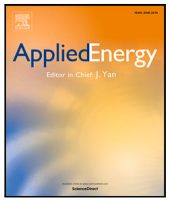
Applied Energy 339 (2023) 120928

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



End-to-end wind turbine wake modelling with deep graph representation learning

Siyi Li, Mingrui Zhang, Matthew D. Piggott *

Department of Earth Science and Engineering, Imperial College London, London, SW7 2AZ, UK

ARTICLE INFO

Keywords:
Geometric deep learning
Graph neural networks
Computational fluid dynamics
Wind turbine wake modelling
Wind farm power

ABSTRACT

Wind turbine wake modelling is of crucial importance to accurate resource assessment, to layout optimisation, and to the operational control of wind farms. This work proposes a surrogate model for the representation of wind turbine wakes based on a state-of-the-art graph representation learning method termed a graph neural network. The proposed end-to-end deep learning model operates directly on unstructured meshes and has been validated against high-fidelity data, demonstrating its ability to rapidly make accurate 3D flow field predictions for various inlet conditions and turbine yaw angles. The specific graph neural network model employed here is shown to generalise well to unseen data and is less sensitive to over-smoothing compared to common graph neural networks. A case study based upon a real world wind farm further demonstrates the capability of the proposed approach to predict farm scale power generation. Moreover, the proposed graph neural network framework is flexible and highly generic and as formulated here can be applied to any steady state computational fluid dynamics simulations on unstructured meshes.

1. Introduction

As one of the cleanest and most sustainable sources of renewable energy, wind energy has been undergoing rapid and unabated expansion worldwide. As the capacity of wind turbine farms increases, through the potentially closer clustering of increasing numbers of larger turbines to most efficiently exploit the available wind energy resource, it is inevitable that downstream turbines will at some times be operating within the full or partial wakes of upstream turbines. This can lead to reduced power generation as well as increased structural loads. Consequently, wind turbine wake modelling has been widely considered as one of the most crucial aspects of the optimal design and operational control of wind farms, see [1] and the references therein.

Wake models across different levels of fidelity have been thoroughly studied by researchers over the years. Analytical models including the Jensen model [2], the Larsen model [3] and the Gaussian wake model [4] are commonly implemented in industrial standard software such as FLORIS [5], thanks to their very rapid execution speed, however their accuracy is consequently limited. In comparison, higher fidelity models based on computational fluid dynamics (CFD) simulations, such as Reynolds-Averaged Navier–Stokes (RANS) or Large Eddy Simulation (LES), can provide more accurate flow field predictions but at significantly higher computational cost and execution time, hampering their value for rapid resource assessment, and as part of iterative design optimisation and control tools. For instance, the computing time required by RANS modelling for the simulation of a wind farm tends to be in the order of several CPU hours, whereas LES simulations could take days of distributed computation on hundreds of processors [6].

One possible approach to retain high accuracy in wake predictions while simultaneously maintaining short computation times is through the utilisation of deep learning algorithms trained on high-fidelity CFD data. The work presented here aims to develop a novel data-driven wake model that is based on machine learning and high-fidelity CFD simulations. In particular, this work utilises a graph representation learning method termed a graph neural network (GNN), which is based on a nascent deep learning research area operating on graph structured data. By operating directly on unstructured CFD meshes, the GNN approach eliminates the need for the training data to be interpolated to a uniform grid, as is commonly performed. It can therefore better preserve flow details through the accommodation of the different spatial resolutions across the simulation domain that are often beneficial for the CFD study of multi-scale fluid dynamical processes. A well-trained GNN on high-fidelity flow field data is thus able to capture the entirety of the primary characteristics of the wake flow structure, which cannot be achieved by analytical wake models, while maintaining competitive evaluation speeds.

* Corresponding author.
E-mail address: m.d.piggott@imperial.ac.uk (M.D. Piggott).

https://doi.org/10.1016/j.apenergy.2023.120928
Received 29 November 2022; Received in revised form 5 February 2023; Accepted 1 March 2023
Available online 21 March 2023

The primary novelties and contributions of this work can be summarised as follows:

1. A novel wind turbine wake model based on graph representation learning was developed and trained on RANS CFD data. The use of graph neural networks is of particular significance as many leading high-fidelity CFD solvers seek to increase their efficiency in the simulation of complex, multi-scale problems through the use of unstructured meshes, block-structured meshes, or some other non-regular discretisation of the spatial domain. This is in contrast to most of the prior works on deep learning methods for wind turbine wake modelling which have typically operated on uniform grids.
2. To the best of the authors' knowledge, this work is also one of the first attempts to leverage graph neural networks in the modelling of relatively large-scale, 3D CFD data comprising of the order of hundreds of thousands of graph vertices. The nature of this task entails the construction of deep graph neural networks, which are known to suffer from over-smoothing. To this end, this work explored the usage of various different graph neural network models and architectures, and conducted extensive experimentation, before adopting the *GraphSAGE* (Graph SAmple and aggreGatE) model which has great scalability with large data sizes. Moreover, the deep GraphSAGE neural network was further improved by adding a *jumping knowledge layer* and *layer-wise* as well as *initial residual connections*.
3. In order to train the deep learning surrogate model, RANS-based wake data was generated using the *generalised actuator disk (GAD) model* coupled with the CFD solver package *OpenFOAM*. The GAD model's ability to simulate turbine wakes and turbine wake interactions were further validated against wind tunnel tests performed at the Norwegian University of Science and Technology (NTNU). Also developed within the proposed framework was the ability to convert OpenFOAM based meshes to graph data structures that are compatible as input to GNNs. The performance of the GNN model in predicting single turbine wakes was extensively tested at varying levels of inlet velocities, turbulence intensities as well as turbine yaw angles. The resulting relative accuracy in predicting flow velocity on data unseen during training reached a median of 99.71%, with each prediction capable of being made within 15 ms.
4. The ability of the proposed geometric deep learning approach to model wind farms was further tested on a real-scale case study based on Sweden's Lillgrund offshore wind farm, by superimposing multiple individually calculated turbine wakes. The results showed that the proposed model can accurately predict the generated power in comparison to both full CFD simulations of the farm as well as direct observation data. In addition, the geometric deep learning surrogate model was able to simulate wind farms within seconds compared to many CPU hours of parallel computing which might be needed for a RANS simulation.

The remainder of this article is organised as follows: A brief overview of closely related research works is given in Section 2. The numerical CFD model used to simulate wind turbine wakes for training data generation is introduced in Section 3, along with model validation studies against experimental data. The proposed graph representation learning based surrogate for wind turbine wake modelling is detailed in Section 4. In addition to the preliminaries of graph neural networks, a series of training experiments on graph neural network architectures and a case study on the Lillgrund offshore wind farm are also presented. Conclusions and potential future plans for this work are detailed in Section 5.

2. Related work

This work is related to various previous works across different disciplines, such as machine learning, deep learning, general fluid dynamics as well as the specific application of wind turbine wake modelling. This section provides a brief overview of these diverse connections.

Machine learning and CFD

With the rapid advancements in computational power and an explosion of available data from experiments, simulations and historical records, machine learning has seen an increasing level of success in many scientific disciplines, including in computational fluid dynamics. For example, Thuerey et al. [7] explored the use of U-Net convolutional neural network (CNN) architecture to predict velocity and pressure around airfoils of different shapes based on RANS solutions. Guo et al. [8] trained CNNs to predict steady laminar flow past a range of 2D and 3D objects. More recently, geometric deep learning or deep learning on graphs has seen some limited applications in CFD. Belbute-Peres et al. [9] embedded a differentiable PDE solver as an implicit layer in a graph convolutional network framework and trained an end-to-end deep learning model with improved generalisation capabilities. Pfaff et al. [10] developed a graph neural network based architecture for learning time-dependent physical simulations on meshes. Lino et al. [11] further considered message-passing at multiple scales of resolution and incorporated rotational equivariance into the graph neural network model. Suk et al. [12] trained gauge equivariant mesh convolutional networks to predict wall shear stresses over surface meshes that represented artery models. The work presented here builds on this area of research by modelling a large-scale, 3D CFD dataset on meshes with deep graph neural networks.

Deep learning based surrogate modelling of wind turbine wakes

Despite the enormous potential for data-driven machine learning based wake models, there have been relatively few studies on the surrogate modelling of wind turbine wakes with machine learning methods. Ti et al. [13] used machine learning methods to predict turbine wake fields by interpolating actuator disk with rotation (ADM-R) RANS-based simulation data onto a uniform grid, before splitting it into 2000 partitions and training a multilayer perceptron (MLP) on each partition. Li et al. [14] developed a 2D dynamic wake model using bilateral CNN trained on high-fidelity LES data. Zhang et al. [15] also successfully trained a convolutional conditional generative adversarial neural network on LES data that could provide accurate real-time 2D wake predictions. The work presented here further contributes to the data-driven wake modelling research area by creating a 3D wake model with state-of-the-art geometric deep learning methods that is capable of predicting both velocity and turbulent kinetic energy (TKE) flow fields while also taking into consideration turbine yaw induced wake steering effects.

3. Numerical model

This section provides a brief overview of the generalised actuator disk method as well as model validation with the "blind test" wind tunnel experiments performed at the Norwegian University of Science and Technology.

3.1. Generalised actuator disk (GAD) model

The GAD model parameterises the wind turbine rotor as a virtual permeable disk, where the rotor blades are divided in the radial direction into various sections. The lift and drag forces exerted on the rotor blades by the fluid flow are computed with blade element momentum (BEM) theory on each control volume, summed up and then multiplied by the number of blades to be incorporated into the Navier–Stokes momentum equation as an additional source term. The forces acting

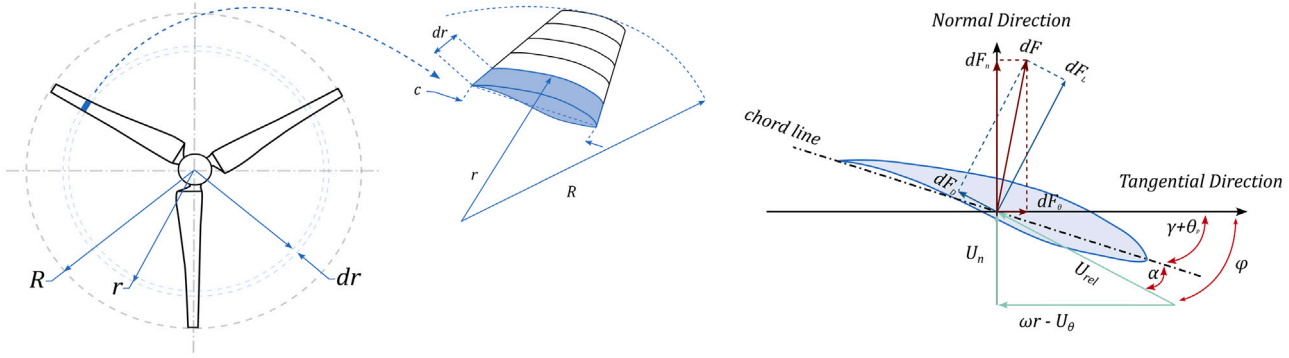


Fig. 1. Blade element momentum discretisation of the rotor blades and the forces exerted on a single blade. R stands for wind turbine radius and r represents distance from a blade element to the centre of the wind turbine in the span-wise direction. The remaining symbols are defined in the main text.

on a blade element with length dr in the span-wise direction can be written as follows:

$$\begin{aligned} dF_D &= \frac{1}{2}\rho c|U_{\text{rel}}|^2 C_D\, dr, \\ dF_L &= \frac{1}{2}\rho c|U_{\text{rel}}|^2 C_L\, dr, \\ dF_n &= dF_L\ \cos\ \varphi + dF_D\ \sin\ \varphi, \\ dF_\theta &= dF_L\ \sin\ \varphi - dF_D\ \cos\ \varphi, \end{aligned} \tag{1}$$

where dF_L, dF_D, dF_n and dF_θ represent respectively the lift, drag, normal and tangential forces on a blade element, U_{rel} denotes the relative wind velocity, c is the chord length which varies in the radial direction and C_L and C_D are the lift and drag coefficients respectively, and are dependent on the local angle of attack α and the Reynolds number Re. φ is the flow inclination angle given by:

$$\varphi = \tan^{-1}\left(\frac{U_n}{\omega r - U_\theta}\right) = \gamma + \theta_p + \alpha, \tag{2}$$

where ω denotes the turbine angular velocity, U_θ and U_n are the tangential and normal velocity, γ, θ_p and α are the blade twist angle, blade pitch angle and angle of attack respectively. The momentum equation source term can be computed as:

$$\begin{aligned} S_i &= S_n + S_\theta, \\ S_n &= N\, dF_n\, \hat{v}_n = \frac{1}{2}\rho N c|U_{\text{rel}}|^2(C_L\ \cos\ \varphi + C_D\ \sin\ \varphi)\, dr\, \hat{v}_n, \\ S_\theta &= N\, dF_\theta\, \hat{v}_\theta = \frac{1}{2}\rho N c|U_{\text{rel}}|^2(C_L\ \sin\ \varphi - C_D\ \cos\ \varphi)\, dr\, \hat{v}_\theta, \end{aligned} \tag{3}$$

where N is the number of turbine blades, $\hat{v}_n$ and $\hat{v}_\theta$ are the unit vector in the normal and tangential directions respectively. The nacelle effect was modelled in a similar way here as a blade element, but with a constant drag coefficient with $C_{D,\text{nacelle}} = 1$ and zero lift ($C_{L,\text{nacelle}} = 0$). An illustration of the GAD discretisation scheme and the forces applied on a single rotor blade is shown in Fig. 1.

The thrust, torque and power can then be calculated by integrating the forces over the virtual disk as follows:

$$\begin{aligned} T &= \sum_i \rho V_i S_{n,i}, \\ Q &= \sum_i \rho V_i r_i S_{\theta,i}, \\ P &= \sum_i \rho V_i r_i \omega S_{\theta,i}, \end{aligned} \tag{4}$$

where T, Q, and P represent thrust, torque and power respectively, i is the cell index of the list of cells that lie within the discretised actuator disk region, and V_i is the cell volume associated with cell i. The numerical modelling framework was implemented here in the *OpenFOAM* CFD package [16] using a modified version of the *GAD-CFD* code developed in [17] and coupled with the steady-state *SimpleFoam* solver. In-depth theoretical background and implementation details of the GAD model can be found in numerous recent studies [18–20].

3.2. Model validation

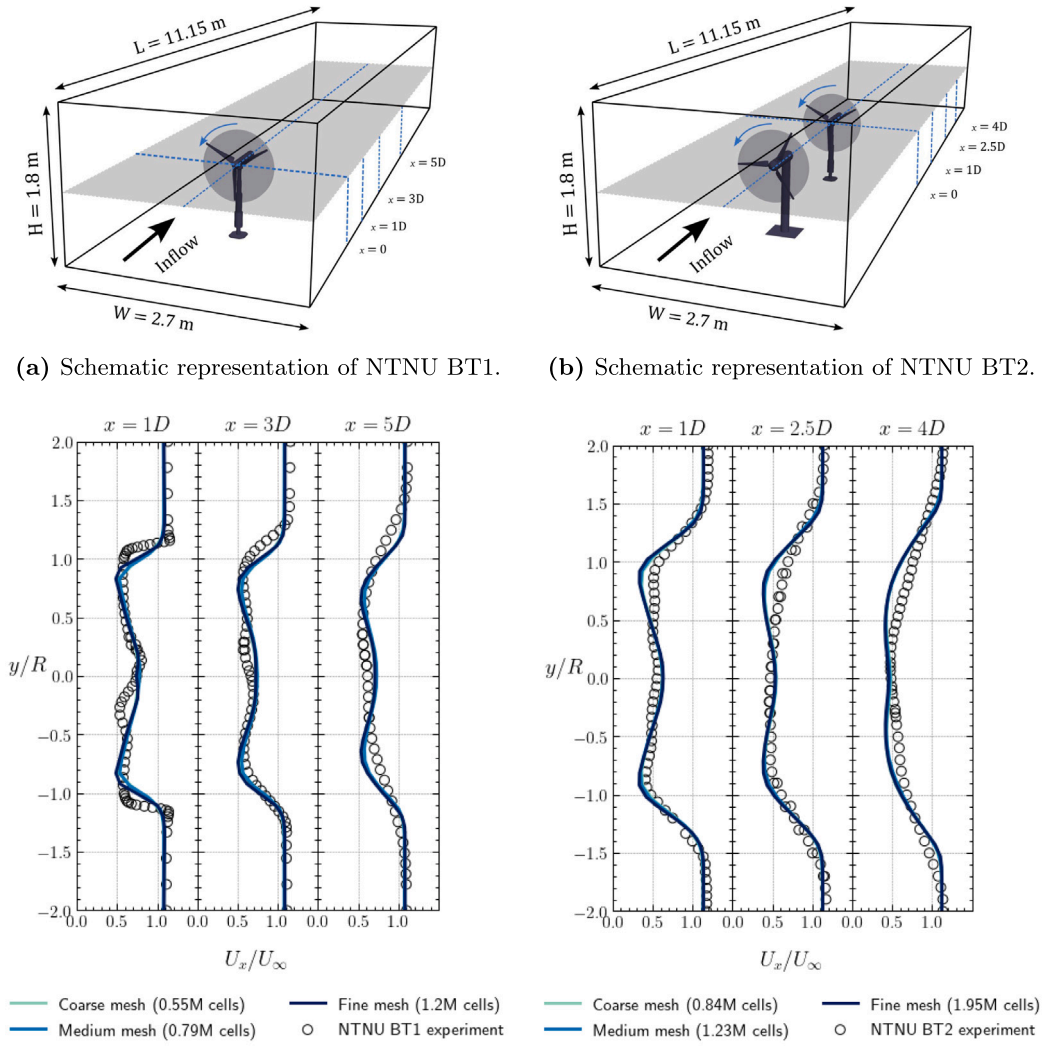
The GAD model has been extensively validated against various experimental studies [17]; for instance, against a series of tidal turbine experiments at the French Research Institute for Exploitation of the Sea (IFREMER) by Mycek et al. [21], experimental work of Selig et al. at the National Renewable Energy Laboratory (NREL) with a full-scale NREL phase III horizontal axis wind turbine [22], as well as a large-scale experiment with multiple tidal turbines at the FloWave ocean energy research facility [23]. This work further validated it against wind tunnel experiments conducted by the Norwegian University of Science and Technology (NTNU) [24,25], referred to as NTNU "blind test" (BT1) and "blind test 2" (BT2).

Both blind test experiments were conducted in a wind tunnel that was approximately 2.7 m wide, 1.8 m high and 11.15 m long. In BT1, a single model wind turbine with diameter $D = 0.894$ m was placed at a distance of 3.66 m from the wind tunnel inlet, whereas in BT2 two similar wind turbines with the same hub height but slightly different diameters ($D_1 = 0.894$ m and $D_2 = 0.944$ m, where the subscripts 1 and 2 correspond to the upstream and downstream turbines respectively) were mounted along the wind tunnel center line at a separation distance of $3D_1$, the upstream turbine was located at $2D_1$ from the wind tunnel inlet. Both experiments had the same operating conditions with a free stream wind speed of $U_\infty = 10\,\text{m s}^{-1}$ and turbulence intensity of $\mathcal{I} = 0.3\%$, and all wind turbines had three bladed rotors with the same blade geometry that used the NREL S826 airfoil. The turbine in BT1 had a design tip speed ratio (TSR) of 6, and wake statistics were measured at locations $x = 1$, 3, and $5D$ downstream of the turbine rotor. BT2 had the upstream turbine operating at $\text{TSR}_1 = 6$ and the downstream turbine at $\text{TSR}_2 = 4$, with wake statistics recorded at $x = 1$, 2.5 and $4D$ behind the downstream turbine rotor. Schematic representations of the two experiments are shown in Figs. 2(a) and 2(b).

GAD simulation results with the realisable $k-\varepsilon$ turbulence model were compared against experimental data from NTNU, shown in Figs. 2(c) and 2(d). The choice of realisable $k-\varepsilon$ turbulence model was due to its better fit with experimental data compared to the standard $k-\varepsilon$ model, as is also reported in various other studies [26,27]. The simulations were run with different mesh resolutions to check for mesh independence of the obtained solutions. An overall good agreement between GAD simulations and NTNU experimental data was found, which further demonstrated the ability of the GAD framework to model individual and multiple turbine interactions.

4. Graph representation learning

Many physical, biological and social systems can be described, at a certain level of abstraction, as graphs. Graph neural networks (GNNs) are powerful tools that are utilised to learn graph representations of complex, non-Euclidean data such as manifolds, gauges and



(a) Schematic representation of NTNU BT1.

(b) Schematic representation of NTNU BT2.

(c) Horizontal wake profile comparison – BT1.

(d) Horizontal wake profile comparison – BT2.

Fig. 2. Schematic representation of the two setups of NTNU blind tests and the comparison of CFD simulation and experimental horizontal wake profiles.

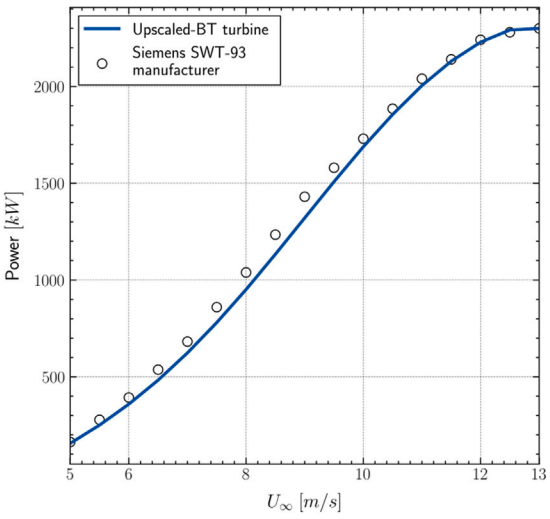


Fig. 3. Power curve of the up-scaled "blind test" turbine compared to the manufacturer power curve of the Siemens SWT93-2.3 MW wind turbine.

meshes [28]. The ability of GNNs to capture the patterns and intricate relationships present in graphs has made them increasingly popular across a wide variety of fields of research, most notably in social networks where GNNs are used to model user-to-item interaction and predict social relations between users [29], in chemistry and drug discovery to predict properties of chemical bonds and entire molecules [30,31], in transportation systems to forecast road traffic and passenger flow in public rail transit systems [32]. This work investigates the utilisation of GNNs as surrogate models of CFD simulations, where a finite computational mesh representing the spatial domain can be naturally defined as a graph. Consequently GNNs have great compatibility with CFD simulation data and can operate directly on unstructured meshes. Importantly, this eliminates the need for interpolation onto uniform grids, which is required for most CNN networks. This section details the graph neural network framework developed here for wake modelling, including training data generation, converting CFD data to appropriate graph data structures, the fundamentals of graph neural network theory and the proposed model architecture as well as relevant training experiments. The models were implemented using the open-source deep learning library PyTorch [33] and PyTorch Geometric [34], which is a geometric deep learning library based on PyTorch.

In order to be able to compare the geometric deep learning approach with a real world scenario, this work considered Sweden's Lillgrund

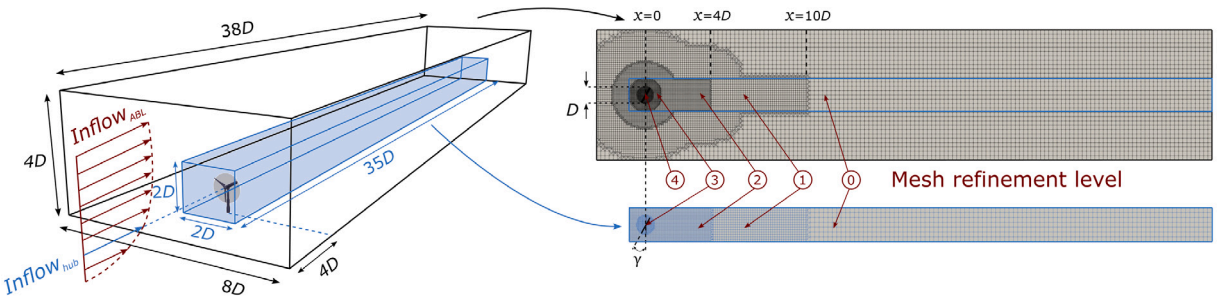


Fig. 4. 3D representation and 2D mesh of a horizontal slice (at hub height) of the computational domain for CFD simulations. Training data was extracted from the area shaded in blue.

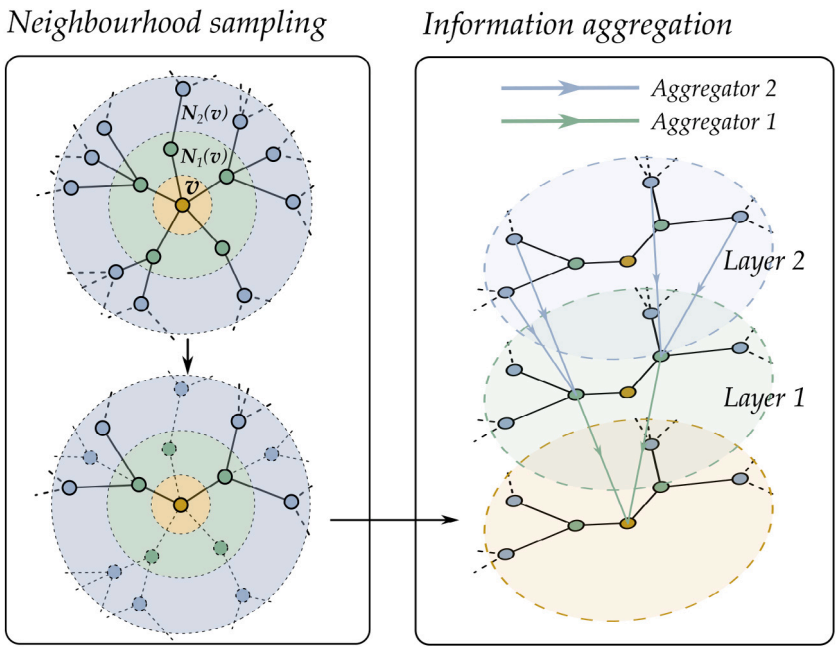


Fig. 5. Visual illustration of the GraphSAGE sample and aggregation approach in a two-layer case for a target vertex v. $N_1(v)$ and $N_2(v)$ represent the one-hop and two-hop neighbourhoods of v, respectively.

offshore wind farm as a benchmark problem. Training data was generated based on CFD simulations under various operating conditions (inlet velocity, turbulence intensity and turbine yaw angle) of a standalone Siemens SWT93-2.3 MW wind turbine, which are deployed in the Lillgrund farm. CFD simulations were also performed on different rows of the Lillgrund wind farm to compare against superimposed single wake fields produced from the deep learning model.

4.1. Data generation

Training data was generated based on simulations of the Siemens SWT93-2.3 MW wind turbine, which had a turbine rotor with radius $R = 46.5$ m and hub height $H_{\text{hub}} = 65$ m. The exact blade geometry and airfoil characteristics have not, to the best of our knowledge, been disclosed to the public. As a consequence, an up-scaled version of the "blind test" turbine chord profile and airfoil characteristics were used, in a similar approach to [35]. This choice could be partially justified by the good agreement between the power curve for the up-scaled "blind test" turbine and the Siemens SWT93-2.3 MW turbine according to its manufacturer [36], as shown in Fig. 3.

Training data was generated by varying the inflow velocity $U_{\infty,\text{hub}}$ and turbulent intensity $I_{\infty,\text{hub}}$ at hub height as well as turbine yaw angle γ, the ABL boundary condition varied for each run with $U_{\infty,\text{hub}}$ and $\text{TKE}_{\infty,\text{hub}} = \frac{3}{2} I^2_{\infty,\text{hub}} U^2_{\infty,\text{hub}}$ set as the reference velocity and TKE, and hub height set as reference height. The simulation domain was set to be sufficiently wide and high in order not to have an impact on simulation results, based upon sensitivity testing, and long enough to cover the longest row of the Lillgrund wind farm. For the deep learning task, training data was extracted from a smaller section of the simulation domain as shown in Fig. 4. The motivation for this is that it would be sufficient for the deep learning model to learn directly from the reference domain that was embedded within a simulation of a larger domain, and that it is unnecessary for the deep learning model to be able make predictions on the entire simulation domain, but rather to concentrate on the area close to and behind the wind turbine where other turbines will likely be located. The size of the training data domain was tested to be large enough to capture the entirety of the wake structure, including in the yawed cases. The computational mesh for both running CFD simulations and training deep learning models had several levels of resolution across the domain. As detailed in Fig. 4, the mesh was densest in areas within and around the wind turbine actuator disk, and gradually coarsened at distance from these. A spherical refinement region with diameter $1D$ was placed at the location of the wind turbine so that the same mesh could be used for simulations with different turbine yaw angles. The CFD simulation mesh had 0.6 million cells and a narrower area of 0.1 million cells was clipped from the simulation domain to be used as training data. Data from a total of 7700 simulations was generated, with the inflow velocity ranging from 5 m/s to 10 m/s at 0.56 m/s interval, turbulence intensity from 5% to 15% at 0.56% interval and yaw angle from −30° to 30° at

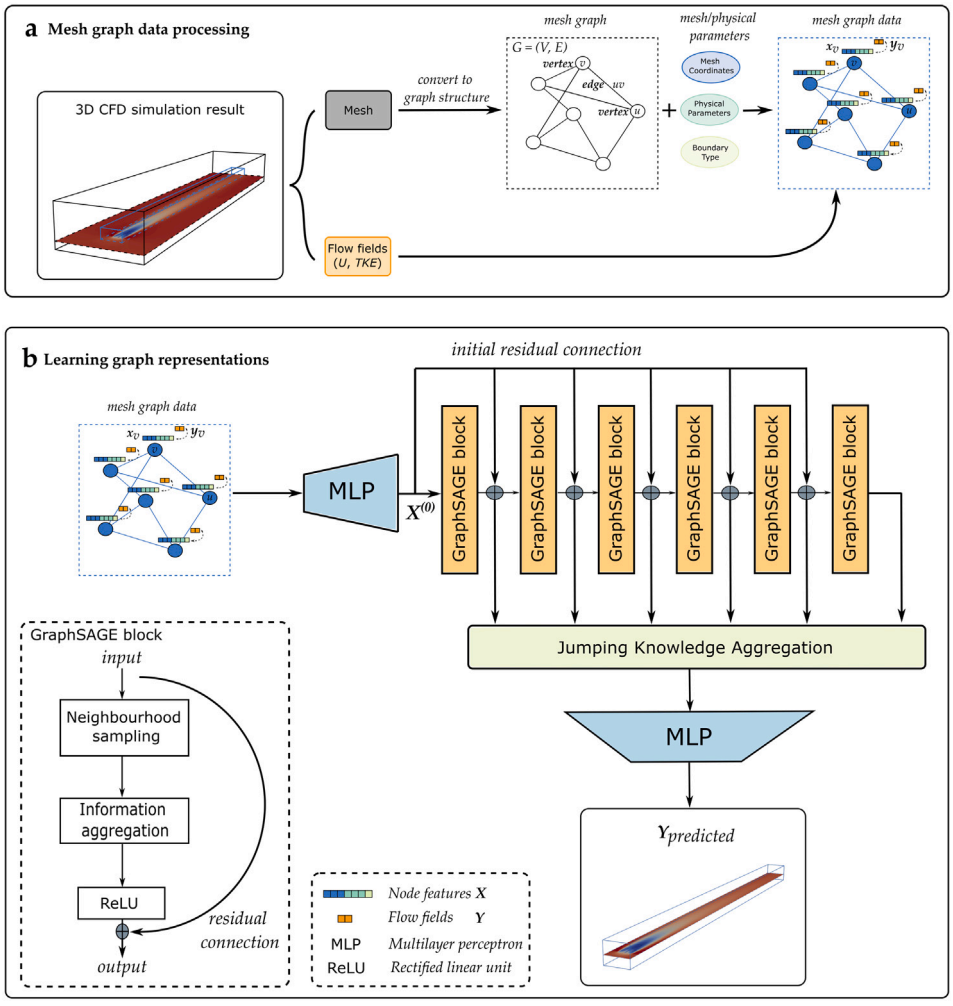


Fig. 6. The overall workflow for converting CFD simulation data to mesh graph data and the GraphSAGE graph neural network architecture with jumping knowledge and residual connections.

1.25° interval. The 7700 simulation data took about 4000 CPU hours to generate on Imperial College London's CX1 HPC cluster, with each simulation taking on average approximately 0.5 h for the SimpleFoam steady-state solver to converge, and was partitioned into sets of size 6200/750/750 for training, validation and testing respectively.

4.2. Graph neural networks and model architecture

This work utilised graph neural networks for a supervised graph representation learning task, and took as input a graph $G = (V, E)$, where $V \in \mathbb{R}^{n \times f}$ represents the n vertices with f features on each vertex, and $E \in \mathbb{R}^{2 \times n_e}$ stands for the n_e number of edges, represented via the pairs of vertices that form each of them. The objective is to train the graph neural network to complete a vertex-level prediction task, which requires the GNN to learn a representation of each graph vertex based solely on its own features and those of its neighbours. A forward pass in a GNN trained in this manner is able to return an updated graph with the same graph connectivity and updated vertex features. The updated vertex features can then be used to make predictions on vertex level targets.

This work made extensive use of the GraphSAGE framework due to its strong inductive learning capabilities and ability to scale very well on larger graphs [37,38]. The GraphSAGE neural network with mean aggregation was used, the vertex-wise update rule of the kth layer of the GraphSAGE network with mean aggregation on a vertex embedding $\mathbf{x}_v$ can be written as:

$$\begin{aligned} \mathbf{x}_v^{(k)} &= \sigma\left(\mathbf{W}^{(k)}\left[\mathbf{x}_v^{(k-1)} \oplus \mathbf{x}_{\mathcal{N}(v)}^{(k)}\right]\right), \\ \mathbf{x}_{\mathcal{N}(v)}^{(k)} &= \frac{1}{|\mathcal{N}(v)|}\sum_{u \in \mathcal{N}(v)} \mathbf{x}_u^{(k-1)}, \end{aligned} \tag{5}$$

where $\mathcal{N}(v)$ is a fixed-sized sampled neighbourhood of vertex v used to aggregate information, and not the full neighbourhood, $\sigma(\cdot)$ is a non-linear activation function, $\mathbf{W}^{(k)}$ is the weight matrix of kth layer, and $\oplus$ stands for concatenation. An illustration of the GraphSAGE network is shown in Fig. 5.

Various other modern GNN architectures, most notably the standard graph convolutional network (GCN) [39] and graph attention network (GAT) [40], were also explored in this work as potential alternatives to GraphSAGE. The GCN update rule can be written as:

$$\mathbf{x}_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{|\mathcal{N}(u)| \cdot |\mathcal{N}(v)|}} \mathbf{W}^{(k)} \mathbf{x}_u^{(k-1)}\right), \tag{6}$$

whereas the update rule for GAT is:

$$\mathbf{x}_v^{(k)} = \oplus_{h=1}^{H} \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}^{(h)} \mathbf{W}^{(h)} \mathbf{x}_u^{(k-1)}\right), \tag{7}$$

where H is the number of attention heads and α_{uv} represents the attention coefficient. GAT is a powerful model that uses the attention mechanism to allow for the implicit assignment of different levels

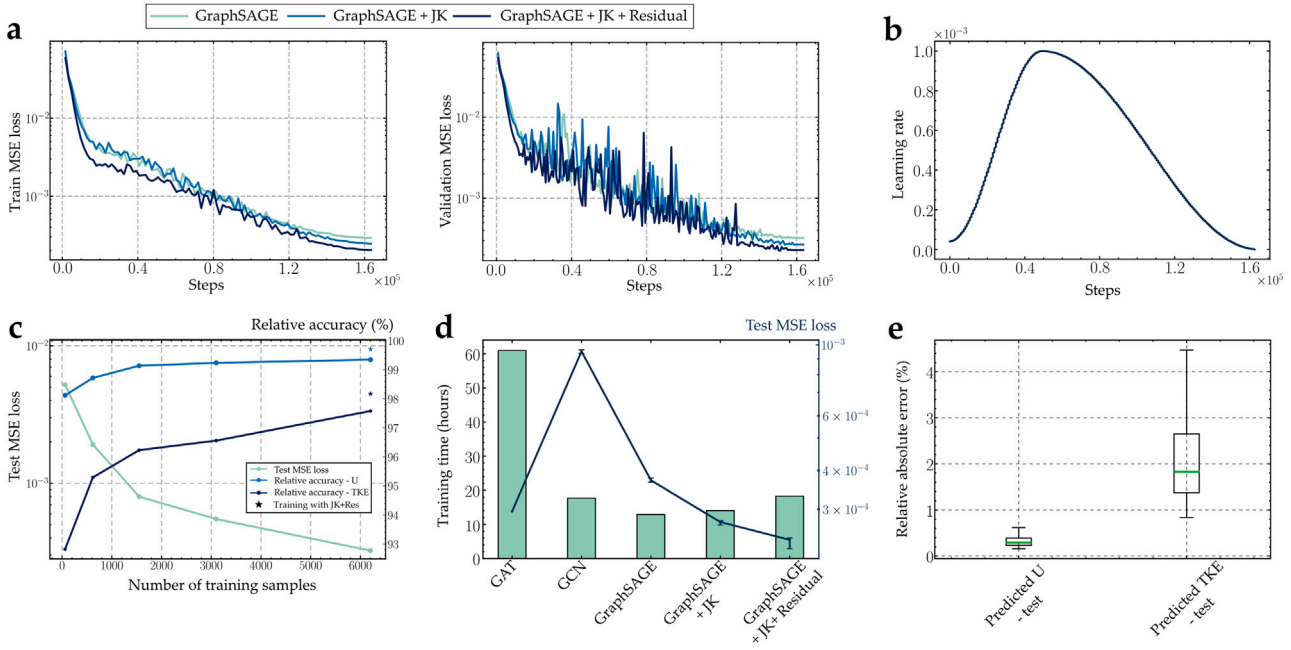


Fig. 7. Supervised training experiments on GNN architectures. **a**: Training and validation learning curves for the three GraphSAGE variants. **b**: The one-cycle learning rate scheme — learning rate was set to start at a small value and climbs up to reach a pre-defined maximum value before gradually decreasing. **c**: Test MSE loss and relative accuracy of predictions made by the standard GraphSAGE model at different training sample sizes. Stars represent prediction accuracy of the GraphSAGE+JK+Res model. **d**: Training time and MSE loss on the unseen test set for different graph neural network models. Training time and test MSE loss for GAT were taken from a single run due to the excessive amount of time needed for training, while for other models training time was averaged over three runs and test MSE loss was reported with error bars. **e**: Box-and-whisker plots of relative absolute error of the final model at predicting U and TKE flow fields on the held-out test data set. The whiskers represent 1.5 times the inter-quartile range (IQR). Green line represents median accuracy on test data.

of importance to a vertex's neighbours, therefore leading to a sizeable increase in model capacity [40]. It should be noted that while GraphSAGE performs layer-wise sampling from the neighbourhood of each vertex, other GNN methods including GCN and GAT, use the full neighbourhood of vertices.

One limitation of deep graph neural networks is over-smoothing, which refers to similarity of vertex representations after several iterations of message passing. This can occur when more layers are added to the structure, as shown in Fig. 5, and eventually every vertex in the graph is able to aggregate information from distant neighbours therefore generating similar graph embeddings. Indeed, various modern GNN models including GCN and GAT achieved their best performance on benchmark problems with models that had only two layers. In particular, CFD data on meshes could have drastically different resolutions across the simulation domain, as a consequence vertices might also need to aggregate information from neighbours of different distances depending on their local spatial mesh resolution. The reason being that vertices in an area of lower mesh density might be closer to each other in graph space but very far apart in mesh space, and similarly vertices in regions with denser meshes might be far from each other in graph space, but very close in mesh space. In order to alleviate over-smoothing, this work adopted jumping knowledge connections (JK) [41,42] which uses dense skip-connections to enable the adaptive learning of structure-aware vertex representations, as well as layer-wise residual connections similar to ResNet [43] and an additional initial residual connection inspired from GCNII [44].

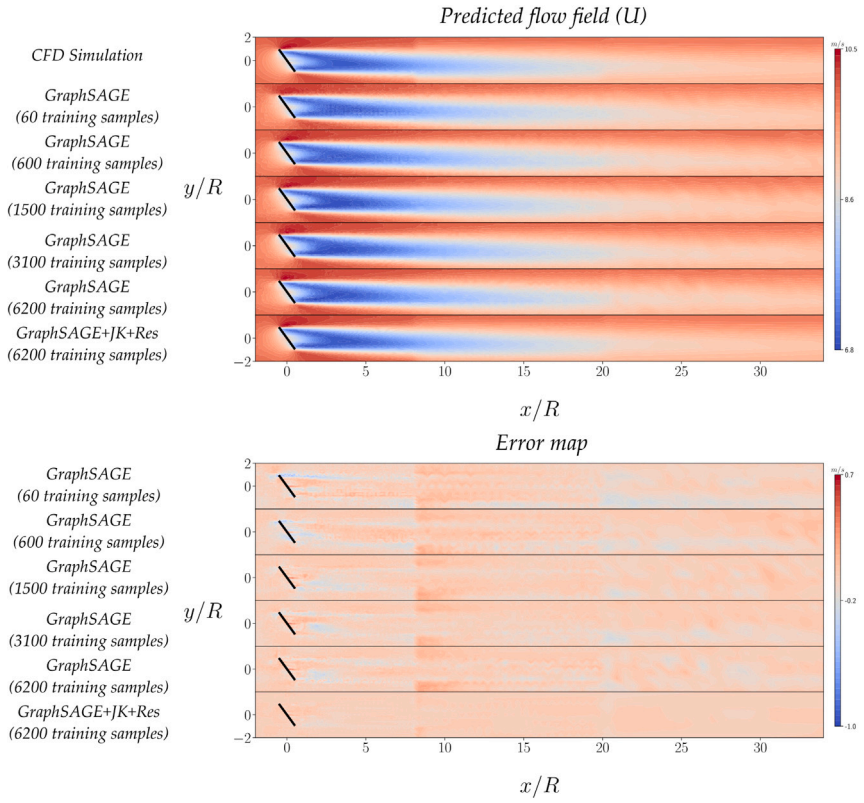
Together with residual connections, the vertex-wise update rule of the weighted GraphSAGE network with mean aggregation can be written as:

$$\mathbf{x}_v^{(k)} = \sigma\left(\mathbf{W}^{(k)}\left[\mathbf{x}_v^{(k-1)} \oplus \frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)}\mathbf{x}_u^{(k-1)}\right]\right) + \alpha\mathbf{x}_v^{(0)} + \beta\mathbf{x}_v^{(k-1)}, \tag{8}$$

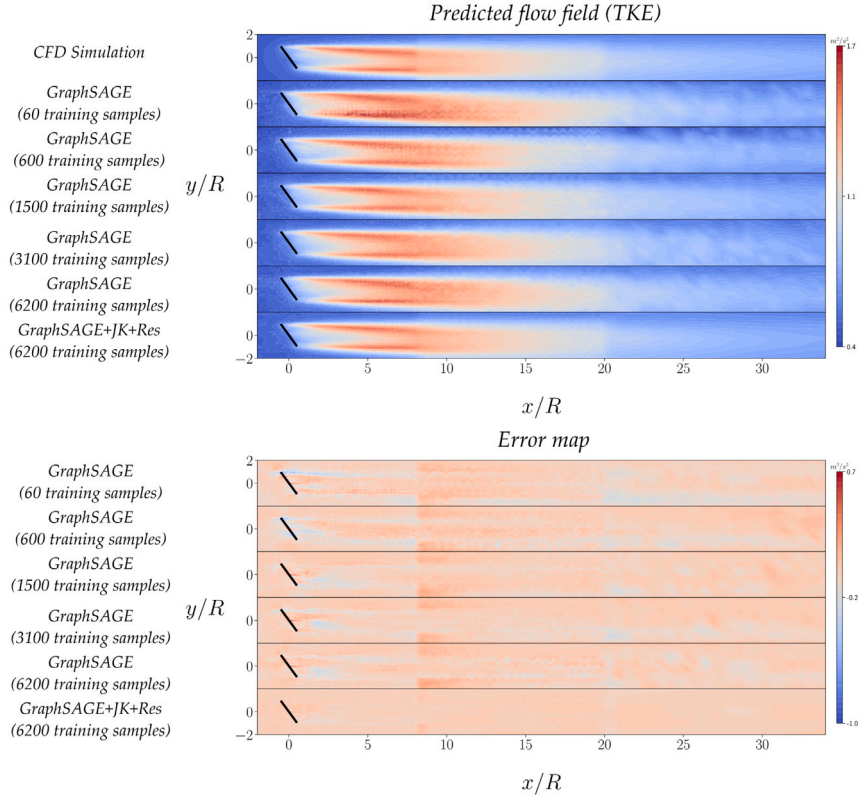
where α and β are hyper-parameters that signify scales of residual representations from previous layers, and were set to 0.1 and 0.9 respectively in this work.

The overall workflow of the mesh graph data processing and network architecture of GraphSAGE with jumping knowledge and residual connections (GraphSAGE+JK+Res) is illustrated in Fig. 6. Training data for the graph network was prepared first by converting the OpenFOAM based meshes and simulation results into a series of unweighted and undirected graphs. The computational mesh directly defines the matrix of edge indices E and can be converted to a graph structure by appending the (x, y, z) coordinates of the mesh vertices as well as their corresponding one-hot encoded boundary types as vertex features in V. Additionally the physical parameters that define the flow structure and that vary among each simulation (i.e. inlet velocity U_∞, turbulence intensity I_∞ and turbine yaw angle γ) are also appended to each vertex in V as global features, so that the network can differentiate among simulation training data and learn to make predictions based on various input physical parameters. Target flow fields were associated with each vertex as output responses.

A single forward pass of the proposed architecture can be described as follows: the input mesh graph data is first encoded by a multi-layer perceptron (MLP), which maps the input vertex level features onto a high-dimensional latent space. The output from the MLP is then processed through a series of GraphSAGE blocks which consist of GraphSAGE updates and ReLU activation functions. The utilisation of initial and layer-wise residual connections means that a fraction of the output $X^{(0)}$ from the MLP is added to the outputs of all subsequent GraphSAGE blocks, and that part of the input $X^{(i-1)}$ to each GraphSAGE block is also added to its output $X^{(i)}$. The output from each GraphSAGE block is concatenated and sent through a jumping knowledge layer which adaptively determines the importance of outputs from each graph convolutional layer, and the final processed output is decoded by another MLP to make vertex level predictions (U and TKE). Other variants of the proposed architecture such as GCN and GAT can be constructed by replacing the GraphSAGE layer in Fig. 6 with the corresponding graph convolutional layers.

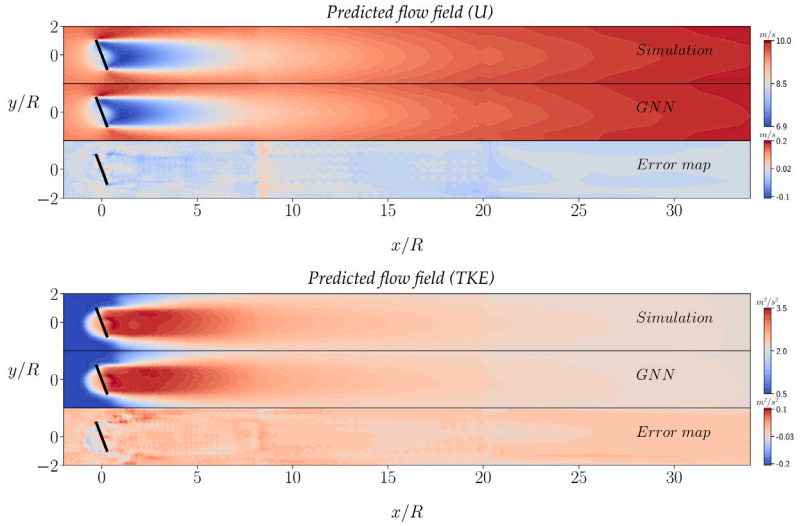


(a) Horizontal slice of predicted velocity magnitude and the corresponding error map when different number of training samples were used.

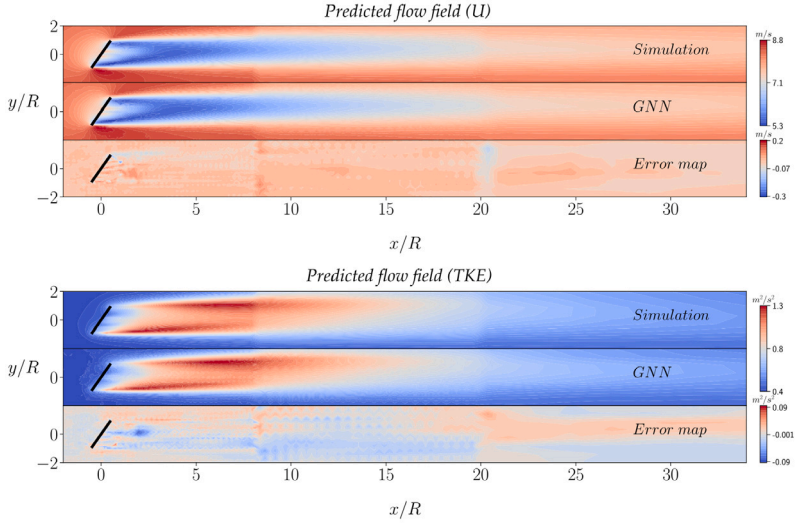


(b) Horizontal slice of predicted TKE and the corresponding error map when different number of training samples were used.

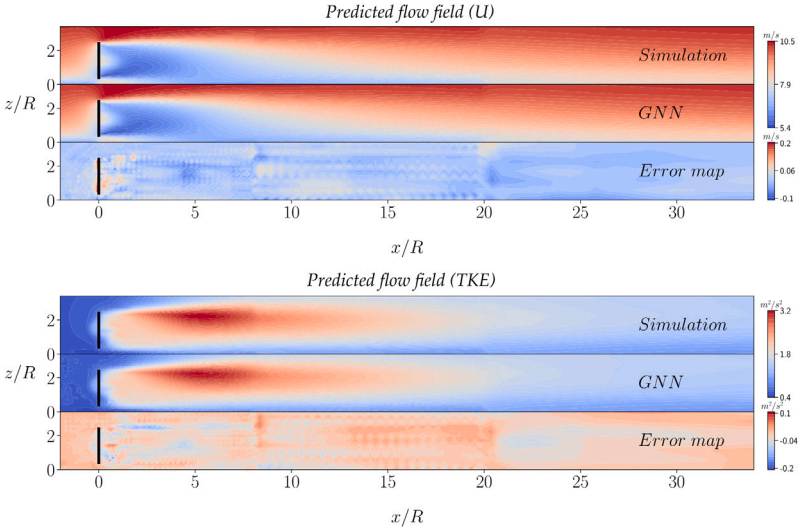
Fig. 8. Illustration of the effect of number of training samples on predictions by the GraphSAGE model and a comparison between GraphSAGE and GraphSAGE+JK+Res model predictions. The case parameters used were $U_\infty = 10$ m/s, $I_\infty = 7.1\%$, $\gamma = -28°$.



(a) Case 1 parameters: U_∞=9.4 m/s, I_∞=13.9%, γ=−16°, horizontal slice shown.



(b) Case 2 parameters: U_∞=8.3 m/s, I_∞=7.6%, γ=28°, horizontal slice shown.



(c) Case 3 parameters: U_∞=9.4 m/s, I_∞=9.2%, γ=−4°, vertical slice shown.

Fig. 9. Comparison between CFD simulation result and corresponding machine learning predictions made by the GraphSAGE+JK+Residual model illustrated in the form of 2D horizontal and vertical slices. Machine learning predictions were made on simulation parameters that were unseen during training. The top rows show the flow fields (velocity magnitude U and TKE) computed from CFD simulation, the middle rows show the predictions made by the trained GNN model and the bottom rows display the difference between the CFD and machine learning predictions.

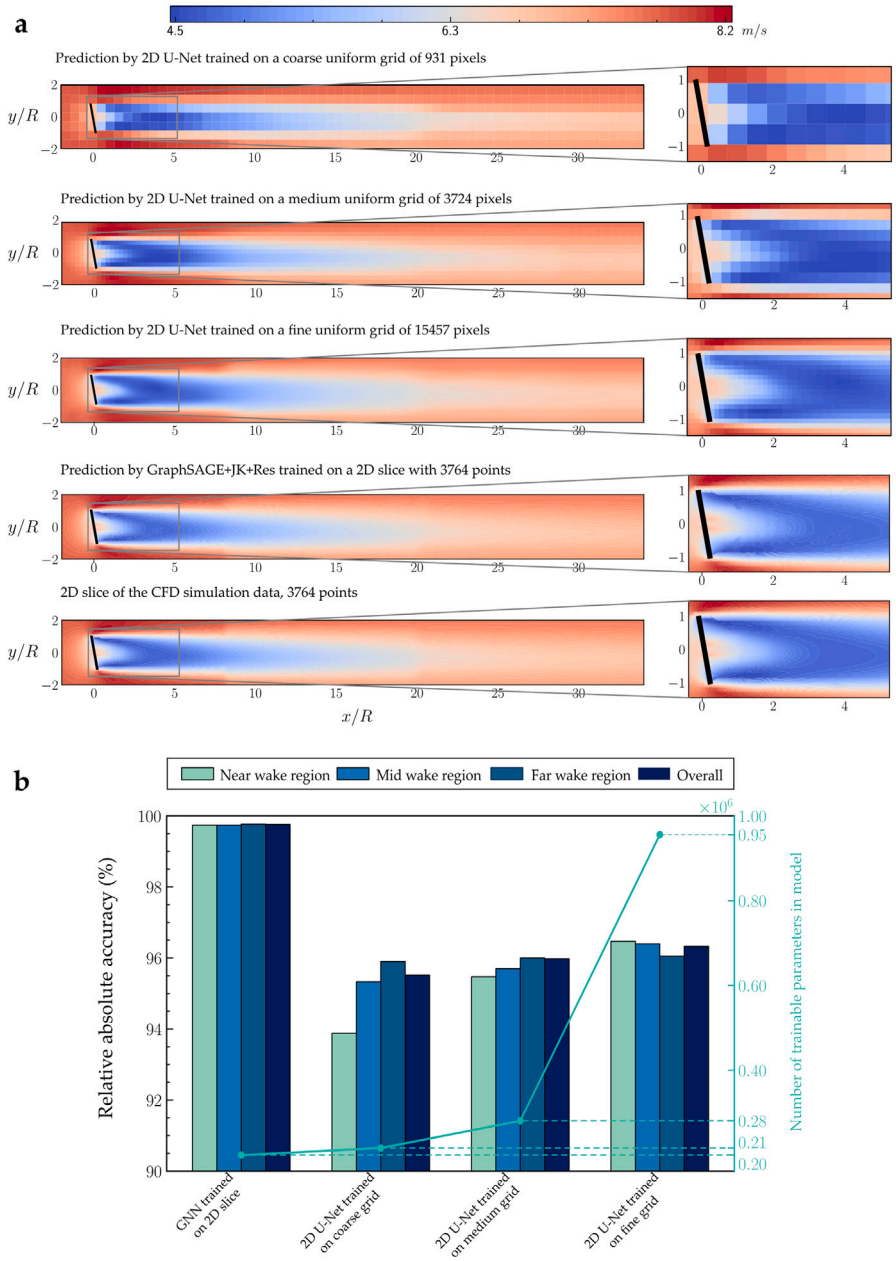


Fig. 10. Comparison of flow field predictions made by CNNs trained on 2D uniform grids of different resolutions and the GraphSAGE+JK+Res model trained on a 2D unstructured slice. **a**: Qualitative comparison of predicted flow fields, including a zoomed in view of parts of the near wake region. Predictions of the models were made with the case parameter: $U_\infty = 7.8$ m/s, $I_\infty = 6.6\%$, $\gamma = -10°$. **b**: Quantitative comparison against CFD data of the accuracy of flow field predictions made by CNNs and the GNN across different areas of the domain as well as the number of trainable parameters contained within each machine learning model.

4.3. Supervised training experiments

Training experiments and ablation studies were performed in an attempt to improve model performance. Unless otherwise specified, all experiments performed in this work were trained to minimise the mean squared error (MSE) of the predicted flow fields, with the AdamW optimiser [45] and the one-cycle [46] learning rate schedule for 160k steps. With the maximum learning rate set to 10^{-3} the one-cycle learning rate schedule yielded very good training convergence with all models and ensured that different training runs were comparable; the learning rate scheme is illustrated in Fig. 7b. A batch size of one was used due to limited GPU memory, gradients were accumulated over 16 mini-batches in compensation. 16 bit automatic mixed precision (AMP) training was used to speed up training and save GPU memory, validation error was checked twice per epoch and models were checkpointed each time upon reaching a lower validation error. All experiments were run on a single NVIDIA GeForce RTX 2080 GPU with 8 GB of memory three times with different random seeds, and when comparing different models, all models comprised of six GNN layers with 128 hidden units in each layer, with a total of 220k trainable parameters.

The core results from the experiments are shown in Fig. 7. Fig. 7a shows an ablation study of the three variants of the GraphSAGE model, and demonstrates that jumping knowledge and residual connections led to improved model performance, as the GraphSAGE model managed to reach lower training and validation loss after adding a JK aggregation layer and residual connections. The amount of data required for model training was investigated and the relevant findings are reported in Fig. 7c. The amount of training data utilised was varied from 60 samples to 6200 samples, and the performance of the GraphSAGE

model was reported after training with samples of different sizes for a fixed number of 160k steps. It can be observed that the graph neural network is able to achieve relatively low MSE error and good prediction accuracy even when trained with a small fraction of the full training data. The number of training samples has a larger impact on the prediction of TKE than the velocity flow field. A generally good level of accuracy could be achieved with 1500 training samples; however, the performance of the GNN surrogate continued to increase as more training data are learned by the model. The effect of number of training samples on model performance is further illustrated in Fig. 8, where flow field predictions from models trained with different numbers of training samples were compared for a given unseen test case. Noticeably even with a relatively small number of training samples the standard GraphSAGE model is able to correctly capture the majority of the primary features of the fluid flow. The near-wake region has the densest mesh and has the most impact on the model performance criterion, and machine learning predictions (particularly around the near-wake area) became more accurate as more training samples were included. It is also manifest that the utilisation of jumping knowledge and residual connections led to markedly better predictions around regions of changing mesh resolution. It should be noted that while this work kept the mesh in the near wake regions dense in order to preserve the entirety of the fluid flow structures, it is possible to customise the mesh used during training, and to further refine or coarsen the mesh in areas that are of more or less significance. An exploration of fully exploiting the combined flexibility of both unstructured mesh CFD and GNNs will be explored in future work.

All GraphSAGE variants proved to scale very well with the large data sizes used in this work, with the GraphSAGE+JK+Residual model taking approximately the same time to train as the standard GCN model, while achieving a much lower test loss. In contrast the GAT model, despite managing a lower test loss than the standard GCN and GraphSAGE model, took considerably longer time to train, as shown in Fig. 7d. The GraphSAGE+JK+Residual model was able to attain better accuracy than GAT, while taking only $1/3$ of the training time. It is worth noting that jumping knowledge and residual connection could also be used with GAT to further improve performance, but the training time is prohibitive, and the increase in training time could be further exacerbated when larger data sizes are considered. The scalability advantages of the GraphSAGE variants are expected to become even more prominent as the size and scale of training data grow. Fig. 7e shows the performance of the GraphSAGE+JK+Residual model on the unseen test dataset. The model was trained more extensively for 320k steps, and achieved a median relative absolute accuracy of 99.71% in predicting U and 98.17% in predicting TKE. The model was able to generalise well to unseen data and the vast majority of the predictions on the test dataset had relative absolute error of less than 5%. This model was considered the final trained model and several examples of its predictions are shown in Fig. 9. The final model had 220k parameters and can make predictions in 13.2 ms ± 120 μs on a single GPU (including the time needed to transfer data from the CPU to GPU) and 2.93 s ± 207 ms on an Intel Core i9-9900K 8 core CPU.

4.4. Model comparison with CNNs

Many popular choices of deep learning based surrogate models for wind turbine wakes, or CFD simulations in general, are currently based on convolutional neural networks and their variants. Additional experiments were thus conducted by comparing the prediction results from CNNs with our GNN. In this experiment, new training data was prepared by extracting a 2D horizontal slice of the 3D flow field at the wind turbine hub height. Training data for CNNs was interpolated onto uniform grids at different resolutions, leading to a 133 by 7 coarse grid with 931 pixels, a 266 by 14 medium grid with 3724 pixels and a 533 by 29 fine grid with 15,457 pixels. The coarse, medium and fine resolutions correspond approximately to the level 0, 1, and 2

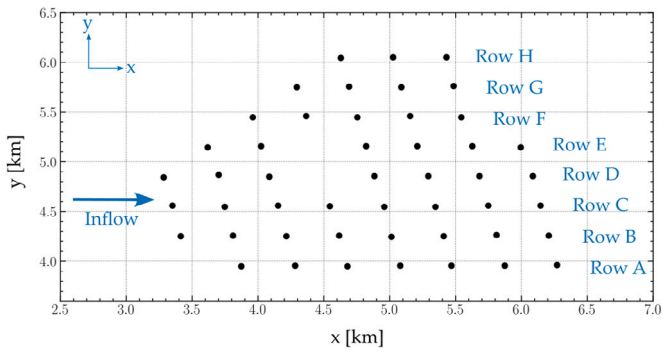


Fig. 11. Lillgrund wind farm layout where the $x-y$ coordinate system were aligned with the southwestern statistically dominant wind direction.

mesh refinements used in the multi-scale CFD simulations, detailed in Fig. 4. The graph neural network model GraphSAGE+JK+Res was also trained on the 2D horizontal slice, which possessed 3764 points. The CNNs all had a U-Net convolutional auto-encoder architecture with the same number of convolutional and de-convolutional layers, a detailed description of the U-Net architecture is given in [47]. As a modern and sophisticated CNN architecture, U-Net and its variants are very commonly used in a wide range of tasks related to computer vision, and also in CFD problems including the prediction of fluid flow with deep learning [7,48,49].

All CNN and GNN models were trained with a constant learning rate of 10^{-3} for a fixed number of 100 epochs. Through experimentation, the optimal U-Net architecture for learning flow field data on the coarse, medium and fine grids were determined to have three, four and five convolutional and de-convolutional layers, respectively. Data on the fine grid was additionally trained with convolutional kernel sizes of five for all layers, compared to kernel sizes of three for data on the coarse and medium grids. Due to the low aspect ratio, all data was resized to be four times longer in the y-axis direction prior to training with CNNs. The same GraphSAGE+JK+Res architecture used earlier in this work for 3D simulation data was used for training a GNN on 2D slices. All models managed to converge during training, and were checkpointed upon reaching the lowest validation loss. An example of the predicted velocity flow fields for each model are shown in Fig. 10a, while prediction accuracy of the models and their number of trainable parameters are shown in Fig. 10b. In particular, prediction accuracy of the models across different regions of the simulation domain was also computed, where the near wake, mid wake and far wake regions refer to areas with different mesh resolution, which correspond to downstream location (x/D) between $[0,4]$, $[4,10]$ and $[10,35]$ respectively.

Comparisons between CNNs and GNNs are complicated by numerous factors. As predictions made on uniform grids and unstructured meshes cannot be quantitatively compared directly, predictions by CNNs on grids of different levels of resolutions and the GNN on unstructured 2D slices were all up-sampled to a very fine 1066 by 59 uniform grid. CFD simulation results were mapped to the same very fine grid to compute the accuracy of different models in Fig. 10b. It can be observed from Fig. 10 that despite CNNs trained on grids of different levels of resolution having similar levels of overall accuracy, the differences in prediction accuracy in the near and mid wake regions are more significant. Remarkably, the GNN model was able to consistently achieve good accuracy across all regions while keeping both the size of the training data and the number of model parameters at a minimum. It should be noted that through more extensive tuning of hyperparameters or optimisation of model architectures, it is possible for both the CNNs and the GNN to achieve better performance.

When dealing with 3D surrogate modelling specifically, the ability of GNNs to operate directly on unstructured meshes means that for multi-scale or multi-resolution problems GNNs are able to work with

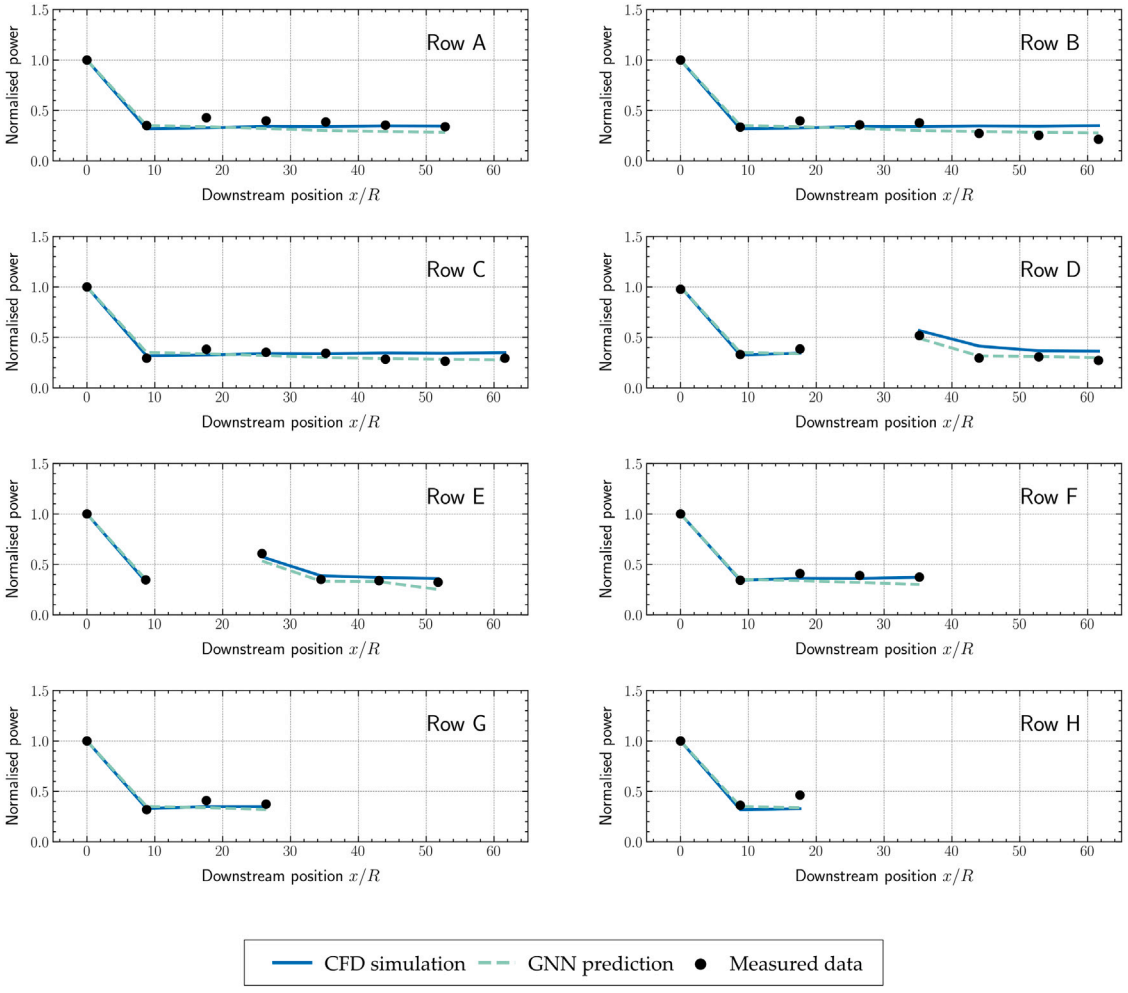


Fig. 12. Comparison of predicted power from CFD simulations and GNN predictions with wake superposition against measured data for each row of the Lillgrund wind farm.

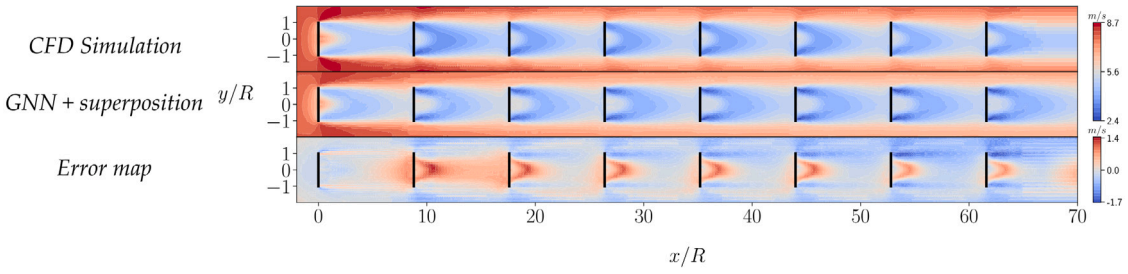


Fig. 13. Velocity flow field of Row B of the Lillgrund wind farm. The top row shows the flow fields computed from a CFD simulation, the middle row shows the SOS superimposed predictions made by the GNN model and the bottom row displays the difference between CFD and machine learning predictions.

large scale training data relatively efficiently, whereas other methods including CNNs that resort to interpolating data onto uniform grids would have to deal with the increase in the size of training data due to interpolation, which can be computationally prohibitively expensive for large scale problems. GNN's compatibility with data that is multi-scale in nature, its ability to more easily adapt to the modelling of 3D flow fields and accurately predict large and complex datasets with relatively small model sizes all make GNN a more suitable choice for the type of surrogate modelling investigated in this work. In particular, training of CNNs on 2D slices of finer resolution or on 3D uniform grids proved to be prohibitively expensive computationally for the dataset considered.

4.5. Model testing on multiple wind turbines

The ability of GNN based wake model to model a real wind farm was tested by using the trained GNN model to predict individual turbine wakes, and superimposing the predicted wakes with a standard wake superposition approach. Specifically, the sum of squares superposition method was used, which has the following form:

$$U_i = \left(1 - \sqrt{\sum_{j=1}^{n_i}(1 - \frac{U_{ij}}{U_{\infty,j}})^2}\right)U_\infty, \tag{9}$$

where U_i is wind velocity at turbine i, U_{ij} refers to the wind velocity at wind turbine i influenced by the wake of wind turbine j, $U_{\infty,j}$ is the

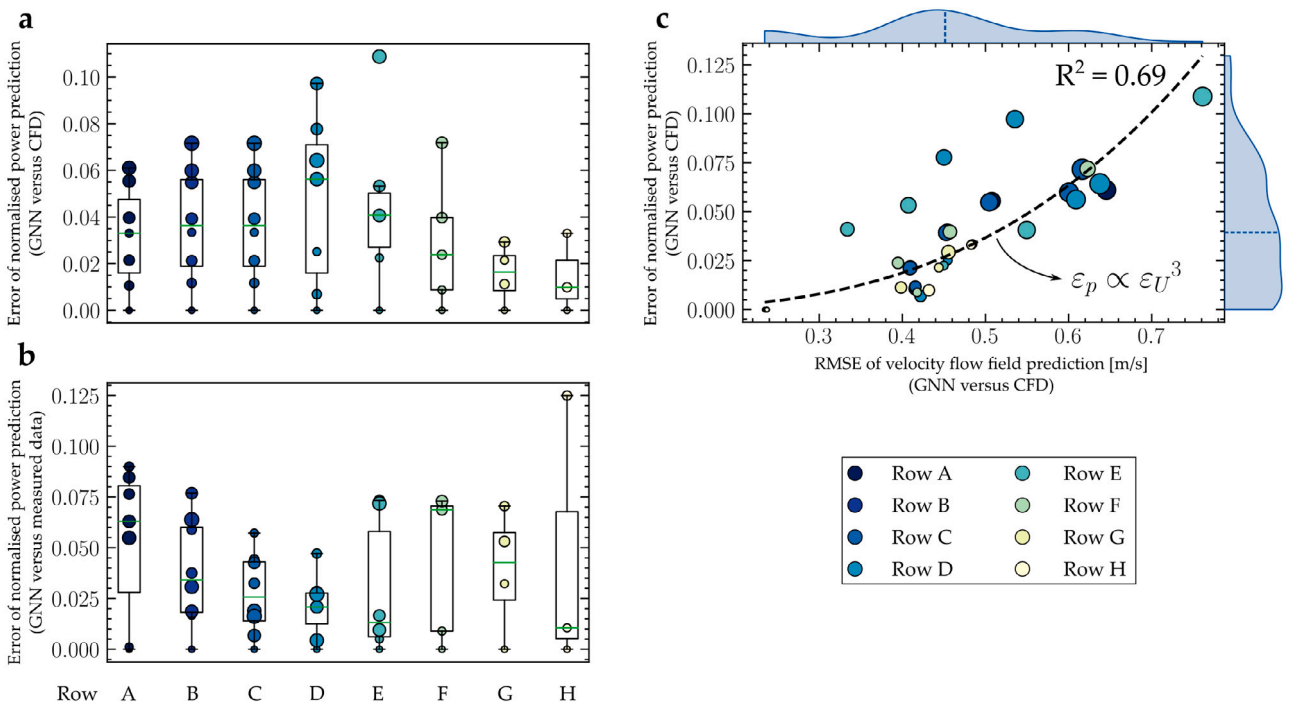


Fig. 14. Analysis of predicted normalised power from CFD simulations and the GNN surrogate against observed data. Dots represent absolute error in power predictions of wind turbines, with wind turbines that are farther downstream represented with larger diameters. **a**: Box-and-whisker plot of errors of predicted power from GNN compared to CFD simulations. Green line represents median prediction error in each wind farm row. **b**: Box-and-whisker plot of errors in predicted power from GNN compared to observed data. **c**: Scatter plot of error in predicted power of turbines by GNN compared to CFD simulation, against the root-mean-square error in velocity flow field predictions around and behind wind turbines (up to $4D$). Fitted regression curve contains only a cubic term. Kernel densities are shown in areas shaded in blue, with the blue dashed lines representing medians of the two distributions.

inlet velocity experienced by turbine j and n_i is the total number of upstream wind turbines. The choice of the optimal superposition method is dependent on the relative positions of the turbines [50]. Other wake superposition methods proved unsuitable for this test case, with the linear and the largest deficit superposition methods underestimating and overestimating the wind velocity respectively.

The wind farm investigated was the Lillgrund offshore wind farm, which consisted of 48 Siemens SWT93-2.3 MW wind turbines distributed in eight rows (A–H). The layout of the wind farm considering the statistically dominant wind direction is shown in Fig. 11. CFD simulations were performed on different rows of the Lillgrund wind farm assuming independence of rows in order to enable fair comparison with the GNN surrogate.

The power generated by each turbine in the GNN based wake model was computed as:

$$P = \frac{1}{2}\rho C_p A U^3 \tag{10}$$

where A stands for the area swept by the wind turbine rotor, and C_p is the power coefficient given by the power curve shown in Fig. 3. Power generated by wind turbines in different rows of Lillgrund from CFD simulations, the GNN based model and measured data [36] are compared in Fig. 12. It can be observed that in general there is good agreement among all three of the CFD computations, GNN predictions and the measured data. The overestimation of the power for wind turbines that are farther downstream in the CFD simulations is in part due to the assumption of independence of rows within the wind farm, a similar phenomenon has also been reported in other studies [35]. In particular, the GNN wake model also managed to accurately capture the predicted power in rows D and E where there was a gap within the turbine rows. Nevertheless, superimposing individual turbine wakes led to a small underestimation in the generated power of turbines located far downstream compared to CFD, as can be observed in most rows of this test case. The velocity flow field predictions from CFD simulation and GNN prediction for row B of Lillgrund is shown in Fig. 13.

A more detailed comparison of the predicted normalised power from CFD simulations and GNN predictions against measured data is shown in Fig. 14. In Fig. 14a and b power predictions by the GNN surrogate are compared against CFD and measured data respectively. The average absolute error of GNN power predictions compared against CFD and measured data are 0.034 and 0.036. However, despite being able to achieve relatively good accuracy in power predictions, GNN predictions have a tendency to accumulate errors, as observed in Fig. 14a where further downstream turbines tend to have larger errors in power prediction. Moreover, there is also a trend for the power prediction error to be slightly larger on the 2nd downstream turbine. Both phenomenons can be attributed to the use of superposition methods when combining individual turbine wakes. The utilisation of wake superposition methods are not without limitations. For instance, linear wake superposition method can predict the power of the second downstream wind turbine in all rows more accurately, but there would be even more severe overestimation of the wake effect further downstream. Using the largest deficit wake superposition, on the other hand, has the opposite effect and would underestimate the wake effect. Future developments related to this work will include data-driven machine learning based methods for combining individual wind turbine wakes to further enhance model performance. Fig. 14c details the root-mean-square error (RMSE) in the velocity flow field prediction around and behind wind turbines in each row of the Lillgrund farm and its relationship with the error in predicted power by the GNN surrogate compared to CFD simulation. Regression analysis showed that a cubic relationship is the most appropriate, which has an R^2 value of 0.69, compared to a linear line ($R^2 = 0.36$) and a quadratic polynomial ($R^2 = 0.47$). This is consistent with the prior belief that with the formulation of power computation given by Eq. (10), the prediction error in power generation (ε_p) should be proportional to the prediction error in the velocity flow field (ε_U) to the third power.

5. Conclusion and future work

This work proposed a deep GraphSAGE neural network with jumping knowledge and residual connections (GraphSAGE+JK+Res) that is flexible and can operate directly on unstructured meshes with varying resolution. As the first attempt to introduce graph representation learning into wind turbine wake modelling, the trained GraphSAGE neural network was capable of accurately learning the complex nonlinear relationship between the inlet conditions and the resulting flow fields and achieved high prediction accuracy on data unseen during training (99.71% accuracy on predicting U and 98.17% on TKE). The proposed model and workflow are highly generic and as currently formulated can be readily applied to any steady state CFD simulation on arbitrary meshes. A case study on Sweden's Lillgrund offshore wind farm was carried out using both the GAD-RANS-based CFD simulation and the proposed machine learning surrogate model. The results showed that the proposed model could accurately predict generated power compared to CFD simulation results as well as real world measured data. The proposed model with the same architecture and hyperparameters could be extended to the modelling of other wind farms with different types of wind turbines with relative ease either through transfer learning with new training data associated with the wind turbine type used, or by training on a combined dataset and treating the type of wind turbine as an additional global graph level feature.

With the ability to make accurate predictions under different inflow conditions and turbine yaw angles in less than 15 ms, the graph neural network approach has the potential to be utilised in wind farm control and optimisation problems including yaw angle based wake steering. Future work could also include using higher fidelity LES based dynamic training data and machine learning based modelling of wake interactions.

CRediT authorship contribution statement

Siyi Li: Conceptualization, Methodology, Software, Formal analysis, Visualisation, Writing – original draft. **Mingrui Zhang:** Conceptualization, Methodology, Writing – review & editing. **Matthew D. Piggott:** Conceptualization, Methodology, Project administration, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

MDP acknowledges funding from the UK Engineering and Physical Sciences Research Council under project EP/R029423/1. The authors would like to thank Dr Xiaorong Li for granting access to the GAD-CFD repository, Professor Per-Åge Krogstad of NTNU for providing access to an electronic version of the "blind test" dataset, and Che Liu and Dr Sibo Cheng for insightful discussions.

References

[1] Piggott Matthew D, Kramer Stephan C, Funke Simon W, Culley David M, Angeloudis Athanasios. Optimization of marine renewable energy systems. In: Comprehensive renewable energy, Vol. 8. 2nd ed.. Elsevier; 2022, p. 176–220.
[2] Katić Ivan, Højstrup Jørgen, Jensen Niels. A simple model for cluster efficiency. In: European wind energy association conference and exhibition. 1987, p. 407–10.
[3] Larsen GC. Simple wake calculation procedure. Technical report, Roskilde: Risoe National Lab.; 1988.
[4] Bastankhah Majid, Porté-Agel Fernando. A new analytical model for wind-turbine wakes. Renew Energy 2014;70:116–23, Special issue on aerodynamics of offshore wind energy systems and wakes.
[5] Annoni J, Fleming P, Scholbrock A, Roadman J, Dana S, Adcock C, Porte-Agel F, Raach S, Haizmann F, Schlipf D. Analysis of control-oriented wake modeling tools using lidar field results. Wind Energy Sci 2018;3(2):819–31.
[6] Gebraad PMO, Teeuwisse FW, van Wingerden JW, Fleming PA, Ruben SD, Marden JR, Pao LY. Wind plant power optimization through yaw control using a parametric model for wake effects – a CFD simulation study. Wind Energy 2016;19(1):95–114.
[7] Thuerey Nils, Weißenow Konstantin, Prantl Lukas, Hu Xiangyu. Deep learning methods for Reynolds-Averaged Navier–Stokes simulations of airfoil flows. AIAA J 2020;58(1):25–36.
[8] Guo Xiaoxiao, Li Wei, Iorio Francesco. Convolutional neural networks for steady flow approximation. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. KDD '16, New York, NY, USA: Association for Computing Machinery; 2016, p. 481–90.
[9] De Avila Belbute-Peres Filipe, Economon Thomas, Kolter Zico. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In: Proceedings of the 37th international conference on machine learning. Proceedings of machine learning research, vol. 119, PMLR; 2020, p. 2402–11.
[10] Pfaff Tobias, Fortunato Meire, Sanchez-Gonzalez Alvaro, Battaglia Peter. Learning mesh-based simulation with graph networks. In: International conference on learning representations. 2021.
[11] Valencia Mario Lino, Fotiadis Stathi, Bharath Anil Anthony, Cantwell Chris D. REMus-GNN: A rotation-equivariant model for simulating continuum dynamics. In: ICLR 2022 workshop on geometrical and topological representation learning. 2022.
[12] Suk Julian, de Haan Pim, Lippe Phillip, Brune Christoph, Wolterink Jelmer M. Mesh convolutional neural networks for wall shear stress estimation in 3D artery models. In: Statistical atlases and computational models of the heart. multi-disease, multi-view, and multi-center right ventricular segmentation in cardiac MRI challenge - 12th international workshop, STACOM@MICCAI 2021, Strasbourg, France, September 27, 2021. Lecture notes in computer science, vol. 13131, Springer; 2021, p. 93–102.
[13] Ti Zilong, Deng Xiao Wei, Yang Hongxing. Wake modeling of wind turbines using machine learning. Appl Energy 2020;257:114025.
[14] Li Rui, Zhang Jincheng, Zhao Xiaowei. Dynamic wind farm wake modeling based on a bilateral convolutional neural network and high-fidelity LES data. Energy 2022;258:124845.
[15] Zhang Jincheng, Zhao Xiaowei. Wind farm wake modeling based on deep convolutional conditional generative adversarial network. Energy 2022;238:121747.
[16] Weller HG, Tabor G, Jasak H, Fureby C. A tensorial approach to computational continuum mechanics using object-oriented techniques. Comput Phys 1998;12(6):620–31.
[17] Edmunds Matt, Williams Alison J, Masters Ian, Banerjee Arindam, VanZwieten James H. A spatially nonlinear generalised actuator disk model for the simulation of horizontal axis wind and tidal turbines. Energy 2020;194:116803.
[18] Edmunds M, Williams AJ, Masters I, Croft TN. An enhanced disk averaged CFD model for the simulation of horizontal axis tidal turbines. Renew Energy 2017;101:67–81.
[19] Mirocha JD, Kosovic B, Aitken ML, Lundquist JK. Implementation of a generalized actuator disk wind turbine model into the weather research and forecasting model for large-eddy simulation applications. J Renew Sustain Energy 2014;6(1):013104.
[20] Daaou Nedjari H, Guerri O, Saighi M. Full rotor modelling and generalized actuator disc for wind turbine wake investigation. Energy Rep 2020;6:232–55, Technologies and Materials for Renewable Energy, Environment and Sustainability.
[21] Mycek Paul, Gaurier Benoît, Germain Grégory, Pinon Grégory, Rivoalen Elie. Experimental study of the turbulence intensity effects on marine current turbines behaviour. Part I: One single turbine. Renew Energy 2014;66:729–46.
[22] Giguere P, Selig M S. Design of a tapered and twisted blade for the NREL combined experiment rotor. Tech rep 1999, NREL/SR-500-26173, 1999.
[23] Badoe Charles E, Edmunds Matt, Williams Alison J, Nambiar Anup, Sellar Brian, Kiprakis Aristides, Masters Ian. Robust validation of a generalised actuator disk CFD model for tidal turbine analysis using the FloWave ocean energy research facility. Renew Energy 2022;190:232–50.

[24] Krogstad Per-Åge, Eriksen Pål Egil. "Blind test" calculations of the performance and wake development for a model wind turbine. Renew Energy 2013;50:325–33.
[25] Pierella Fabio, Krogstad Per-Åge, Sætran Lars. Blind test 2 calculations for two in-line model wind turbines where the downstream turbine operates at various rotational speeds. Renew Energy 2014;70:62–77, Special issue on aerodynamics of offshore wind energy systems and wakes.
[26] Shih Tsan-Hsing, Liou William W, Shabbir Aamir, Yang Zhigang, Zhu Jiang. A new k-ε eddy viscosity model for high Reynolds number turbulent flows. Comput & Fluids 1995;24(3):227–38.
[27] O'Doherty T, Mason-Jones Allan, O'doherty DM, Byrne CB, Owen I, Wang YX. Experimental and computational analysis of a model horizontal axis tidal turbine. In: Proceedings of the 8th European wave and tidal energy conference, Uppsala, Sweden. 2009, p. 7–10.
[28] Bronstein Michael M, Bruna Joan, LeCun Yann, Szlam Arthur, Vandergheynst Pierre. Geometric deep learning: Going beyond euclidean data. IEEE Signal Process Mag 2017;34(4):18–42.
[29] Fan Wenqi, Ma Yao, Li Qing, He Yuan, Zhao Eric, Tang Jiliang, Yin Dawei. Graph neural networks for social recommendation. In: The world wide web conference. WWW '19, New York, NY, USA: Association for Computing Machinery; 2019, p. 417–26.
[30] Wang Yuyang, Wang Jianren, Cao Zhonglin, Barati Farimani Amir. Molecular contrastive learning of representations via graph neural networks. Nat Mach Intell 2022;4:1–9.
[31] Chen Chi, Ye Weike, Zuo Yunxing, Zheng Chen, Ong Shyue Ping. Graph networks as a universal machine learning framework for molecules and crystals. Chem Mater 2019;31(9):3564–72.
[32] Jiang Weiwei, Luo Jiayun. Graph neural network for traffic forecasting: A survey. Expert Syst Appl 2022;207:117921.
[33] Paszke Adam, Gross Sam, Chintala Soumith, Chanan Gregory, Yang Edward, DeVito Zachary, Lin Zeming, Desmaison Alban, Antiga Luca, Lerer Adam. Automatic differentiation in PyTorch. In: NIPS 2017 workshop on autodiff. 2017.
[34] Fey Matthias, Lenssen Jan E. Fast graph representation learning with PyTorch Geometric. In: ICLR 2019 workshop on representation learning on graphs and manifolds. 2019.
[35] Deskos Georgios, Piggott Matthew D. Mesh-adaptive simulations of horizontal-axis turbine arrays using the actuator line method. Wind Energy 2018;21(12):1266–81.
[36] Dahlberg J-Å. Assessment of the Lillgrund wind farm: power performance, wake effects. Tech. rep., LG Pilot Rep., Vat- tenfall Vindkraft AB; 2009.
[37] Hamilton Will, Ying Zhitao, Leskovec Jure. Inductive representation learning on large graphs. In: Advances in neural information processing systems, Vol. 30. Curran Associates, Inc.; 2017.
[38] Hamilton William L. Graph representation learning. Synth Lect Artif Intell Mach Learn 2020;14(3):1–159.
[39] Kipf Thomas N, Welling Max. Semi-supervised classification with graph convolutional networks. In: 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24-26, 2017, conference track proceedings. 2017.
[40] Veličković Petar, Cucurull Guillem, Casanova Arantxa, Romero Adriana, Liò Pietro, Bengio Yoshua. Graph attention networks. In: International conference on learning representations. 2018.
[41] Xu Keyulu, Li Chengtao, Tian Yonglong, Sonobe Tomohiro, Kawarabayashi Kenichi, Jegelka Stefanie. Representation learning on graphs with jumping knowledge networks. In: Proceedings of the 35th international conference on machine learning, ICML 2018, StockholmsmäSsan, Stockholm, Sweden, July 10-15, 2018. Proceedings of machine learning research, vol. 80, PMLR; 2018, p. 5449–58.
[42] Chi Huixuan, Wang Yuying, Hao Qinfen, Xia Hong. Residual network and embedding usage: New tricks of node classification with graph convolutional networks. J Phys Conf Ser 2022;2171(1):012011.
[43] He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition. CVPR, 2016, p. 770–8.
[44] Chen Ming, Wei Zhewei, Huang Zengfeng, Ding Bolin, Li Yaliang. Simple and deep graph convolutional networks. In: Proceedings of the 37th international conference on machine learning. Proceedings of machine learning research, vol. 119, PMLR; 2020, p. 1725–35.
[45] Loshchilov Ilya, Hutter Frank. Decoupled weight decay regularization. In: International conference on learning representations. 2019.
[46] Smith Leslie N, Topin Nicholay. Super-convergence: very fast training of neural networks using large learning rates. In: Artificial intelligence and machine learning for multi-domain operations applications, Vol. 11006. SPIE, International Society for Optics and Photonics; 2019, 1100612.
[47] Ronneberger Olaf, Fischer Philipp, Brox Thomas. U-net: Convolutional networks for biomedical image segmentation. In: Navab Nassir, Hornegger Joachim, Wells William M, Frangi Alejandro F, editors. Medical image computing and computer-assisted intervention – MICCAI 2015. Cham: Springer International Publishing; 2015, p. 234–41.
[48] Hou Yuqing, Li Hui, Chen Hong, Wei Wei, Wang Jiayue, Huang Yicang. A novel deep U-net-LSTM framework for time-sequenced hydrodynamics prediction of the SUBOFF AFF-8. Eng Appl Comput Fluid Mech 2022;16(1):630–45.
[49] Jiang Zhihao, Tahmasebi Pejman, Mao Zhiqiang. Deep residual U-net convolution neural networks with autoregressive strategy for fluid flow predictions in large-scale geosystems. Adv Water Resour 2021;150:103878.
[50] Vogel Christopher, Willden Richard. Investigation of wind turbine wake superposition models using Reynolds-averaged Navier-Stokes simulations. Wind Energy 2020;23:593–607.