

A Physics-Informed Graph Neural Network Surrogate for Operational Estimation of Wake-Induced Power Losses in Wind Farms

Darsheel Paila

School of Science and Engineering

High School Student Researcher

darsheelpaila@gmail.com

Oct 14, 2026

Abstract

Limited land packs wind turbines a few rotor diameters apart as electricity demand continues to rise. Wind already supplies about 10% of U.S. electricity and 23% of Texas ERCOT generation. Each rotor leaves a wake of slower, more turbulent air. Power scales with the cube of wind speed, so a 15% speed drop is about a 39% power drop, and dense farms often lose 10–20% of isolated-rotor energy. This study asks how to estimate that wake-affected energy on an operating farm from quantities the farm already records. The questions are whether a one-to-one upstream-to-downstream map can represent overlapping wakes, and whether a farm graph can score those wakes in one iteration. The variables are upstream generation, wind-alignment angle, spacing, and downstream energy. The layout is fixed (Sweetwater, Texas, 2013, ten-minute records). The study is limited to neural estimators. The purpose was to build a reusable surrogate that estimates wake-affected energy for the turbines in a farm graph in one pass. The project was undertaken because the historian already logs every meter over 2 months; what is missing is an estimator that sees the neighborhood that produced the reading. The hypothesis was that pairwise neural networks would fit a hub meter but could not scan overlapping wakes in one pass; that using Jensen’s wake equation only as a soft constraint in the loss would improve the close-pair estimate; and that a physics-induced graph neural network would then estimate many links with one set of weights in one iteration. Filtered 2013 ten-minute generation records (negatives set to zero; values above 1600 kilowatt-hours removed) were aligned with wind direction on turbine coordinates. One hundred eight validated directed wake links formed the farm graph. Networks were trained in a 1000-epoch window and evaluated on a chronological 80%/20% split. Hub T118 was estimated from T117 (0.54 km) and T145 (2.61 km). Errors were reported in three isolated stages: pairwise feed-forward, long short-term memory, and temporal-convolutional

networks; then a physics-induced pairwise feed-forward network; then the physics-induced graph neural network. The feed-forward network reached 292.13 kilowatt-hours ($R^2 = 0.724$) on T117 to T118 and 242.02 (0.811) on T145 to T118. The 24-hour memory and convolutional networks were worse (336.79 and 322.21; 303.42 and 262.66). Adding two pairwise guesses would count the same meter twice. The physics-induced pairwise network reached 270.45 (0.764) and 240.29 (0.813), supporting the close-pair hypothesis. The graph network on the same single edge reached 276.14 and 254.41. A shared-weight scan of 28 links reached mean error 198.86 kilowatt-hours and mean R^2 0.866, supporting the graph hypothesis.

Contents

List of Tables	1
List of Figures	2
1 Introduction	2
1.1 Purpose of the Study	2
1.2 Hypotheses	3
1.3 Background: Wind-Farm Wakes	3
1.4 Background: Graphs and Graph Neural Networks	4
1.5 Background: Why a Physics-informed Graph Net	4
2 Materials, Methods, and Procedures	5
3 Results	7
3.1 Fast-Resolving Wake Models (Simulation)	9
4 Discussion and Conclusions	9
Literature Cited	10

List of Tables

1	Shared experimental settings	6
2	Pairwise-range test RMSE and R^2	7
3	Shared-weight farm scan on 28 links	7
4	Fast wake models on the two T118 links	9
5	Fast wake models on five neighborhood links	9

List of Figures

1	Overlapping wakes made visible by fog, Horns Rev 1	2
2	Wind rose at the Sweetwater mast, 2013	3
3	Four-step engineering wake procedure	4
4	Sweetwater array, 2-D top view	5
5	Replication procedure	5
6	Farm graph on estimated topography	6
7	Physics-informed graph neural network	6
9	Isolated estimation stages	7
8	PyWake flow field at 8 m/s	8
10	Chronological holdout, first week	9

1 Introduction

1.1 Purpose of the Study

The purpose of this study was to build a reusable engineering surrogate that estimates wake-affected energy for the turbines of an operating commercial wind farm from quantities the farm already records, and that can score overlapping neighboring wakes in a single forward pass. Wind already supplies roughly 10% of United States electricity and roughly 23% of generation on the Texas ERCOT grid. Meeting that demand on limited land packs rotors only a few diameters apart, and every operating rotor leaves behind a wake of slower, more turbulent air. Because power scales with the cube of hub-height wind speed, a 15% reduction in incoming speed produces roughly a 39% reduction in power, and densely packed arrays commonly lose 10–20% of the energy an isolated rotor would have produced. Figure 1 is an aerial photograph of that phenomenon at the Horns Rev 1 off-shore farm in Denmark, where fog happens to make the overlapping wakes visible.



Figure 1: Overlapping wakes made visible by fog at Horns Rev 1, Denmark. Background illustration of the wake phenomenon only; not the Sweetwater farm used in this study.

The Sweetwater historian already records turbine-level generation. The engineering need is an estimator that incorporates the neighborhood of turbines contributing to a measurement and can be reused as wind direction changes which turbines are upstream. Figure 2 is the 2013 wind rose recorded at the Sweetwater meteorological mast: southerly flow dominates, which is why the validated wake-link list used in this study is built primarily from a southerly sector. A wake, graph, and graph neural network are defined before the site-specific results are presented. A three-dimensional large-eddy simulation is not the accuracy comparison. The fast visual comparison is a PyWake flow field, which resolves a wake without a volume mesh. Through the duration of this paper, I define fidelity by error: the proximity of an estimation to a true value, within the context of fast-computation models: not in the context of comparing the PI-GNN’s accuracy in power estimation with intensive computational fluid dynamics simulations like Reynolds-Averaged Navier Stokes (RANS) or Large-Eddy Simulations (LES).

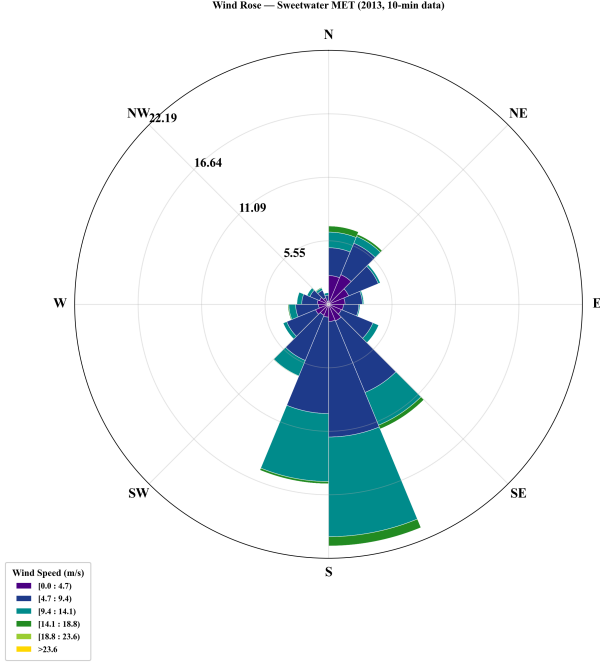


Figure 2: Wind rose at the Sweetwater meteorological mast, 2013, ten-minute records. Southerly flow dominates.

1.2 Hypotheses

1. The Sweetwater farm and the recorded wind can be represented as a directed wake graph (turbines as nodes; validated upstream-to-downstream links as edges) so that any ten-minute farm state can be written as a single input.
2. Pairwise neural estimators (feed-forward, long short-term memory, and temporal convolutional networks) can map one upstream turbine onto one downstream meter using upstream energy, wake-alignment angle, and spacing, with test root-mean-square error (RMSE) and R^2 on a chronological holdout showing measurable skill.
3. Keeping the same pairwise feed-forward body and adding Jensen’s wake equation as a buffer with physical loss that enforces a physically-abiding value during training will improve the accuracy of the estimate.
4. A physics-informed graph neural network (PI-GNN) trained on the farm graph will allow one set of shared weights to estimate wake-affected energy at many turbines in a single iteration, with two-node diagnostics reported separately from the multi-link shared-weight farm scan.
5. The procedure is replicable from farm-recorded quantities (filtered 2013 ten-minute generation, the mast vane, and surveyed coordinates), within the main limits of the study: one farm, 60 days (2 months), and no completed simultaneous multi-neighbor run over every validated upstream of T118.

1.3 Background: Wind-Farm Wakes

Prior work on farm-scale wakes falls into three broad categories: 1. High-resolution computational-fluid-dynamics solvers like large-eddy simulations approximate the governing Navier–Stokes field directly. While they are the most physically faithful description of the flow, they are far too expensive to refresh every ten minutes to match wind-sensor measurement rates during operation. 2. Engineering wake models derived from conservation of momentum, such as Jensen’s original top-hat cone and later Gaussian forms, approximate a steady velocity deficit and then combine several upstream deficits with a fixed rule, such as a root-sum-square. Those are the fast-resolving models in this paper: PyWake’s NOJ and Bastankhah Gaussian closures, which return a visual velocity field in one pass. 3. Data-driven models instead map wind or supervisory features directly onto a target turbine’s energy output.

Engineering practice still leans on a four-step procedure (Figure 3) to compute the power of a downstream turbine. Step 1 constructs a wake-deficit field $\delta u(d, r)$ at downstream distance d and radial distance r from the axis of an upstream rotor. A continuous form of that field, used in place of Jensen’s discontinuous original cone so that the deficit varies smoothly across the wake boundary, is

$$\delta u(d, r, \alpha) = 2\alpha \left(\frac{R_0}{R_0 + \kappa d} \right)^2 \exp \left(- \left(\frac{r}{R_0 + \kappa d} \right)^2 \right), \quad (1)$$

where α is the induction factor, R_0 is the rotor radius, and κ is a surface-roughness-dependent decay constant. Step 2 averages that deficit over the downstream rotor disk to obtain a pairwise deficit δu_{ij} between an upstream turbine i and a downstream turbine j . Step 3 superposes the deficits of every upstream turbine that influences turbine i , most commonly as a root-sum-square,

$$\delta u_i = \sqrt{\sum_{\{j \mid W_{ij}=1\}} (\delta u_{ij})^2}, \quad (2)$$

where $W_{ij} = 1$ marks turbine i as an upstream influence on turbine j . Step 4 converts the recovered wind speed to power with the familiar cube law,

$$p_i = \frac{1}{2} \rho A C_p \bar{u}_i^3, \quad \bar{u}_i = (1 - \delta u_i) U, \quad (3)$$

with air density ρ , rotor-swept area A , and power coefficient C_p . Step 3 decides how several overlapping wakes are combined. The rule and its weights are fixed by the engineering model rather than learned from data. The residual coded for this study uses the centerline Jensen deficit $\Delta U/U_\infty = 2a/(1 + 2kx/D)^2$ with $a = (1 - \sqrt{1 - C_T})/2$, multiplied by a Gaussian lateral weight that is active only for wake-alignment angles $|\theta| \leq 30^\circ$, together with representative constants $D = 91$ m, $C_T = 0.80$, and $k = 0.05$.

Traditional four-step engineering wake procedure for fast, neighborhood-wide wind farm estimation

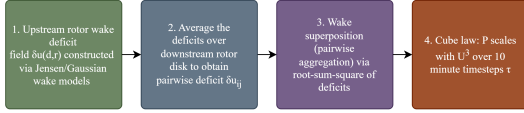


Figure 3: Four-step engineering wake procedure. The physics-informed graph neural network keeps Step 1 as a loss constraint and replaces Steps 3–4 with learned messages. The same four steps are what the PyWake flow field in Figure 8 resolves visually.

Several one-to-one pairwise neural nets might appear to provide a neighborhood estimate by combining their outputs through an idea called wake superposition. For example, separate models could estimate T117→T118 and T145→T118 and then be averaged. This does not represent overlapping wakes correctly: both models predict the same T118 quantity, so their outputs are not independent measurements. The physical wake model instead combines velocity deficits before applying the turbine power relation (Equation 2). Combining pairwise energy estimates would therefore ignore the shared downstream rotor and would not provide a neighborhood representation. Instead, the present study proposes a potentially suited neighborhood model, the PI-GNN, which aggregates all available graph messages with one shared set of weights, attempting to reduce the inaccuracies that arise from the superposition of power outputs of multiple linked turbine pairs.

1.4 Background: Graphs and Graph Neural Networks

A feed-forward network represents each training example as an independent vector. For a single turbine pair, the inputs in this study are upstream energy, wake-alignment angle, and spacing. A wind farm requires more structure because the number of relevant upstream turbines changes with wind direction and several neighbors can contribute to the same measurement. A graph provides that structure.

Formally, a graph $G = (V, E)$ contains nodes V and relationships E . In a directed graph, each edge has an ordered source and destination. Graphs can also store feature vectors on nodes and edges. In the present application, turbines are nodes and validated upstream-to-downstream wake relations are directed edges.

Graph neural networks use *message passing* to update node states from their neighbors. The mean-aggregation form used here follows GraphSAGE and the broader message-passing framework of Gilmer et al. (2017). For node i ,

$$h_i^{(k+1)} = \sigma \left(W_{\text{self}} h_i^{(k)} + W_{\text{nbr}} \sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} h_j^{(k)} \right), \quad (4)$$

where $\mathcal{N}(i)$ is the set of validated upstream neighbors,

W_{self} and W_{nbr} are learned matrices shared across nodes, and σ is a nonlinear activation. Because the aggregation does not depend on neighbor ordering, the network is permutation-invariant. One layer incorporates immediate upstream information and second layer can incorporate information from the upstream of those neighbors.

This structure maps directly to Sweetwater. Node features contain local Cartesian north–south and east–west coordinates, while edge features contain the upstream generated electricity value, the signed wake-alignment angle (not the wind direction itself, and not the bearing alone), and spacing. At T118, for example, messages from T117 and T145 are processed by the same shared functions and aggregated without regard to their listing order. A final linear readout produces the predicted energy at each turbine. The implementation uses plain PyTorch.

The graph formulation also avoids treating a variable-size neighborhood as a fixed-length vector. A flat feed-forward representation would have to assume a particular ordering and number of neighbors, even though the active wake relationships change with wind direction. The graph instead represents the physical relationships directly and allows the same learned weights to be reused throughout the farm.

1.5 Background: Why a Physics-informed Graph Net

The choice of the PI-GNN follows from comparisons with simpler or closely related approaches.

Fast-resolving wake models. Jensen’s top-hat model and the Bastankhah Gaussian provide a visual velocity field without a three-dimensional mesh. PyWake is the implementation used for that picture (Figure 8). Their multi-wake combination rules are fixed by the engineering model rather than learned from the Sweetwater records. Inside the PI-GNN, Jensen enters only as a soft loss. The stand-alone energy scores of the two closures are reported separately from the four neural stages.

Pairwise snapshot and sequence networks. Feed-forward, LSTM, and temporal convolutional networks provide one-to-one mappings from an upstream turbine to one downstream meter. Temporal memory can improve a single edge, but these models do not contain an operator for combining a variable number of upstream neighbors in one forward pass. Wakes are dynamic; one wake can affect more than one turbine. The pairwise analysis may not account for the nuance of indirect wake currents, and therefore the same node can produce different power outputs and RMSE values when analyzed with different upstream turbines. When finished pairwise energies are averaged into one neighborhood value, they are still two guesses of the same meter.

Data-only graph networks. A graph network without a physics term already provides permutation-invariant message passing, but its loss does not prefer a wake-shaped deficit on aligned links. The present PI-

GNN adds a Jensen residual and a consistency penalty to the same graph structure.

Physics-informed pairwise network. The PI-FFN uses the same Jensen-in-loss formulation as the PI-GNN but has no graph structure. It therefore isolates the contribution of the physics residual without changing the pairwise task.

The present study uses an ordinary GraphSAGE mean-aggregation architecture, shown with its units in Figure 7, with learned node and edge encoders, and applies Jensen’s model as a soft loss constraint.

This distinction matters because the present targets are real SCADA measurements, which can contain curtailment, yaw misalignment, and sensor effects that are not represented by an idealized wake function. Additionally, instead of traditional transductive modeling, GraphSAGE enables inductive modeling for wind farm power estimation by learning to generate embeddings for newly added or completely unseen wind turbines from their local spatial relationships and features, eliminating the need to retrain the entire network when a layout changes. Regardless, the present study treats the PI-GNN as a fixed-layout estimator whose main engineering contribution is shared neighborhood aggregation with an explicit physics residual.

2 Materials, Methods, and Procedures

The study used the Sweetwater wind farm in Nolan County, Texas. Data included the 2013 ten-minute SCADA generation archive, a meteorological vane file, surveyed turbine coordinates, and a laboratory-provided list of validated upstream/downstream wake pairs. Software was Python 3 with NumPy, pandas, scikit-learn, PyTorch, matplotlib, and PyWake 2.6, run on a local workstation.

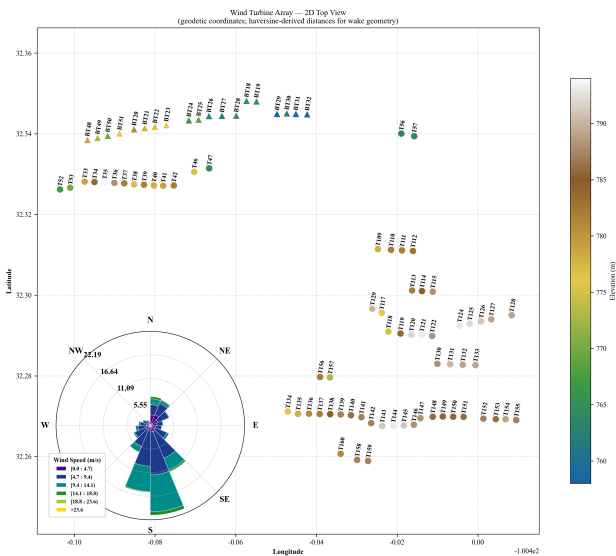


Figure 4: Sweetwater turbine array, 2-D top-down plan view. Marker color is elevation. The same 2013 mast record underlies the inset wind rose.

The separation drawn in Figure 4 between turbine pair T_1 and T_2 is calculated from latitude and longitude with the Haversine formula:

$$D = 2R \sin^{-1} \left[\sin^2 \left(\frac{\phi_2 - \phi_1}{2} \right) + \cos(\phi_2) \cos(\phi_1) \sin^2 \left(\frac{\lambda_2 - \lambda_1}{2} \right) \right]^{1/2}, \quad (5)$$

where ϕ and λ denote latitude and longitude in radians and $R = 6371$ km is the Earth’s radius.

To capture directional alignment relative to the wind, the bearing angle θ_b pointing from T_1 to T_2 is found using a two-argument arctangent:

$$\theta_b = \text{atan2}(Y, X), \quad (6)$$

with

$$Y = \cos \phi_1 \sin \phi_2 - \sin \phi_1 \cos \phi_2 \cos(\lambda_2 - \lambda_1), \quad X = \cos \phi_2 \sin(\lambda_2 - \lambda_1). \quad (7)$$

Subtracting the wind direction θ_w from this bearing gives the angular offset between the turbine-to-turbine direction and the incoming wind:

$$\Delta\theta = \theta_b - \theta_w. \quad (8)$$

Since angular values wrap around every 360° (that is, $0^\circ \equiv 360^\circ$), $\Delta\theta$ is encoded using sine and cosine components rather than the raw angle. This prevents discontinuities at the wrap-around point and keeps angles that are close together, including those near $0^\circ/360^\circ$, represented as numerically close values.

Figure 5 is the six-step procedure used to produce every neural-network number in Section 3. It is written to be specific enough to replicate. Table 1 records the shared experimental settings.

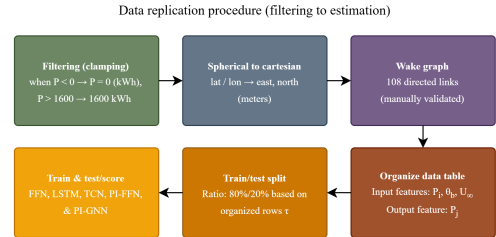


Figure 5: Replication procedure. Step 6 scores the PI-GNN and the fast wake closures on the same chronological holdout.

Figure 6 is the validated farm graph: 108 directed wake links on the same layout as Figure 4. The colored field is a smoothed elevation estimate constrained by surveyed hub heights, drawn with 4 m isolines; it is not a surveyed digital elevation model. Turbine T117 sits north of hub T118, so that couple is a northerly-inflow edge; turbines T141–T146 sit south of T118 and dominate under the farm’s prevailing southerly inflow (Figure 2). A *pairwise sample* is the two-node subgraph induced by a single edge, used only to isolate the pairwise stages of Section 3 from the graph stages. The

Table 1: Shared experimental settings used across every model reported in this study.

Item	Value
Site / year / resolution	Sweetwater, Texas, 2013, 10 min
Filter	$P < 0 \rightarrow 0$; $P > 1600 \text{ kWh} \rightarrow 1600$
Validated directed links	108
Primary hub / neighbors	T118; T117 (0.54 km, north); T145 (2.61 km)
Split	80% / 20%, chronological
Physics weights	$\lambda_{\text{Jensen}} = 0.15$, $\lambda_{\text{cons}} = 0.05$, $ \theta \leq 30^\circ$
PI-GNN	2 SAGE layers, hidden width 64, 41,857 parameters
Fast wake models	PyWake NOJ and Bastankhah Gaussian
Disk constants	$D = 91 \text{ m}$, $C_T = 0.80$, $k = 0.05$
Shared farm scan	28 links, one weight set

operational object of this study is the full graph: every incoming message at a hub is aggregated in one forward pass. No layered sum of separately trained pairwise kilowatt-hour outputs was trained or scored, for the reasons given in Section 1.3.

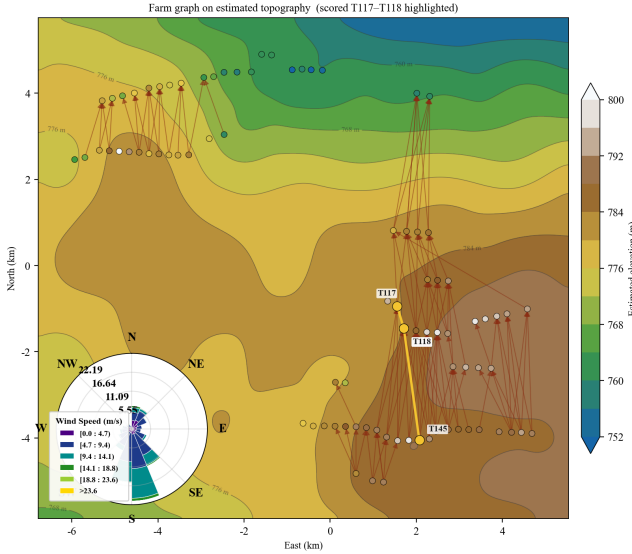


Figure 6: Directed farm graph on estimated Sweetwater topography, 108 manually validated wake links. Contours are 4 m isolines. Gold highlights T117–T118 and T145–T118. The inset is the 2013 mast wind rose.

Figure 7 is the PI-GNN itself. A linear node encoder and a two-layer edge encoder map raw features to a hidden width of 64. Two residual mean-aggregation GraphSAGE layers (Equation 4) then pass messages along validated wake edges and update every hub in the graph. A final linear readout writes an estimated energy at every node based on the environmental context generated from message-passing (41,857 trainable parameters in total), and maps each hidden state to a ten-minute power. The

drawing below shows the unit pattern, not all 64 hidden units. The scored PyWake simulation in Figure 8 includes both Jensen (NOJ) and Bastankhah estimation models.

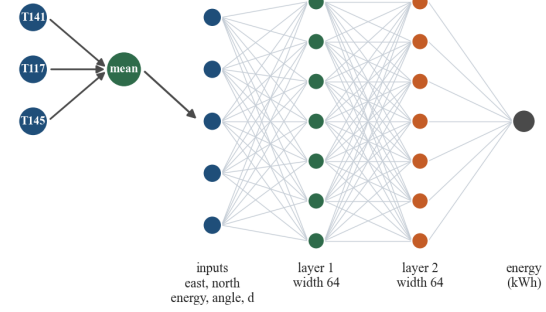


Figure 7: Physics-informed graph neural network. Neighbor messages are mean-aggregated, then two hidden layers of width 64 write a ten-minute energy. Seven of the 64 units are drawn in each hidden layer.

The full training objective combines a data term with two physics terms:

$$\mathcal{L} = \mathcal{L}_{\text{data}} + 0.15 \mathcal{L}_{\text{Jensen}} + 0.05 \mathcal{L}_{\text{cons}}. \quad (9)$$

$\mathcal{L}_{\text{data}}$ is the mean-squared error against filtered, recorded downstream energy. $\mathcal{L}_{\text{Jensen}}$ is the mean-squared error between the model’s prediction and the differentiable Jensen target of Section 1.3 on wakes with $|\theta| \leq 30^\circ$. $\mathcal{L}_{\text{cons}}$ is a squared hinge, $\text{ReLU}(\hat{p}_{\text{downstream}} - p_{\text{upstream}})^2$, applied on those same aligned wakes to discourage a downstream estimate that exceeds the energy of the turbine supposedly generating its wake, a no-free-energy penalty. The constants $D = 91 \text{ m}$, $C_T = 0.80$, and $k = 0.05$ used inside $\mathcal{L}_{\text{Jensen}}$ are representative values that add context from the Sweetwater farm. The PI-FFN control uses the identical Equation 9 on a snapshot feed-forward network (hidden layers of width 128, 64, and 32) with no graph structure at all. The LSTM (two layers, hidden width 64) and the TCN (dilation factors 1, 2, 4, 8) each read a 24-hour window of the same three input features. Every network was trained with the Adam optimizer. The PI-GNN used early stopping with patience 10; the best epoch on the two-node diagnostic was as low as epoch 2 for T117→T118 and epoch 4 for T145→T118.

The fast visual model is PyWake 2.6 provide power estimations at data-driven model-adjacent speeds for more comparison baselines. The turbine object is a rotor of diameter 91 m and hub height 80 m, with $C_T = 0.80$ up to the rated speed. NOJ and the Bastankhah–Porté-Agel Gaussian are each run with wake expansion $k = 0.05$ and root-sum-square superposition. Figure 8 is that simulation at a freestream of 8 m/s. The upper row is southerly inflow (wind from 180°) through T141–T146 toward T118. The lower row follows the T117→T118 bearing (wind from 343°), so the wake centerline passes through T118. At the T118 rotor the southerly NOJ and

Bastankhah fields leave 7.60 m/s and 7.86 m/s. Along the close pair they leave 6.24 m/s and 6.55 m/s. The holdout energy scores use the same two closures on every test row, scaled by the upstream SCADA meter, because a manufacturer $C_P(U)$ curve was not available for a PyWake-simulated power-curve object.

In this study’s graph formulation, upstream turbines that have no validated incoming wake edge are handled by setting their incoming edge vector to a zero vector of the same dimension the edge encoder would otherwise produce, so that a hub with fewer active neighbors is processed by the identical shared weights as a hub with more. That is the variable-neighborhood behavior expected of a message-passing network.

3 Results

A raw RMSE gap between a pairwise feed-forward network and a 28-link PI-GNN would confound two variables at once: the range of the comparison (one pair versus a 28-link scan) and the type of network (feed-forward versus graph). That confounded gap is not used anywhere in this study as the decisive comparison. Neural results are isolated into four stages (Figure 9, Stages 1–3 keep the same one-edge span so that architecture and the Jensen residual can each be read without a simultaneous change of range. Stage 4 is a different scientific claim, one shared weight file scanning 28 links in one iteration, and is reported in Table 3, not folded into Table 2 as though it were a better RMSE than the pairwise result. The fast-resolving NOJ and Bastankhah scores are a further, separate comparison on the same chronological holdout. Test error throughout is reported as RMSE in kilowatt-hours and R^2 on a time-ordered holdout. No t -test or ANOVA was applied, because the comparison is predictive error on a holdout, not a randomized controlled treatment.

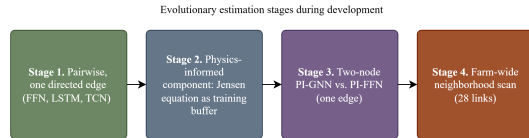


Figure 9: Evolutionary isolated estimation stages. Stages 1–3 keep the same one-edge (pairwise) range. Stage 4 is a different scientific claim (28-link shared scan, neighborhood) and is reported separately.

Table 2: Pairwise-range test RMSE (kWh) and R^2 on the chronological holdout. Stages 1–3 only; same one-edge span. $n = 1728$ each.

Model / evaluation	T117→T118	T145→T118
FFN (pairwise, data only)	292.13 (0.724)	242.02 (0.811)
LSTM, 24 h (pairwise)	336.79 (0.628)	322.21 (0.659)
TCN, 24 h (pairwise)	303.42 (0.698)	262.66 (0.774)
PI-FFN (Jensen in loss only)	270.45 (0.764)	240.29 (0.813)
PI-GNN, 2-node diagnostic	276.14 (0.754)	254.41 (0.791)

Stage 1 (pairwise, one edge). Among the three pairwise networks, the snapshot FFN reached $R^2 = 0.724$ and 0.811 on the two neighbors of T118. The lower $R^2 = 0.724$ value of the T117 → T118 was inevitable since training data for timesteps τ where wind direction was northerly was significantly sparser. The 24-hour LSTM and TCN were both worse than the plain FFN on the close pair, so additional temporal memory was not the missing ingredient. Ultimately, none of these three networks can scan overlapping wakes: both outputs are estimates of the identical T118 meter, and layering their outputs is not a neighborhood operator (Section 1.3).

Stage 2 (physics residual, one edge). The PI-FFN reached 270.45 kWh ($R^2 = 0.764$) on T117→T118, below the data-only FFN on the close pair while using the identical feed-forward body and the identical one-edge span; the far 2.61 km pair barely moved (240.29 kWh). This stage isolates the value of the physics residual, not the graph structure, and it does not by itself score the cone as a stand-alone flow model. That stand-alone score is Table 4.

Stage 3 (architecture change, one edge). On a single incoming edge, the PI-GNN sits close to the PI-FFN (test RMSE 276.14 kWh, $R^2 = 0.754$ on T117→T118; 254.41 kWh, $R^2 = 0.791$ on T145→T118). That comparison holds the range fixed. A two-node graph is still, in effect, a pair, and the comparison isolates the architecture change. It is therefore not a comparison of the graph model against all pairwise models on the 28-link task. Restricted to well-aligned readings on T117→T118, diagnostic RMSE was 214.75 kWh ($R^2 = 0.869$, $n = 353$) versus 304.93 kWh when $|\theta| > 60^\circ$, consistent with a wake-shaped residual that is informative near the rotor and uninformative far from alignment.

Table 3: Shared-weight farm scan on 28 links. Different span from Table 2; not a pairwise RMSE comparison.

Farm-scan claim	RMSE (kWh)	R^2	n
PI-GNN, 28 links, one weight set	198.86	0.866	6048
Aligned subset of that shared holdout ($ \theta \leq 30^\circ$)	154.95	0.916	1089

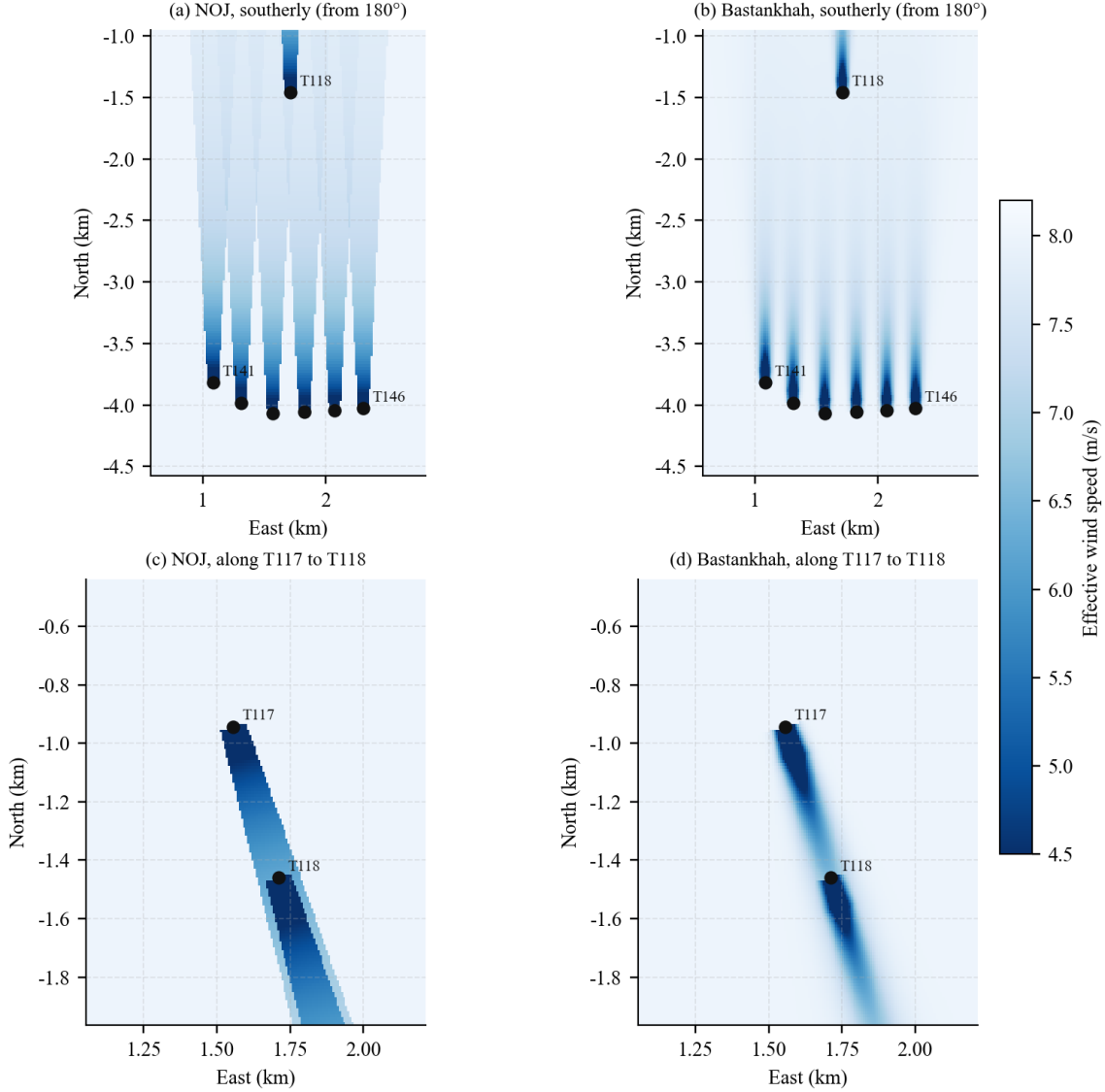


Figure 8: PyWake 2.6 effective wind speed at a freestream of 8 m/s, $D = 91$ m, hub height 80 m, $C_T = 0.80$, $k = 0.05$. (a, b) Southerly inflow from 180°, T141–T146 toward T118. (c, d) Wind from 343°, along the T117→T118 bearing. At the T118 rotor the four fields leave 7.60, 7.86, 6.24, and 6.55 m/s. This is one snapshot, not the holdout RMSE.

Stage 4 (neighborhood-wide physics-informed farm scan). One shared weight file, evaluated on 28 links in a single iteration, reached test RMSE 198.86 kWh, and $R^2 = 0.866$ ($n = 6048$). This number describes a shared-weight scan across multiple Sweetwater links. It is the first architecture that produces a lower demonstrated RMSE when expanded to message-based neighborhood-wide analysis and the first below 200 kWh RMSE. However, the PI-GNN is not a simultaneous multi-neighbor inference over **every** validated upstream of T118, and it is not direct evidence that the PI-GNN is more accurate than the FFN on a common task, since both the span and the architecture changed relative to Stage 1, which are confounding variables. For that reason this RMSE is reported in Table 3 rather than appended to Table 2. That said, Schøler, J. P. et al. explain that pairwise models that rely on singular wake

velocity deficits decline in fidelity when aggregated into neighborhood data, remaining limited by their algebraic structure and cannot fully capture the nonlinear nature of wake interactions. Conversely, the PI-GNN has shown to improve in Table 3. This important RMSE increase vs. decline indirectly illustrates the PI-GNN’s relative robustness in neighborhood analysis. Figure 10 shows the PI-GNN on the first week of the chronological holdout for the close diagnostic T117→T118.

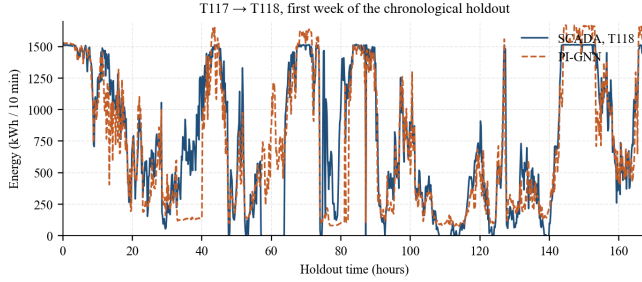


Figure 10: Chronological holdout, first week: SCADA-recorded energy at T118 versus the two-node PI-GNN diagnostic T117→T118, 0.54 km.

3.1 Fast-Resolving Wake Models (Simulation)

Figure 8 is a visual comparison of wake-informed power loss estimation. NOJ leaves a flat-topped cone. The Bastankhah Gaussian leaves a smooth deficit that decays off the centerline. Neither panel is a three-dimensional mesh.

Table 4 scores the same two closures as ten-minute energy on the chronological holdout used in Table 2. Downstream energy is the upstream meter times the cube of the remaining speed, $\hat{P}_i = P_j(1 - \delta u_{ij} w(\theta))^3$, with the lateral Gaussian weight $w(\theta)$. On T117→T118 the two-node PI-GNN (276.14 kWh, $R^2 = 0.754$) is closer to the meter than NOJ (301.26 kWh, $R^2 = 0.707$) and Bastankhah (297.74 kWh, $R^2 = 0.714$). On T145→T118 the cone has mostly decayed: NOJ is 264.41 kWh, Bastankhah 264.54 kWh, and the PI-GNN 254.41 kWh. A data-only feed-forward net is lower still on that far link (242.02 kWh).

Table 4: NOJ and Bastankhah test RMSE (kWh) and R^2 on the same chronological holdout as Table 2, $n = 1728$ each. The closures are applied to upstream SCADA energy. The PyWake speeds in Figure 8 are a separate 8 m/s snapshot.

Model	T117→T118	T145→T118
Jensen (PyWake NOJ)	301.26 (0.707)	264.41 (0.774)
Bastankhah Gaussian	297.74 (0.714)	264.54 (0.774)
PI-GNN, 2-node	276.14 (0.754)	254.41 (0.791)

Table 5 applies the same closures to five links that sit inside the 28-link scan and whose pair tables are on disk ($n = 1728$ each). The unweighted means are 213.77 kWh (NOJ) and 211.31 kWh (Bastankhah). The shared PI-GNN number for the full 28-link set is 198.86 kWh ($R^2 = 0.866$, $n = 6048$). Since these 5 links were randomly sampled for the neighborhood estimation accuracy analysis of the NOJ and Bastankhah models, the returned RMSE values can generalize to the entire 28 links and can be directly compared with the PI-GNN.

Table 5: NOJ and Bastankhah test RMSE (kWh) and R^2 on five links inside the 28-link scan. The shared PI-GNN mean of 198.86 kWh is a different row set and is not an average of these five rows.

Link	Jensen (NOJ)	Bastankhah
T160→T139	156.12 (0.920)	151.85 (0.924)
T159→T143	151.80 (0.924)	150.67 (0.925)
T130→T122	225.72 (0.826)	222.78 (0.830)
T114→T112	260.52 (0.776)	258.74 (0.779)
T113→T112	274.70 (0.751)	272.50 (0.755)
Unweighted mean	213.77	211.31

4 Discussion and Conclusions

The four stages in Section 3 address different questions which is why they are reported separately. Stage 1 showed that a snapshot FFN fits a substantial fraction of the T118 meter, with test R^2 above 0.72 for both neighbors. Adding temporal memory did not improve the pairwise result on the close edge. Combining separate pairwise energy estimates does not represent an overlapping neighborhood, so that construction was not used. Stage 2 showed that inserting Jensen’s model only as a loss constraint lowered close-pair RMSE and barely moved the 2.61 km pair, consistent with a deficit cone that is informative near a rotor and uninformative far from it. Stage 3 showed that, on the identical pair, the two-node PI-GNN did not outperform the PI-FFN. A two-node graph is architecturally, still a pair, which is precisely why that comparison is the fair architecture-only contrast in this study. Stage 4 addresses shared-weight inference across a larger set of wake links: one weight file across 28 links achieved $R^2 = 0.866$ in one iteration. This shared-weight formulation is related to the transferable interaction used by Park and Park (2019) on simulated layouts, but here instead it was trained on real chronological data derived from SCADA measurements.

For the fast-resolving models, on the close, the strongly waked link, the PI-GNN is closer than either PyWake closure. On the far link the advantage shrinks to about 10 kWh, and a data-only feed-forward net is closer still. The flow picture in Figure 8 is why. At 8 m/s the southerly cones have largely decayed by the time they reach T118 (7.60 m/s and 7.86 m/s), while the close pair still carries a deep deficit (6.24 m/s and 6.55 m/s). A learned message can use the meter when the cone has little left to say. The five-link means (213.77 kWh and 211.31 kWh) and the 28-link scan (198.86 kWh) remain different row sets.

The cones were a fast visual model and the training residual in this study. Pairwise deep networks remain strong one-to-one maps, but none of them has a neighborhood operator. A data-only graph network can already pass messages and is already permutation-invariant, but it carries no wake-physics residual anywhere in its loss. The PI-FFN carries the residual and still has no farm-wide scan. The PI-GNN of this

study combines the directed wake graph, a permutation-invariant incoming neighborhood, one shared weight file, one iteration, and a Jensen residual, and differs from Park and Park’s PGNN specifically in where that physics prior is allowed to act: as a loss-level residual on real SCADA data here, rather than as a hard-wired edge-weighting basis function on simulated FLORIS data there (Section 1.5). Speculation, explicitly labeled as such: because the PI-GNN is differentiable end to end, a later study could take gradients of total farm energy with respect to turbine coordinates or yaw angle through the trained network, in the same spirit as the gradient-ascent layout optimization Park and Park demonstrated on a simulated 20-turbine farm, where their trained PGNN carried a deliberately poor single-line layout from a normalized farm power of 7.494 to 19.988 after 5,000 optimization steps. This study did not attempt a layout or yaw optimization and did not close any control loop. The Sweetwater layout and turbine controllers were treated as fixed throughout.

Limitations. The layout studied is one farm over only two months of data. The chronological split avoids shuffling weather across the train/test boundary but does not by itself demonstrate transfer to a different site, a different layout, or a different season. The Jensen constants used inside the loss, and inside the PyWake turbine object, are representative values, not an established power curve written by the manufacturer, and were not independently calibrated against Sweetwater’s actual turbine model. The PyWake figure is one freestream of 8 m/s, not a time-march of the 2013 mast. The holdout energy scores scale the closures by upstream SCADA rather than by a PyWake power-curve object. The primary PI-GNN results in Table 2 are two-node diagnostics. The 28-link shared run in Table 3 is weight sharing across many two-and three-node neighborhoods, not simultaneous multi-neighbor inference over every validated upstream of T118 at once, and that all-neighbors-at once run has not yet been scored. No three-dimensional large-eddy or Navier–Stokes solve is reported; it may have been nice to add it as a computation-time/fidelity ratio baseline to compare with the PI-GNN. RMSE and R^2 on a time-ordered holdout do not substitute for a randomized experiment. Curtailment events, icing, or an undetected sensor fault could still sit inside the test tail and were not independently logged or excluded, though these systematic errors were mitigated as much as possible by initial filtering of the SCADA data. The unsolved part of the original engineering problem is to present all validated upstream neighbors of T118 in one graph per timestep, rather than the 28-link shared-weight subset scanned here.

Until that all-neighbors run exists, and the local workstation has the implementation to do so (computation is not a problem), the conclusion this study can support is narrower, but still useful. In summary, pairwise networks can fit a single hub but cannot scan over-

lapping wakes. Layering several pairwise networks is not a neighborhood estimator. A Jensen residual carried in the training loss measurably helps the close pair and barely affects a distant pair. On the same two links, the two-node PI-GNN is closer to the meter than the NOJ and Bastankhah closures, and the gap is larger where the PyWake field still shows a deep wake. A physics-informed graph neural network with one shared weight file can score many Sweetwater wake links in a single iteration.

Literature Cited

- Barthelmie, R. J., Hansen, K. S., Frandsen, S. T., Rathmann, O., Schepers, J. G., Schlez, W., Phillips, J., Rados, K., Zervos, A., Politis, E. S., & Chaviaropoulos, P. K. (2009). Modelling and measuring flow and wind turbine wakes in large wind farms offshore. *Wind Energy*, 12(5), 431–444.
- Bastankhah, M., & Porté-Agel, F. (2014). A new analytical model for wind-turbine wakes. *Renewable Energy*, 70, 116–123.
- Dong, Y., Tang, G., Jia, Y., et al. (2022). Review of wind-farm wake losses. *Energy Engineering*, 119, 1341–1360.
- Echeverri, G. A. G., Adesunkanmi, R., Tamayo, C. R., Kiran, B., & Adel, A. (2026). Spatio-temporal transformer-based modeling of multi-turbine wake effects for wind farm power forecasting. *Energy and AI*, 24, 100715.
- Electric Reliability Council of Texas. (2024). *Fact sheet*. ERCOT.
- Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. *Proceedings of the 34th International Conference on Machine Learning*.
- Hamilton, W. L., Ying, R., & Leskovec, J. (2018). Inductive representation learning on large graphs. arXiv:1706.02216.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- Howland, M. F., Lele, S. K., & Dabiri, J. O. (2019). Wind farm power optimization through wake steering. *Proceedings of the National Academy of Sciences*, 116(29), 14495–14500.
- Jensen, N. O. (1983). *A note on wind generator interaction* (Risø-M-2411). Risø National Laboratory.
- Lu, Q., Cao, Y., Xie, P., Chen, Y., & Lin, Y. (2025). A scalable data-driven surrogate model for 3D dynamic wind farm wake prediction using physics-inspired neural networks and wind box decomposition. *Energies*, 18(13), 3356.
- Park, J., & Park, J. (2019). Physics-induced graph neural network: An application to wind-farm power estimation. *Energy*, 187, 115883.

- Pedersen, M. M., van der Laan, P., Friis-Møller, M., Riva, R., & Réthoré, P.-E. (2019). *PyWake: An open-source wind farm simulation tool*. DTU Wind Energy.
- Porté-Agel, F., Bastankhah, M., & Shamsoddin, S. (2020). Wind-turbine and wind-farm flows: A review. *Boundary-Layer Meteorology*, 174, 1–59.
- Raschka, S., Liu, Y. (H.), & Mirjalili, V. (2022). *Machine learning with PyTorch and scikit-learn*. Packt Publishing.
- U.S. Energy Information Administration. (2024). *Wind explained: Electricity generation from wind*. U.S. Department of Energy.
- Schøler, J. P., Peder Weilmann Rasmussen, F., Quick, J., & Réthoré, P.-E. (2026). Graph neural operator for wind farm wake flow. *Wind Energy Science*, 11(6), 2229–2256.
- Vattenfall. (n.d.). *Horns Rev 1 offshore wind farm aerial photograph* [Photograph]. NOAA Climate.gov.vb c