	**0.6019 ± 0.1356**	0.5902 ± 0.1259	**0.5999 ± 0.1357**
$\delta_w \geq 10^{-4}$	RMSE [$\mathrm{m\,s^{-1}}$]	0.0214 ± 0.0189	**0.0209 ± 0.0205**	0.0275 ± 0.0202	**0.0225 ± 0.0210**
	MAE [$\mathrm{m\,s^{-1}}$]	**0.0112 ± 0.0110**	0.0124 ± 0.0129	0.0139 ± 0.0119	**0.0131 ± 0.0132**
	F1 [–]	**0.9944 ± 0.0081**	0.6263 ± 0.1598	**0.9903 ± 0.0089**	0.6246 ± 0.1602
$\delta_w \geq 10^{-3}$	RMSE [$\mathrm{m\,s^{-1}}$]	0.0239 ± 0.0192	**0.0234 ± 0.0212**	0.0311 ± 0.0201	**0.0253 ± 0.0215**
	MAE [$\mathrm{m\,s^{-1}}$]	**0.0137 ± 0.0116**	0.0147 ± 0.0138	0.0173 ± 0.0122	**0.0156 ± 0.0139**
	F1 [–]	**0.9850 ± 0.0141**	0.9079 ± 0.1180	**0.9804 ± 0.0162**	0.9054 ± 0.1184

how the error behaves for the two considered models and for their variations trained from scratch during the transfer learning study. Figure 14 presents a comparison between the best-performing fine-tuned ARU-Net models and equivalent models trained without weight initialisation. The results indicate that the ARU-Net benefits significantly from pre-training compared to the GNO, with the RMSE often reduced by more than half in specific regimes. These improvements are most pronounced at wind speeds near the rated velocity ($U_\infty \in [10,15]\,\mathrm{m\,s^{-1}}$), in the lower turbulence intensity range ($I_0 \in [3,8]\%$), and within the largest ($n_{wt} \in [70,190]$) and densest ($s_{wt}/D \in [3,7]$) wind farms. In contrast, the GNO only minor benefits from transfer learning, showing only slight improvements or, in some instances, slightly degraded performance.

While more rigorous experimentation would be required to ascertain the precise cause of this behaviour, a plausible explanation is the structural differences between the two architectures. The GNO operates on an input-output graph whose nodes correspond to physical turbine positions, meaning the farm layout is encoded directly in the data representation fed to the model. The ARU-Net also receives layout information via a raster representation, which is propagated across the encoder and decoder stages via skip connections. However, this constitutes a weaker form of spatial enforcement than the explicit node-wise correspondence of the GNO graph. This stronger geometric inductive bias in the GNO may allow it to learn the underlying flow physics more effectively from limited high-fidelity data, reducing its dependence on low-fidelity pre-training and explaining the comparatively modest gains it achieves through transfer learning. From Fig. 14, it is clear that the ARU-Net trained from scratch performs much worse than any of the other models. It is, however, less clear whether the fine-tuned ARU-Net or GNO model performs better, though the GNO appears to have a slight edge, consistent with its lower RMSE in most scenarios in Tab. 5. It is important to note that the main objective of this work was to evaluate the ARU-Net and the GNO in capturing

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

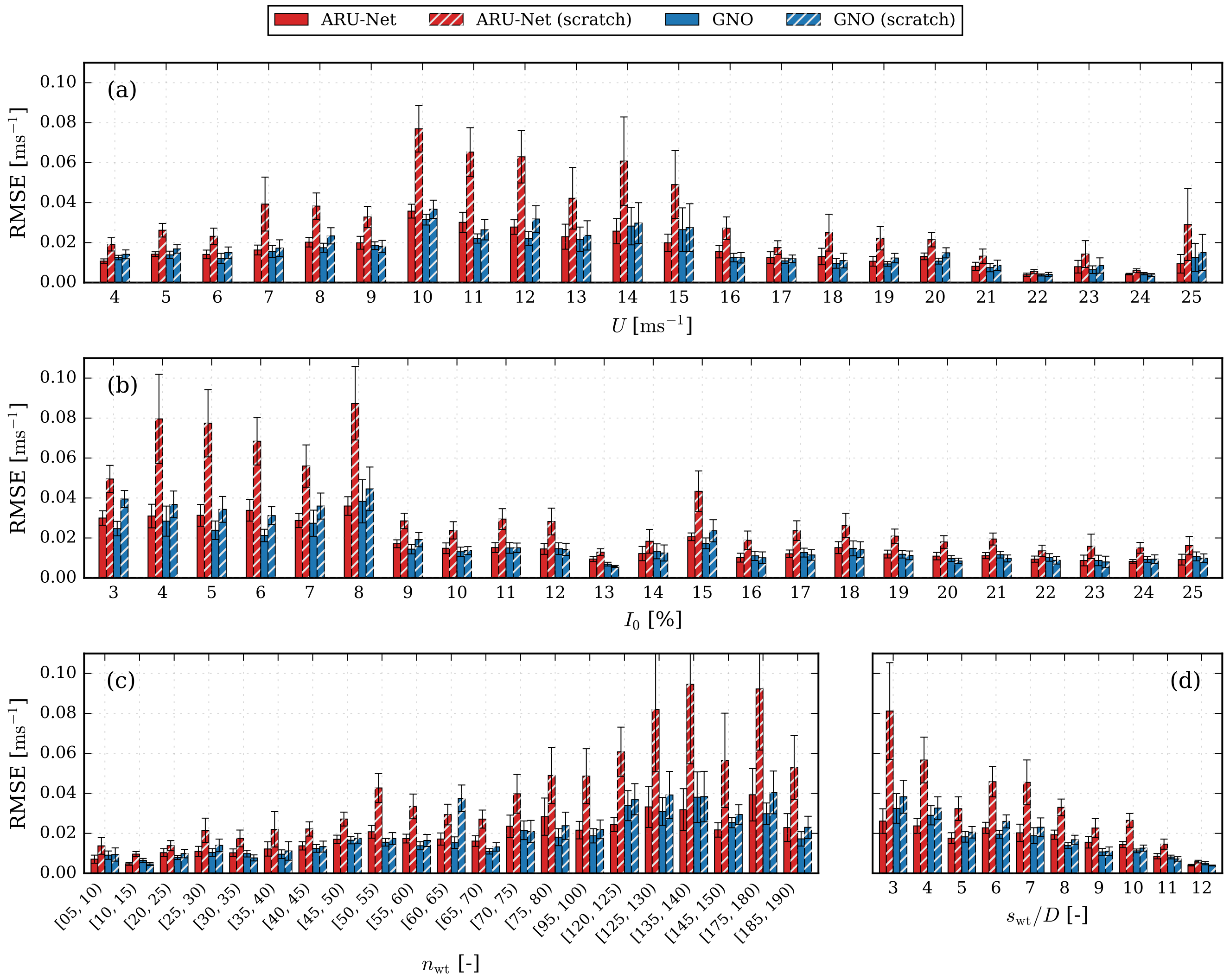

Figure 14. ARU-Net and GNO binned RMSE metrics with respect to inputs: (a) free-stream velocity U_∞, (b) ambient TI I_0, (c) number of wind turbines n_{wt}, and (d) wind turbine separation factor s_{wt}.

the inter-farm effects simulated by the reference RANS-AWF dataset, rather than demonstrating that these predicted wake deficits are comparable to those generated by large wind turbine clusters in practice. The RANS-AWF dataset is limited to the capabilities of both RANS and the conditions simulated. As mentioned in **??**, inter-farm wake effects can be highly influenced by large-scale atmospheric variability. Therefore, we also plan to extend the evaluation of the methods proposed in this work to account for the range of multiscale phenomena encountered by offshore wind farms.

3.5 Future work

The models presented in this work show great promise, and several avenues for further development have emerged. Drawing on the complementary strengths of each architecture, one promising direction is to use the ARU-Net's superior capability to characterise the wake region as a spatially informed background field for the GNO, replacing the current uniform background-flow assumption. This hybridisation could enable the GNO to match the ARU-Net in wake boundary prediction accuracy, as both models are already orders of magnitude faster than full RANS-AWF. This additional inference cost is not expected to be problematic. Alternatively, a more specialised ARU-Net could be trained exclusively to capture the wake boundary, potentially surpassing the current capability demonstrated here. From the ARU-Net perspective, the neural operator framework could be adopted by replacing the current convolutional kernels with Fourier layers similar to those in Fourier Neural Operator (FNO) networks, potentially improving generalisation to unseen spatial configurations. Finally, the scope of the training data could be broadened in several directions: extension to three-dimensional flow fields, inclusion of C_T curves to account for multiple turbine types, use of higher-fidelity full RANS simulations, and incorporation of more complex atmospheric conditions such as Coriolis force, atmospheric stability, and mesoscale effects.

4 Conclusions

This work presented a systematic comparison between an ARU-Net and a GNO as surrogate models for inter-farm wake deficit predictions. Both architectures were pre-trained on low-fidelity TurbOPark engineering model data and fine-tuned on high-fidelity RANS-AWF simulations using a multi-fidelity transfer learning strategy, though the GNO benefited only marginally from this approach. Both architectures have demonstrated strong capability to predict downstream wake deficits across a wide range of arbitrarily generated wind farm layouts and randomly generated ambient wind speed and turbulence intensity. The two models have shown complementary strengths: evaluated over the wake region ($\delta_w \geq 10^{-3}\,\mathrm{m\,s^{-1}}$) and averaged across both evaluation grids, the GNO achieves a lower RMSE (0.024 vs. $0.028\,\mathrm{m\,s^{-1}}$), indicating slightly better accuracy in estimating the flow field wind speed, while the ARU-Net attains a higher F_1 score (0.98 vs. 0.91), indicating a better ability to correctly predict wake boundaries. The advantage of the ARU-Net for predicting wake boundaries may be attributed to its logarithmic target transform, which improves its ability to estimate wake deficits across different magnitudes by equally rewarding the ARU-Nets ability to predict subtle and strong wake effects.

Qualitative analysis has revealed distinct failure modes for each network architecture. The ARU-Net is prone to over-smoothing its predictions, particularly for sparse wind farms with large turbine spacing, where the convolutional kernel was

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026
© Author(s) 2026. CC BY 4.0 License.

insufficient to resolve fine flow structures. In contrast, the GNO tended to overestimate peak wake deficits, most notably in dense wind farm configurations. Both models perform worst at low wind speeds and low ambient TI, consistent with the underlying wake physics: these conditions are dominated by high thrust coefficients and low turbulent mixing, leading to stronger, more persistent, and more complex wake structures, which are inherently more difficult to predict.

The transfer learning results demonstrate that the ARU-Net benefits substantially more from pretraining on engineering models than the GNO, with RMSE reductions often exceeding 50% in specific regimes compared to training from scratch. In contrast, the GNO has shown only marginal improvements from transfer learning, suggesting that its intrinsic graph structure, where nodes directly represent physical turbines and edges encode positional relationships, already provides a strong inductive bias for learning wake interactions even with limited high-fidelity data. Regarding fine-tuning the full-tuning and frozen encoder strategy performed comparably for the ARU-Net, while using the LoRA-based parameter efficient fine-tuning offered no significantly computational or accuracy benefit over training the model from scratch for either architecture, likely due to the relatively small size of the models compared to large language models for which LoRA was originally designed.

From a practical standpoint, the choice between the two models involves trade-offs beyond pure prediction accuracy. The ARU-Net operates on fixed resolution grids, making it straightforward to obtain complete flow fields, but it offers limited flexibility when wind farm domains vary in size and resolution. The GNO, being grid-invariant, can evaluate the flow at arbitrary spatial locations without recomputing the upstream turbine graph, offering greater flexibility for diverse farm configurations and variable resolution queries.

Code and data availability. The code used to train the GNO is publicly available at https://github.com/jenspeterschoeler/Wind-Farm-GNO. The code used to train the ARU-Net is publicly available at https://github.com/FPWRasmussen/Wind-Farm-ARU-Net. The RANS-AWF dataset using the DTU-10-MW reference turbine has been made available at Zenodo (**?**).

Appendix A: Convolution and message-passing mathematical equivalency

Consider a graph $G = (V, E)$ where each node i carries a feature vector $\boldsymbol{h}_i$ and each directed edge from j to i carries an edge feature $\boldsymbol{a}_{ji}$. A single message-passing step computes

$$\boldsymbol{h}_i' = \rho_{e\to v}\big(\{\,\phi(\boldsymbol{h}_j, \boldsymbol{a}_{ji}) \mid j \in \mathcal{N}(i)\,\}\big), \tag{A1}$$

where $\mathcal{N}(i)$ is the set of neighbours of node i, ϕ is a learnable node update function, and $\rho_{e\to v}$ is a permutation-invariant aggregation. Now consider a regular pixel grid in which nodes are arranged on a lattice, and each node is connected to its neighbours within a $k \times k$ footprint. If the edge features encode the relative spatial offset $\boldsymbol{a}_{ji}$ between nodes, the aggregation is chosen as summation, and the node update function returns an offset-dependent weight matrix multiplied by the sender feature, i.e. $\phi(\boldsymbol{h}_j, \boldsymbol{a}_{ji}) = \mathbf{K}_{j-i}\,\boldsymbol{h}_j$, then Eq. (A1) reduces to

$$\boldsymbol{h}_i' = (\mathbf{K} * \boldsymbol{h})_i = \sum_{j \in \mathcal{N}(i)} \mathbf{K}_{j-i}\,\boldsymbol{h}_j\,, \tag{A2}$$

where $*$ denotes the discrete convolution operator and $\mathbf{K} \in \mathbb{R}^{k\times k\times c_{\text{out}}\times c_{\text{in}}}$ is the learnable convolution kernel of spatial extent k, whose entries $\mathbf{K}_{j-i} \in \mathbb{R}^{c_{\text{out}}\times c_{\text{in}}}$ are weight matrices indexed by the relative offset between nodes. This is precisely the standard discrete convolution used in CNNs. In other words, CNNs exploit the fixed, regular structure of the grid to share weights by spatial offset. In contrast, GNNs achieve the same effect through edge connections during the message passing without the strict requirement of a regular grid.

Author contributions. JPS, FPWR and PER conceived the research; FPWR developed the code and methodology around the ARU-Net, JPS developed the code and methodology around the GNO, MPL generated the RANS-AWF data, both JPS and FPWR performed the analysis and contributed to the model comparison code, MPL, AP and PER supervised the work; JPS and FPWR wrote the original draft; and JPS, FPWR, MPL, AP, and PER reviewed the draft.

Acknowledgements. This work was partially funded by TotalEnergies under the "Inter-Farm Interactions (IFI)" project. Model training was performed on the Sophia cluster (Technical University of Denmark, 2019). The authors also acknowledge the use of AI tools in preparing this manuscript and the associated code. In particular, Claude, Gemini, and Grammarly were used for proofreading, and Claude-Code for code development. All AI-assisted work has been reviewed by the authors.

Competing interests. At least one of the (co-)authors is a member of the editorial board of *Wind Energy Science*. The authors have no other competing interests to declare.

References

Agarap, A. F.: Deep Learning using Rectified Linear Units (ReLU), http://arxiv.org/abs/1803.08375, 2019.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M.: Optuna: A Next-generation Hyperparameter Optimization Framework, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2623–2631, https://doi.org/10.1145/3292500.3330701, 2019.

Anagnostopoulos, S. J., Bauer, J., Clare, M. C., and Piggott, M. D.: Accelerated wind farm yaw and layout optimisation with multi-fidelity deep transfer learning wake models, Renewable Energy, 218, 119 293, https://doi.org/10.1016/J.RENENE.2023.119293, 2023.

Bak, C., Zahle, F., Bitsche, R., Kim, T., Yde, A., Henriksen, L., Hansen, M., Blasques, J., Gaunaa, M., and Natarajan, A.: The DTU 10-MW Reference Wind Turbine, https://orbit.dtu.dk/en/publications/the-dtu-10-mw-reference-wind-turbine/, danish Wind Power Research 2013 ; Conference date: 27-05-2013 Through 28-05-2013, 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.

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.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P.: Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, Openreviews, http://arxiv.org/abs/2104.13478, 2021.

Cañadillas, B., Beckenbauer, M., Trujillo, J. J., Dörenkämper, M., Foreman, R., Neumann, T., and Lampert, A.: Offshore wind farm cluster wakes as observed by long-range-scanning wind lidar measurements and mesoscale modeling, Wind Energ. Sci, 7, 1241–1262, https://doi.org/10.5194/wes-7-1241-2022, 2022.

Daenens, S., Verstraeten, T., Daems, P.-J., Nowé, A., and Helsen, J.: Spatio-temporal graph neural networks for power prediction in offshore wind farms using SCADA data, Wind Energy Science, 10, 1137–1152, https://doi.org/10.5194/wes-10-1137-2025, 2025.

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.

Djath, B., Schulz-Stellenfleth, J., and Cañadillas, B.: Impact of atmospheric stability on X-band and C-band synthetic aperture radar imagery of offshore windpark wakes, Journal of Renewable and Sustainable Energy, 10, 17, https://doi.org/10.1063/1.5020437, 2018.

DTU Wind and Energy Systems: PyWakeEllipSys v6.0, https://topfarm.pages.windenergy.dtu.dk/cuttingedge/pywake/pywake_ellipsys, accessed: 5/03-26, 2026.

Duthé, G., de Nolasco Santos, F., Abdallah, I., Réthore, P.-E., 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.

Duthé, G., Abdallah, I., and Chatzi, E.: Graph Transformers for inverse physics: reconstructing flows around arbitrary 2D airfoils, http://arxiv.org/abs/2501.17081, 2025.

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

Frandsen, S. T.: Turbulence and turbulence-generated structural loading in wind turbine clusters, Ph.D. thesis, Risø, ISBN 87-550-3458-6, https://orbit.dtu.dk/en/publications/turbulence-and-turbulence-generated-structural-loading-in-wind-tu/, risø-R-1188(EN), 2007.

García-Santiago, O., Hahmann, A. N., Badger, J., and Peña, A.: Evaluation of wind farm parameterizations in the WRF model under different atmospheric stability conditions with high-resolution wake simulations, Wind Energy Science, 9, 963–979, https://doi.org/10.5194/wes-9-963-2024, 2024.

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.

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.

Hasager, C. B., Vincent, P., Badger, J., Badger, M., Bella, A. D., Peña, A., Husson, R., and Volker, P. J.: Using Satellite SAR to Characterize the Wind Flow around Offshore Wind Farms, Energies 2015, Vol. 8, Pages 5413-5439, 8, 5413–5439, https://doi.org/10.3390/EN8065413, 2015.

Hasanpoor, S., Romero, D. A., Moran, J. E., and Amon, C. H.: Physics-informed deep convolutional hierarchical encoder-decoder neural network for flow field prediction in wind farms, Energy and AI, 21, 100 553, https://doi.org/10.1016/J.EGYAI.2025.100553, 2025.

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.

Hou, G., Zhang, F., Huang, C., and Huang, T.: Multivariate modeling on wake-affected wind farms by two-stage hybrid graph neural network, Applied Energy, 402, 127 018, https://doi.org/10.1016/j.apenergy.2025.127018, 2026.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W.: LoRA: Low-Rank Adaptation of Large Language Models, http://arxiv.org/abs/2106.09685, 2021.

Hu, J., Shen, L., and Sun, G.: Squeeze-and-Excitation Networks, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 7132–7141, https://doi.org/10.1109/CVPR.2018.00745, 2018.

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

Jensen, N. O.: A note on wind generator interaction - RISØ-M-2411, Tech. rep., Risø National Laboratories, 1983.

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.

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

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.

Loshchilov, I. and Hutter, F.: Decoupled Weight Decay Regularization, http://arxiv.org/abs/1711.05101, 2019.

Michelsen, J.: Basis3D - a Platform for Development of Multiblock PDE Solvers: β - release, vol. AFM 92-05, Technical University of Denmark, https://orbit.dtu.dk/en/publications/basis3d-a-platform-for-development-of-multiblock-pde-solvers-%CE%B2-re/, 1992.

Misra, D.: Mish: A Self Regularized Non-Monotonic Activation Function, 31st British Machine Vision Conference, BMVC 2020, https://doi.org/10.5244/C.34.191, 2020.

https://doi.org/10.5194/wes-2026-54
Preprint. Discussion started: 30 March 2026

Nygaard, N. G. and Newcombe, A. C.: Wake behind an offshore wind farm observed with dual-Doppler radars, Journal of Physics: Conference Series, 1037, 072 008, https://doi.org/10.1088/1742-6596/1037/7/072008, 2018.

Nygaard, N. G., Steen, S. T., Poulsen, L., and Pedersen, J. G.: Modelling cluster wakes and wind farm blockage, in: Journal of Physics: Conference Series, vol. 1618, IOP Publishing Ltd, ISSN 17426596, https://doi.org/10.1088/1742-6596/1618/6/062072, 2020.

Nygaard, N. G., Pedersen, J. G., Hansen, S. D., and Krastins, P.: A Turbulence Optimized Park model, Tech. rep., Ørsted, https://github.com/OrstedRD/TurbOPark/blob/main/TurbOPark%20description.pdf, 2022.

Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., Glocker, B., and Rueckert, D.: Attention U-Net: Learning Where to Look for the Pancreas, https://arxiv.org/abs/1804.03999v3, 2018.

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.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: Advances in Neural Information Processing Systems 32, edited by Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., pp. 8024–8035, Curran Associates, Inc., 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.

Peña, A., Mirocha, J. D., and van der Laan, M. P.: Evaluation of the Fitch Wind-Farm Wake Parameterization with Large-Eddy Simulations of Wakes Using the Weather Research and Forecasting Model, Monthly Weather Review, 150, 3051–3064, https://doi.org/10.1175/MWR-D-22-0118.1, 2022.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W.: Learning Mesh-Based Simulation with Graph Networks, http://arxiv.org/abs/2010.03409, 2021.

Platis, A., Siedersleben, S. K., Bange, J., Lampert, A., Bärfuss, K., Hankers, R., Cañadillas, B., Foreman, R., Schulz-Stellenfleth, J., Djath, B., Neumann, T., and Emeis, S.: First in situ evidence of wakes in the far field behind offshore wind farms, Scientific reports, 8, https://doi.org/10.1038/S41598-018-20389-Y, 2018.

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.

Rasmussen, F. P. W., van der Laan, P., Peña, A., and Réthoré, P.-E.: Transfer Learning Approach for Improved Wind Farm Wake Prediction Using Multi-Fidelity Convolutional Neural Networks, in: Proceedings of the ASME 2026 45th International Conference on Ocean, Offshore and Arctic Engineering (OMAE2026), OMAE2026-181761, 2026.

Ronneberger, O., Fischer, P., and Brox, T.: U-net: Convolutional networks for biomedical image segmentation, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9351, 234–241, https://doi.org/10.1007/978-3-319-24574-4_28, 2015.

Sanchez-Lengeling, B., Reif, E., Pearce, A., and Wiltschko, A.: A Gentle Introduction to Graph Neural Networks, Distill, 6, https://doi.org/10.23915/distill.00033, 2021.

Schneemann, J., Rott, A., Dörenkämper, M., Steinfeld, G., and Kühn, M.: Cluster wakes impact on a far-distant offshore wind farm's power, Wind Energy Science, 5, 29–49, https://doi.org/10.5194/WES-5-29-2020, 2020.

Schøler, J. P., Peder Weilmann Rasmussen, F., Quick, J., and Réthoré, P.-E.: Graph Neural Operator for windfarm wake flow, Wind Energy Science Discussions, 2025, 1–38, https://doi.org/10.5194/wes-2025-261, 2025.

Sørensen, N. N., Bechmann, A., Johansen, J., Myllerup, L., Botha, P., Vinther, S., and Nielsen, B. S.: Identification of severe wind conditions using a Reynolds Averaged Navier-Stokes solver, Journal of Physics: Conference series, 75, 1–13, https://doi.org/10.1088/1742-6596/75/1/012053, 2007.

Sørensen, N.: General purpose flow solver applied to flow over hills, Ph.D. thesis, https://orbit.dtu.dk/en/publications/general-purpose-flow-solver-applied-to-flow-over-hills/, published 2003, 1995.

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

Thuerey, N., Weißenow, K., Prantl, L., and Hu, X.: Deep Learning Methods for Reynolds-Averaged Navier–Stokes Simulations of Airfoil Flows, https://doi.org/10.2514/1.J058291, 58, 25–36, https://doi.org/10.2514/1.J058291, 2019.

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.

van der Laan, M. P., García-Santiago, O., Kelly, M., Meyer Forsting, A., Dubreuil-Boisclair, C., Sponheim Seim, K., Imberger, M., Peña, A., Sørensen, N. N., and Réthoré, P.-E.: A new RANS-based wind farm parameterization and inflow model for wind farm cluster modeling, Wind Energy Science, 8, 819–848, https://doi.org/10.5194/WES-8-819-2023, 2023.

van der Laan, M. P., Meyer Forsting, A., and Réthoré, P.-E.: A Computational Fluid Dynamics surrogate model for wind turbine interaction including atmospheric stability, Wind Energy Science Discussions, 2026, 1–25, https://doi.org/10.5194/wes-2025-287, 2026.

Wang, Q., Hu, J., Yang, S., Ti, Z., and Deng, X.: Knowledge-Fusion Graph Transformer network for wind farm assessment with sparse data, Renewable Energy, 256, 124 654, https://doi.org/10.1016/j.renene.2025.124654, 2026.

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.