Wind Energy Forecasting with a Hybrid Quantum Neural Network: Simulator Study and QPU Inference on IBM Quantum Hardware

Superpositions Studio

March 4, 2026

Abstract

Wind energy forecasting can be posed as time-series regression: given recent meteorological conditions, we predict future wind power production. This work reports an end-to-end pipeline consisting of exploratory data analysis, algorithmic design, classical baseline modeling, hybrid quantum–classical training on an ideal simulator, and an inference-only quantum processing unit (QPU) demonstration. Using a dataset of 26,304 observations and eight variables (including the target `Wind_production`), we construct leakage-safe sliding windows of length $W = 24$ and forecast the target $H = 5$ steps ahead. We compare a classical multilayer perceptron (MLP) regressor with a Hybrid Quantum Neural Network (HQNN) that combines an LSTM encoder, a classical bottleneck, and a 9-qubit depth-2 variational quantum circuit layer. On the full held-out test set, the HQNN achieves slightly improved accuracy over the MLP baseline (MAE 0.01714 vs 0.01749). To satisfy the deployment requirement of hardware execution, we additionally run HQNN inference on 64 test windows using an IBM Quantum Heron-class backend with 650 shots, observing limited but measurable degradation relative to ideal simulation due to finite-shot and hardware noise (subset MAE 0.01436 on QPU vs 0.01394 on an ideal simulator).

1 Introduction

Accurate wind power forecasting supports grid stability, reserve planning, and market operations. Because wind production depends on evolving meteorological conditions, many practical formulations treat forecasting as supervised regression on multivariate time series [1, 2, 3, 4, 5]. Alongside classical machine learning and deep learning methods [6, 7, 8], quantum machine learning investigates whether parameterized quantum circuits can serve as trainable nonlinear components within hybrid architectures [9, 10, 11, 12, 13, 14, 15]. Hybrid quantum neural networks (HQNNs) are particularly relevant in the noisy intermediate-scale quantum (NISQ) regime [13], where quantum resources are limited and the quantum component must remain compact.

This paper consolidates the full workflow produced by the platform for the wind energy forecasting use case, including a classical baseline (MLP regressor) and a hybrid model (HQNN). Beyond simulator-first modeling, we include an inference-only QPU experiment to quantify how finite-shot hardware execution affects predictive quality on a small, cost-controlled subset.

2 Dataset and exploratory analysis

2.1 Dataset overview and quality checks

The dataset consists of 26,304 rows and 8 numerical columns: `DHI` (Diffuse Horizontal Irradiance), `DNI` (Direct Normal Irradiance), `GHI` (Global Horizontal Irradiance), `Wind_speed`, `Humidity`,

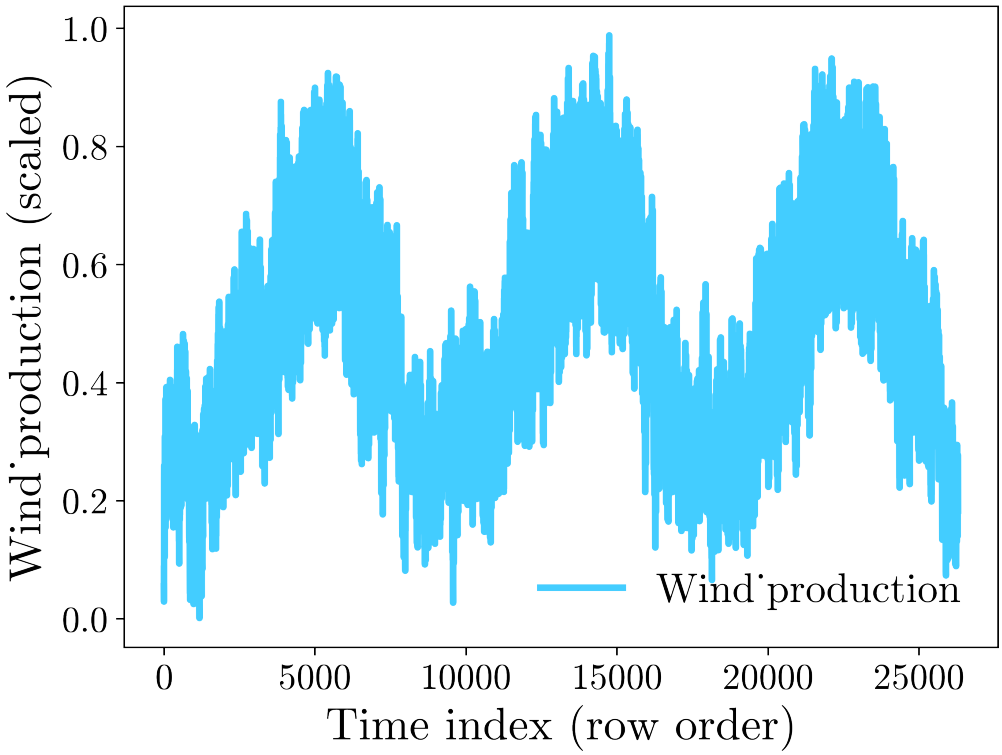

Figure 1: Target series (`Wind_production`) in index order. The absence of timestamps requires interpreting the row index as time.

`Temperature`, `Wind_production` (target), and `Season`. No missing values or duplicate rows were detected.

A notable practical constraint is that the table contains no explicit timestamp column; therefore, we assume that the row index represents chronological order. All splits and windowing operations preserve this order to avoid temporal leakage.

2.2 Feature ranges, regimes, and dependence structure

Most variables appear normalized to approximately $[0, 1]$, but `DHI` takes discrete values in $\{1, 2, 3, 4\}$, suggesting an ordinal/categorical encoding. Several exogenous variables (`GHI`, `DNI`, and `Wind_speed`) exhibit a strong point mass at zero (about 44% of values), indicating regime structure (e.g., night/day or low/high activity periods). In contrast, the target has essentially no exact zeros, making percentage-based metrics such as sMAPE numerically stable with a small ε.

Correlation analysis indicates that `Temperature` is strongly negatively associated with `Wind_production`, while `Wind_speed` and irradiance-related variables (`GHI`, `DNI`) are positively associated. Autocorrelation diagnostics on the target series (using index order) reveal strong persistence at lag 1 and prominent peaks near lags 24 and 168, consistent with daily and weekly periodicity.

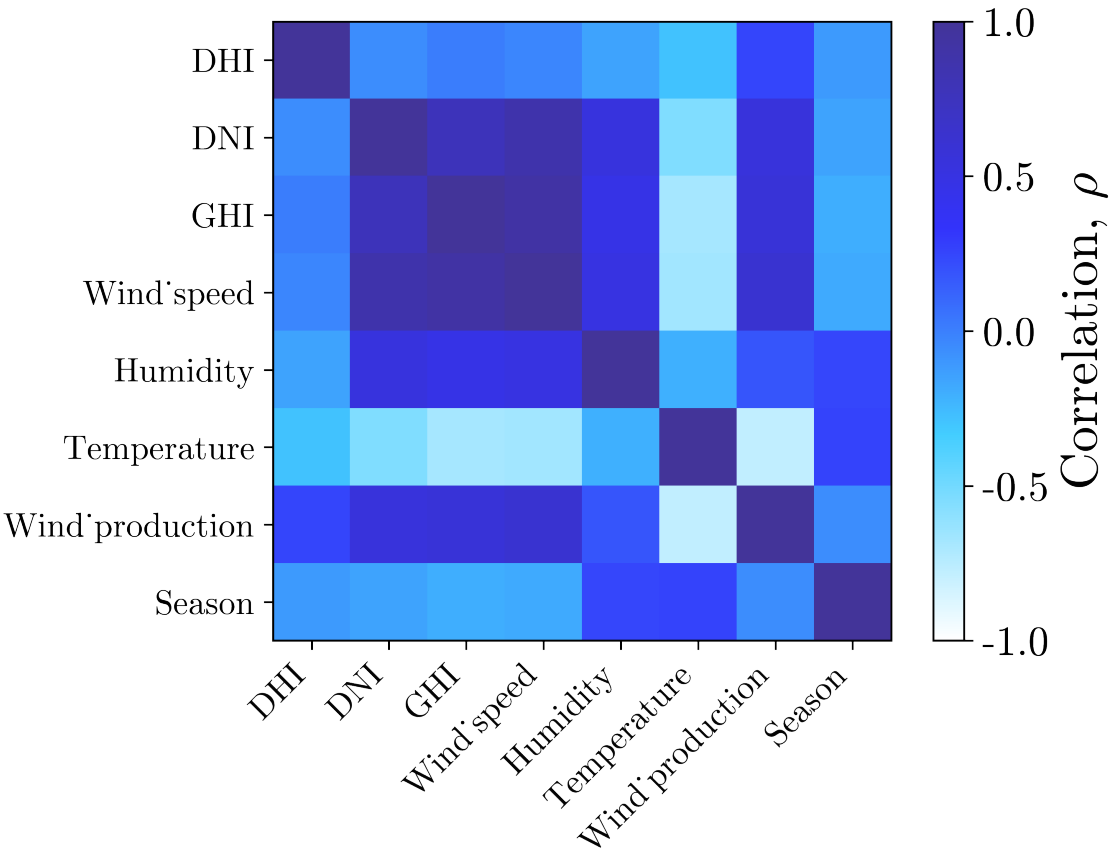

Figure 2: Pearson correlation heatmap between features and target.

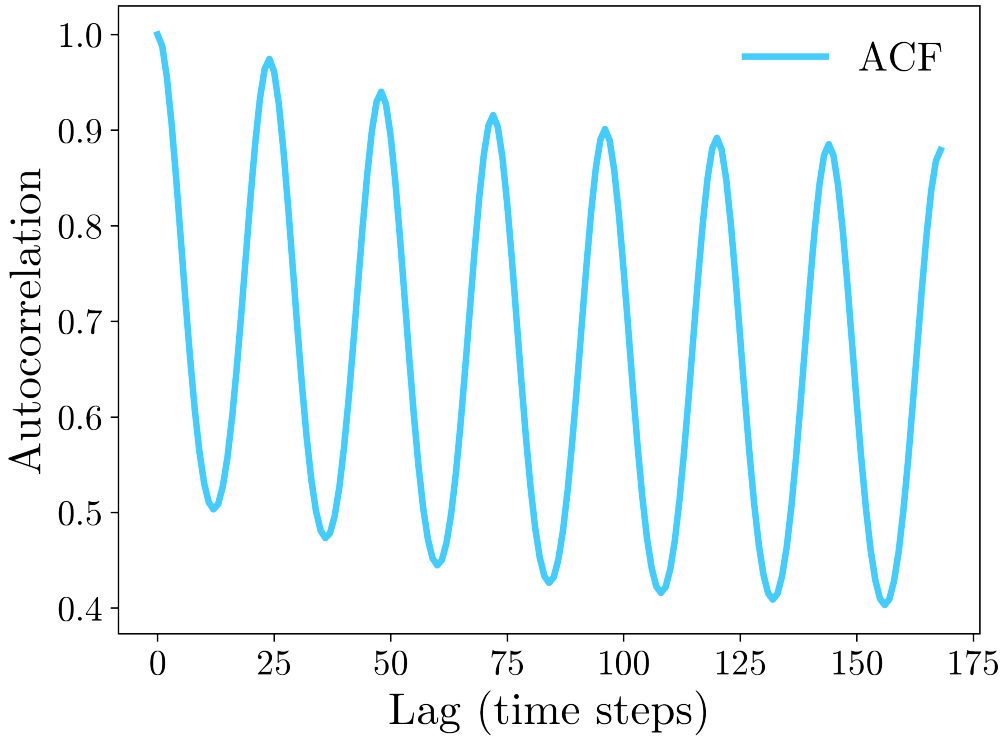

Figure 3: Autocorrelation function (ACF) of the target series in index order, showing strong periodic structure around 24 and 168 lags.

3 Methods

3.1 Forecasting formulation and windowing

Let $\mathbf{x}_t \in \mathbb{R}^d$ denote the feature vector at discrete index t and let $y_t \in \mathbb{R}$ denote wind production. We form supervised samples by extracting sliding windows of length W and predicting at horizon H. With input width $W = 24$, shift (horizon) $H = 5$, and label width 1, each training example uses

$$\mathbf{X}_t = [\mathbf{x}_t, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+W-1}] \in \mathbb{R}^{W \times d}, \qquad \hat{y}_{t+W+H-1} = f_\Theta(\mathbf{X}_t). \tag{1}$$

Because the dataset has no timestamps, t indexes the row order.

3.2 Chronological split and leakage-safe preprocessing

We apply a contiguous chronological split of the raw rows into 70% training, 15% validation, and 15% test partitions (18,412 / 3,945 / 3,947 rows). Sliding-window construction yields 18,384 / 3,917 / 3,919 windows, respectively. Exogenous variables are standardized using `StandardScaler` from scikit-learn [16], fit on the training block only. The target is standardized with a separate scaler, also fit only on training.

To exploit the strong autocorrelation observed in Fig. 3 without leakage, we append the (scaled) past target value to the exogenous features at each time step, so that each input time slice includes both contemporaneous covariates and a lagged target feature. This is a common and effective strategy for autoregressive forecasting [2, 1].

3.3 Baseline model: MLP regressor

The required classical baseline is an MLP regressor (scikit-learn implementation [16]) trained on flattened windows $\mathrm{vec}(\mathbf{X}_t) \in \mathbb{R}^{Wd}$. We use two hidden layers (128 and 64 units) with ReLU activations and Adam optimization [17]. This baseline provides a controlled comparison against the HQNN.

3.4 Hybrid Quantum Neural Network (HQNN)

3.4.1 Architecture

The HQNN is a differentiable hybrid model implemented in PyTorch [18]. A classical temporal encoder (LSTM) maps the input window into a representation that is further compressed by a feed-forward bottleneck. A variational quantum circuit (VQC) layer processes the low-dimensional latent vector and returns expectation values that are concatenated with the classical latent via a skip connection and mapped to the final scalar prediction. This hybrid design follows the variational quantum-classical pattern introduced in early VQE and general VQA formulations [19, 20, 14].

Concretely, the model uses an LSTM with hidden size 24, followed by a multilayer perceptron that maps the flattened LSTM outputs to a $p = 27$ dimensional latent vector. A min–max scaling layer maps the latent coordinates to angles in $[-\pi/8, \pi/8]$ before entering the quantum layer. The quantum component uses $n = 9$ qubits and circuit depth $L = 2$ with Y-rotation encoding, X-rotation variational parameters, a strong entanglement pattern, and Pauli-Z measurements on all qubits, yielding a 9-dimensional vector of expectation values. The final classical head maps the concatenated 36-dimensional vector to a single forecast.

3.4.2 Training protocol

Training minimizes mean squared error (MSE) on standardized targets using Adam with separate learning rates for classical and quantum parameters (2×10^{-3} and 8×10^{-4}, respectively), with 10^{-4}

Table 1: User-defined acceptance criteria (evaluation targets).

Metric	Target threshold
MAE	≤ 0.030
RMSE	≤ 0.050
sMAPE	$\leq 9.0\%$
R^2	≥ 0.85

Table 2: Performance on validation and test splits (full sets). Lower is better for MAE/RMSE/sMAPE; higher is better for R^2.

Model	Split	MAE	RMSE	sMAPE (%)	R^2
MLP	Val	0.01797	0.02321	4.47212	0.98569
MLP	Test	0.01749	0.02310	4.57026	0.98572
HQNN	Val	0.01571	0.02062	3.96119	0.98870
HQNN	Test	0.01714	0.02245	4.30424	0.98651

weight decay on the classical parameters. The end-to-end model is trained by backpropagation through the classical and quantum blocks [7, 21]. Early stopping monitors validation MAE with patience 15 and yields a best epoch of 30 (training stopped at epoch 50). Training and evaluation on the full dataset are performed on an ideal, shot-free simulator backend for the quantum layer.

3.5 Evaluation metrics and acceptance targets

Model performance is reported using MAE, RMSE, sMAPE, and R^2. For a test set $\{(y_i, \hat{y}_i)\}_{i=1}^{N}$,

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \tag{2}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{3}$$

$$\mathrm{sMAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\frac{2|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i| + \varepsilon}, \tag{4}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \tag{5}$$

with $\varepsilon = 10^{-6}$.

The user-defined acceptance criteria are listed in Table 1 and are treated as evaluation targets.

4 Results on an ideal simulator (full validation/test)

4.1 Quantitative comparison

Table 2 reports validation and test performance for the MLP baseline and HQNN. The HQNN achieves improved validation metrics and a modest but consistent improvement on the test set across MAE, RMSE, sMAPE, and R^2. Both models satisfy the acceptance criteria in Table 1.

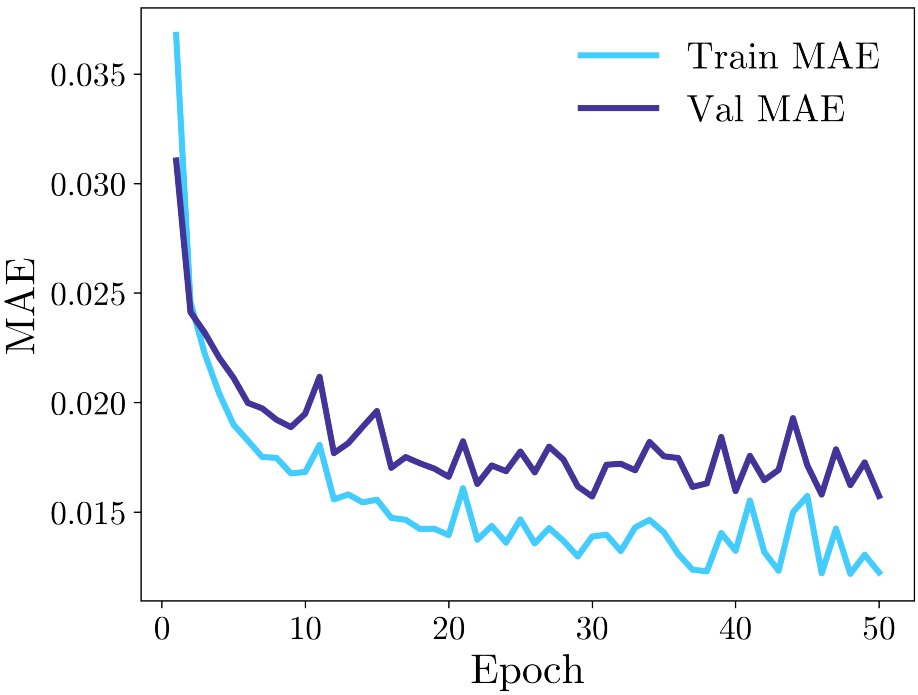

Figure 4: HQNN training and validation MAE across epochs with early stopping (best epoch 30).

4.2 Training dynamics and diagnostic plots

Figure 4 shows the training and validation MAE during HQNN training, illustrating stable optimization with early stopping. Figure 5 visualizes predictions over the tail of the test interval, and Fig. 6 compares parity plots for HQNN and MLP.

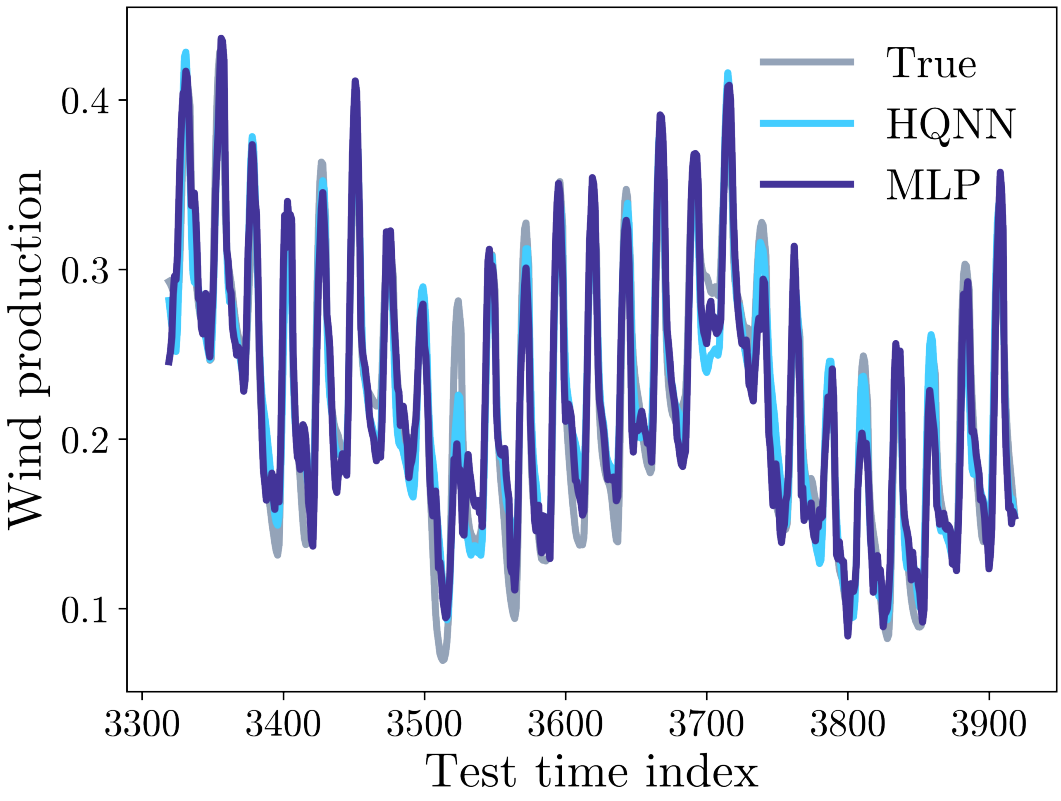

Figure 5: True vs predicted wind production on the test interval (tail segment).

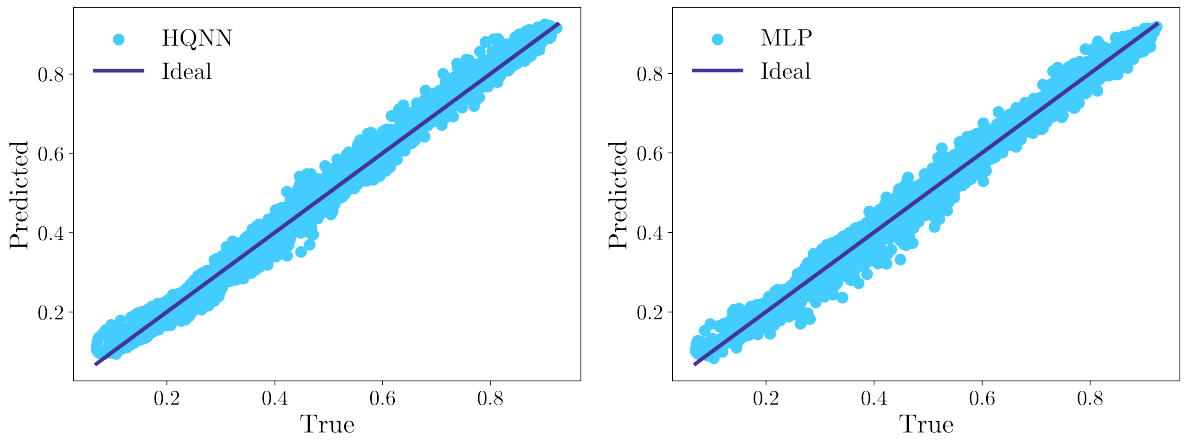

Figure 6: Parity plots on the full test set for HQNN (left) and MLP baseline (right).

Table 3: HQNN inference on a 64-window test subset: ideal simulator vs QPU (650 shots).

Backend	MAE	RMSE	sMAPE (%)	R^2
Ideal simulator	0.01394	0.01625	2.11114	0.97565
QPU (650 shots)	0.01436	0.01662	2.21802	0.97452

4.3 Runtime considerations

The MLP baseline trains in approximately 6.16 s and predicts in about 0.007 s. In contrast, HQNN training on the ideal simulator requires approximately 673 s and inference about 1.60 s. This gap reflects the overhead of repeated quantum circuit evaluations during training and highlights a key trade-off in current HQNN deployments: small accuracy gains may incur substantial computational cost. Scaling and trainability challenges (including barren plateaus) have been widely discussed in the variational quantum algorithms literature [22, 23].

5 QPU execution: inference on IBM Quantum hardware

5.1 Protocol and configuration

To satisfy the requirement of QPU execution under a controlled budget, we perform an inference-only experiment using the trained HQNN weights and the identical preprocessing pipeline. We select the first 64 windows from the chronological test split, run ideal inference (shot-free simulator) and QPU inference with 650 measurement shots, and compute the same regression metrics on this subset.

The QPU execution uses a Qiskit-based backend interface [24] and a parameter-shift differentiation setting (not used for inference). The run is labeled as "IBM Quantum Heron r3"; due to software interface constraints, backend selection is handled internally by the provider wrapper (typically choosing an available IBM backend), and the exact backend identity should be logged for strict reproducibility. This reporting convention is aligned with current IBM hardware-roadmap-oriented workflows where backend generation/family is often tracked at the experiment-design level [25].

5.2 Subset metrics and qualitative comparison

Table 3 reports ideal versus QPU metrics on the 64-window subset. QPU execution shows a small degradation relative to ideal simulation, consistent with finite-shot estimation and hardware noise.

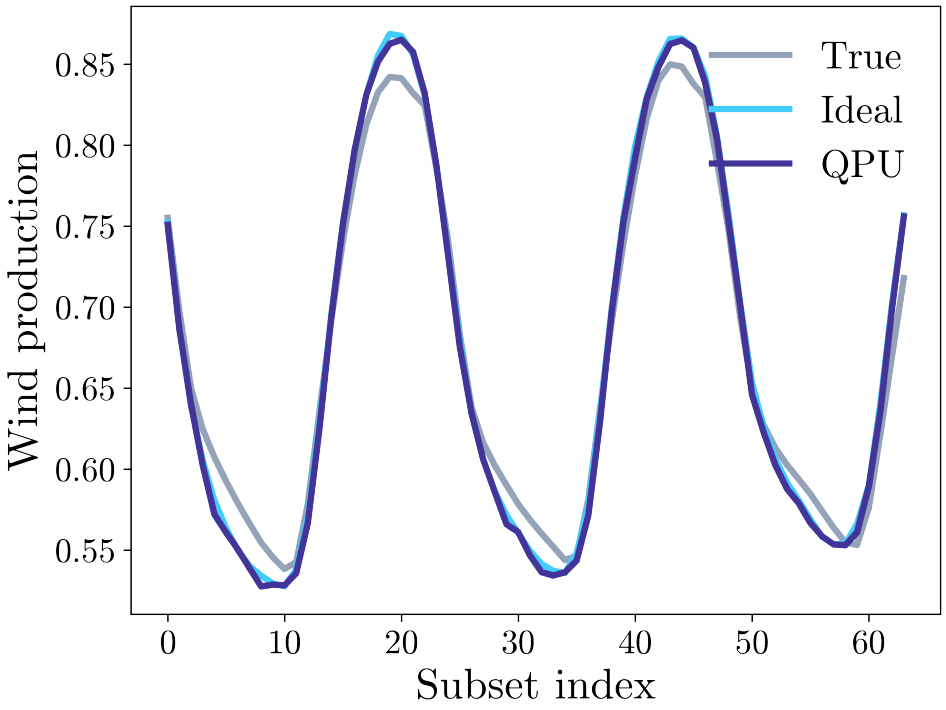

Figure 7: True vs HQNN predictions on the 64-window subset: ideal simulator vs QPU.

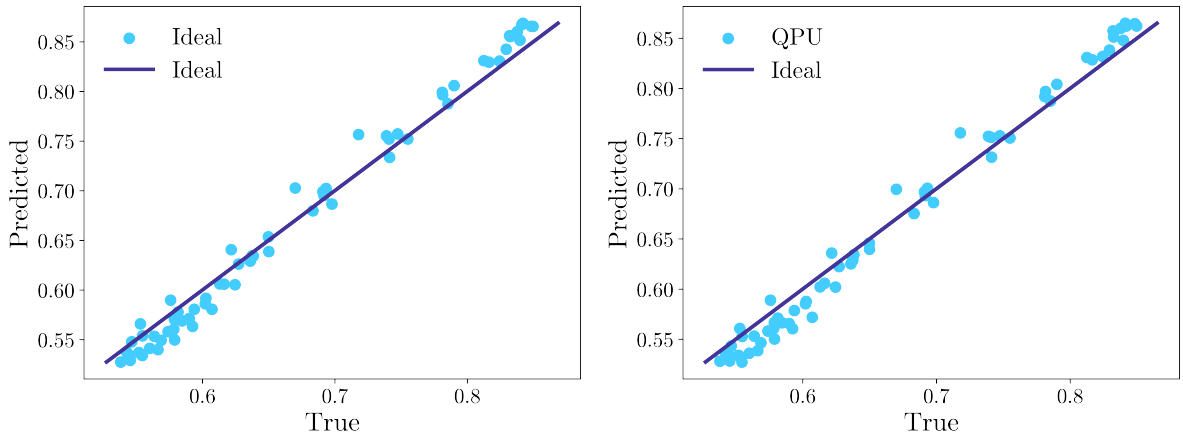

Figure 8: Parity plots on the 64-window subset for ideal inference (left) and QPU inference (right).

6 Discussion

The results demonstrate that a compact HQNN can match and slightly outperform a classical MLP baseline on this dataset, while preserving a leakage-safe evaluation protocol based on chronological splits. The improvement is modest on the full test set, suggesting that most forecastability is already captured by classical modeling when lagged targets are included. The HQNN's main cost is computational: training and inference on a simulator are orders of magnitude slower than the MLP baseline, and QPU execution further increases latency due to job queueing, compilation, and shot-based estimation.

The QPU experiment, while limited to 64 windows, provides a concrete estimate of the performance gap induced by hardware noise and finite shots. The observed degradation is small at 650 shots, but larger subsets or lower shot counts may increase variance. These findings are consistent with broader observations in NISQ-era variational modeling, where expressivity, trainability, and noise must be carefully balanced [14, 22, 23].

7 Future outlook and recommendations

Several directions could improve both scientific rigor and practical performance. Stronger classical baselines (e.g., gradient-boosted trees [26]) would clarify whether the HQNN adds value beyond a simple MLP. Multi-horizon forecasting and probabilistic calibration would better match operational needs. From the quantum perspective, hardware runs would benefit from explicit backend pinning and logging, careful transpilation to the target coupling map, and error-mitigation strategies (e.g., measurement mitigation and noise-aware training). Recent domain studies in renewable forecasting with quantum models also indicate that careful data/feature engineering remains a key determinant of gains [27]. Finally, since simulator cost grows rapidly with qubit count, future HQNN designs should explore more parameter-efficient circuits and problem-informed ansätze.

8 Conclusion

We presented a full wind energy forecasting workflow combining time-series EDA, a classical MLP baseline, and an HQNN with a 9-qubit variational quantum layer. On the full test set, the HQNN achieved slightly better predictive accuracy than the baseline while meeting user-defined acceptance targets. We additionally demonstrated inference on IBM Quantum hardware with 650 shots on a 64-window subset, quantifying a small degradation relative to ideal simulation. Overall, the study illustrates a pragmatic simulator-first path toward QPU experimentation for time-series regression with compact hybrid quantum models.

References

[1] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 3 edition, 2021. https://otexts.com/fpp3/.

[2] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley, 5 edition, 2015.

[3] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, 3 edition, 2016.

[4] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer, 4 edition, 2017.

[5] James D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. https://www.deeplearningbook.org/.

[7] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[8] Sepp Hochreiter and J"urgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[10] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*. Springer, 2018.

[11] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[12] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.

[13] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[14] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3:625–644, 2021.

[15] Stefano Markidis. Programming quantum neural networks on NISQ systems: An overview of technologies and methodologies. *Entropy*, 25(4):694, 2023.

[16] Fabian Pedregosa, Ga"el Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015. https://arxiv.org/abs/1412.6980.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[19] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014.

[20] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[21] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[22] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9:4812, 2018.

[23] Samson Wang, Enrico Fontana, M. Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J. Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications*, 12:6961, 2021.

[24] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2024. https://qiskit.org/.

[25] IBM Quantum. IBM Quantum roadmap and fault-tolerant direction (blog), 2023. https://www.ibm.com/quantum/blog/large-scale-ftqc.

[26] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[27] A. Sagingalieva, S. Komornyik, A. Senokosov, et al. Photovoltaic power forecasting using quantum machine learning. *Solar Energy*, 2025. doi:10.1016/j.solener.2025.114016.