Helping mitigate climate change through efficient reinforcement learning-based wind farm flow control

Elie Kadoche[1,2] **Pascal Bianchi**[1] **Florence Carton**[2] **Philippe Ciblat**[1] **Damien Ernst**[1,3]
[1]Polytechnic Institute of Paris, 19 Place Marguerite Perey, 91120 Palaiseau, France
[2]TotalEnergies OneTech, 2 Place Jean Millier, 92400 Courbevoie, France
[3]Montefiore Institute, University of Liège, 4000 Liège, Belgium
elie.kadoche@totalenergies.com

Abstract

Improving wind farm efficiency is critical for reducing greenhouse gas emissions and scaling renewable energies. One effective approach to increase a wind farm's power output is wake steering, where certain turbines are intentionally misaligned with the wind to enhance downstream airflow and reduce wake losses. However, designing robust, large-scale wake steering controllers remains challenging due to uncertain and time-varying wind conditions. We propose an attention-based reinforcement learning architecture and a carefully designed reward shaping methodology to develop more efficient wake steering controllers. Using a steady-state, low-fidelity simulator, we show that our approach increases energy capture relative to strong baselines, illustrating how machine learning can directly improve renewable energy generation and contribute to climate change mitigation.

1 Introduction

Wake steering Within wind farms, wake effects of upstream turbines reduce the power output of downstream turbines, decreasing the total farm energy production. Wake losses quantify this impact as the percentage of power loss due to wake-induced interference. Greedy control maximizes each turbine's output individually by keeping all turbines aligned with the wind. To mitigate the negative impact of wake effects, wind farm flow control (WFFC), i.e., coordinated turbine control, can be implemented. One method, known as wake steering, consists in misaligning upstream turbines in relation to the wind in order to move their wakes away from the downstream turbines. This is accomplished through yaw control, i.e., active rotation of a turbine's nacelle around its vertical axis. Figure 1 displays a simulation of wake steering applied to a three turbines wind farm with the wind coming from the west. However, as wind farms grow in size, designing robust wake steering controllers in uncertain and time-varying wind conditions is challenging.

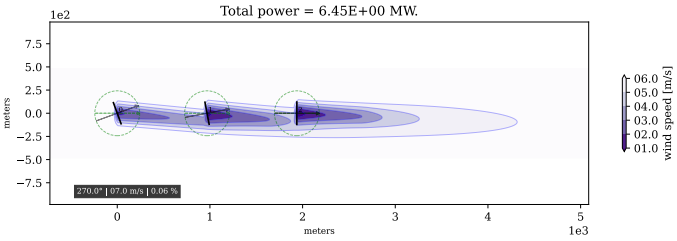


Figure 1: Example of WFFC on a three turbines wind farm.

Tackling Climate Change with Machine Learning: workshop at NeurIPS 2025.

Related works Wake steering is traditionally implemented using lookup tables (LUTs) [1] which fail to adapt to real-time wind dynamics. Model-based approaches offer some adaptability but heavily depend on the accuracy of their underlying wind model. To overcome these limitations, model-free reinforcement learning (RL) has emerged as a powerful alternative to traditional methods [2]. But existing RL-based wake steering methods face significant limitations in terms of scalability, sample efficiency, wind dynamics consideration, and wind conditions generalization. While some studies address one or more of these challenges, none comprehensively tackle all of them together. In this work, we leverage self-attention mechanisms within a single-agent RL policy, demonstrating significant improvements in sample efficiency, learning performance and generalization.

Markov Decision Process The wake steering problem is defined over an episode consisting of 18 discrete time steps, during which the yaw angle of each turbine is adjusted. We consider a wind farm of 19 turbines. The WFFC problem is formalized as a Markov decision process (MDP). At a given time step t, the state is $s_t = (X_{W_t}, X_{F_t}, X_{Y_t})$ where X_{W_t} represents the current, wind conditions (noisy), X_{F_t} represents a wind forecast on the next three time steps (noisy), and X_{Y_t} represents the current absolute orientations of each turbine's nacelle. The action a_t is a vector of each turbine individual yaw setting, i.e., a rotational movement bounded in $[-20, 20]$ degrees due to mechanical constraints of the yaw actuators. Numerical simulations are conducted with FLOw Redirection and Induction in Steady State (FLORIS) [3], a steady-state, low-fidelity simulator developed by National Renewable Energy Laboratory (NREL) and wind data time series are generated using an auto-regressive moving average (ARMA) process of order 1.

2 Reward shaping

Traditional reward function A commonly used reward formulation in the literature is the normalized total farm power output $r_{t+1} = (1/N) \sum_{i=0}^{N-1} (P_t^i / \Phi_t^i)$, with P_t^i the turbine i individual power output and Φ_t^i its theoretical maximum power output at time step t. However, when training a wake steering policy across all wind directions (0-360°), this reward becomes problematic: for directions where wind tracking is nearly optimal, the reward approaches 1, as wake effects are minimal. In contrast, for directions where wake steering is necessary, even the optimal policy yields a reward significantly below 1 due to strong wake losses. Consequently, using this reward over the full directional space biases the policy toward trivial wind tracking, which can achieve high rewards without learning effective wake steering - precisely what we aim to avoid.

Improved reward function We propose a reward function normalizing improvements relative to a baseline and ensuring the agent optimizes wake steering across all conditions. Our reward function is defined as $r_{t+1} = \Delta_{P_t} \mathbf{1}_{\Delta_{P_t} < 0} + \exp(-p\bar{\mathcal{L}}_t) \Delta_{P_t} \mathbf{1}_{\Delta_{P_t} \geq 0}$, with the power difference $\Delta_{P_t} = (P_t - \bar{P}_t)/\bar{P}_t$ and with $\bar{P}_t$ the baseline power output. We use as a baseline a perfect wind tracking controller that does not perform any wake steering. The optimal magnitude of Δ_{P_t} depends on the wake losses: a near-zero ratio can be optimal when wake losses are low (making WFFC unnecessary) but suboptimal when wake losses are high (making WFFC necessary). To address this, we introduce an exponential scaling term, parameterized by $p = 3$, that adjusts the reward based on the baseline wake losses $\bar{\mathcal{L}}_t$. This term ensures a balanced reward across different wind conditions by restricting the power difference when the wake losses are significant. Additionally, if power production falls below the baseline, the agent is penalized with a negative reward.

3 Models

Each model is a policy in an actor-critic framework: given the state s_t, it outputs a critic value v_t and an actor distribution parameterized by turbine-wise von Mises distributions with location $\bar{\mu}_t$ and concentration $\bar{\kappa}_t$. Actions are sampled from these distributions during training, while at test time each turbine's action is set to its mean $\bar{\mu}_t$.

Fully-connected neural network The most commonly used architecture in the literature is a feed forward neural network (FNN)-based model, composed exclusively of fully connected (FC) layers. The input is a single concatenated vector comprising all the state features. However, this architecture

may be inefficient, as it requires the model to simultaneously infer complex spatial and temporal dependencies from a high-dimensional, entangled input vector without explicit structural guidance.

Attention-based neural network We propose an attention-based architecture (Figure 2) to better exploit the multi-modality of the WFFC problem. Inputs are split in four different embeddings. 1) A FNN is used to create the wind embedding E_{W_t}. 2) A FNN is used to create the forecast embedding E_{F_t}. 3) A FNN is used to create each turbine positional encoding $E^i_{pe_t}$. 4) A FNN is used to create each turbine specific embedding $E^i_{Y_t}$ from turbine orientations. The final embedding of each turbine is the sum of all these embeddings. Whereas wind and forecast embeddings are shared between all turbines, positional and turbine embeddings are specific for each turbine. In the context of WFFC, self-attention captures relationships between turbines by identifying which ones are most relevant for yaw control. It allows the model to consider all turbines simultaneously and understand how wake effects propagate across the farm. The multi-head mechanism enhances this by providing multiple perspectives on these interactions.

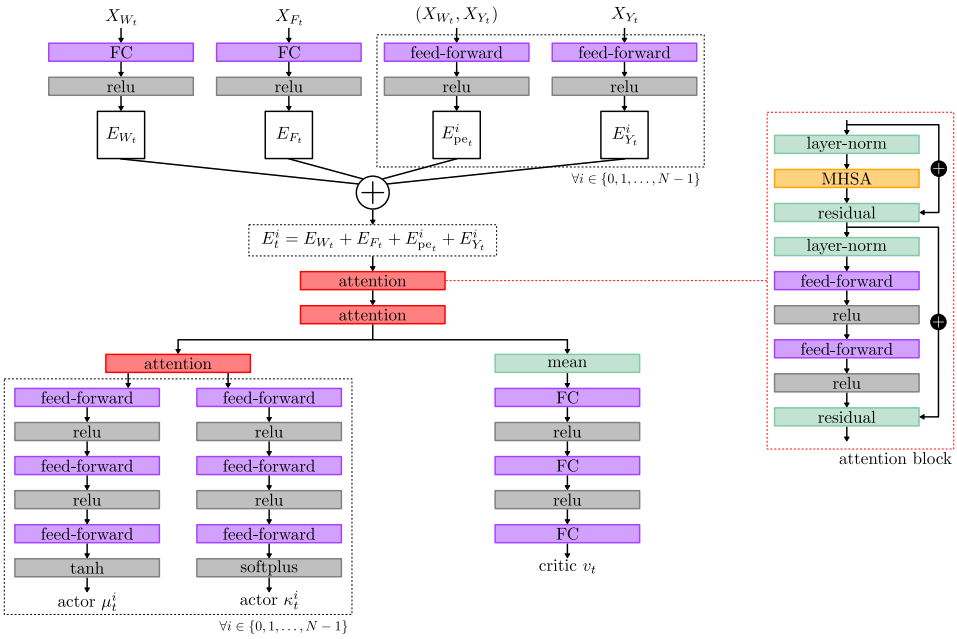


Figure 2: Model architecture, incorporating multi-head self-attention (MHSA) blocks. FC layers refer to standard dense layers applied once to the input, while feed-forward layers apply the same FC layer independently (a) to each turbine's embedding, (b) in attention blocks, (c) and to each turbine's embedding in the actor branch, after the last attention block.

4 Simulations

Training We train each model using a proximal policy optimization (PPO) [4] actor-critic method and generalized advantage estimator (GAE) [5]. At each training step, we simulate 360 independent episodes of 3 hours, resulting in 6,480 time steps. To ensure comprehensive coverage of all possible wind conditions, each of the 360 episodes has a different initial wind direction. More specifically, wind directions are sampled in 1° increments, such that the first episode has an initial direction in $[0, 1]$ degrees, the second episode in $[1, 2]$, etc. It ensures that the entire directional space is covered, speeding up generalization and mitigating sampling biases during training. Each model is trained for 150 steps, corresponding to a total of 972,000 simulated time steps. The training curves, shown in Figure 3, show the superior performance of the attention-based model.

Testing To evaluate the generalization of each solution across all wind conditions, we test them on 360 wind directions, sampled in 1-degree increments. For each direction, we run 10 independent test episodes of 18 time steps, using random seeds not used during training. This ensures that test

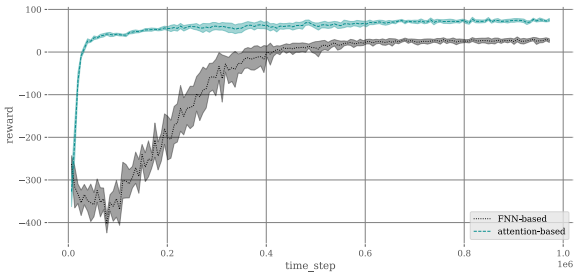
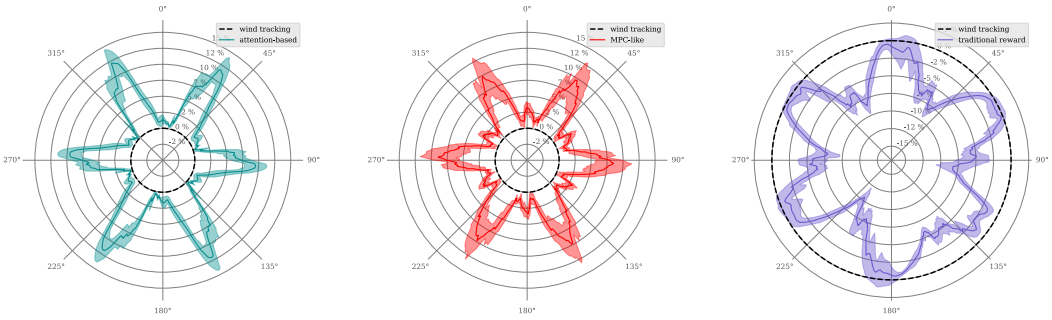


Figure 3: Training curves of each model, showing mean and variance over 10 different seeds. The attention-based model has a much faster and stable convergence that the FNN model.

episodes remain distinct from training and provide comprehensive directional coverage. For each episode, we compute each solution's cumulative power production and quantify its improvement over the standard wind-tracking solution. We then report the mean and variance of these improvements across the 10 seeds for each wind direction and present the results in Figure 4. The attention-based model (sub-Figure 4a) consistently increases wind farm energy production, achieving gains of up to 14 % in high wake-loss scenarios. We use a model predictive control (MPC)-like solution [6] as a strong baseline (sub-Figure 4b). And in sub-Figure 4c, the performance of the attention-based model trained with the traditional reward is displayed.



(a) Attention-based model. (b) MPC-like. (c) Traditional reward.

Figure 4: Performance of each solution relative to a wind tracking policy. The attention-based model achieves performance comparable to the MPC-like solution, but with lower variance and higher gains in strong wake loss conditions. When using the traditional reward, the attention-based model completely fails to learn a wake steering strategy.

5 Conclusion

Our results show that machine learning (ML) can outperform traditional wake steering methods when supported by domain-specific tools such as carefully designed policy models and reward functions. Our attention-based model outperforms a traditional wake steering controller by better capturing wind uncertainty, yielding lower variance and more robust performance across all wind conditions. However, the results remain empirical and rely on simplified, steady-state, low-fidelity wake models. For real-world deployment, future work should incorporate turbine fatigue consideration and validate the approach in higher-fidelity and unsteady flow environments that better capture realistic wind dynamics. As the problem grows in complexity, the strengths of RL - such as its ability to optimize over long horizons, adapt to uncertain dynamics, and operate without explicit system models - should further reinforce its suitability for wake steering control and climate change mitigation.

References

[1] P. Fleming et al. “Field test of wake steering at an offshore wind farm”. In: *Wind Energy Science* 2.1 (2017), pp. 229–239. DOI: 10.5194/wes-2-229-2017. URL: https://wes.copernicus.org/articles/2/229/2017/.

[2] Tuhfe Göçmen et al. “Data-driven wind farm flow control and challenges towards field implementation: A review”. In: *Renewable and Sustainable Energy Reviews* 216 (2025), p. 115605. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2025.115605. URL: https://www.sciencedirect.com/science/article/pii/S1364032125002783.

[3] NREL. *FLORIS. Version 4.2.2*. 2021. URL: https://github.com/NREL/floris.

[4] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[5] John Schulman et al. “High-Dimensional Continuous Control Using Generalized Advantage Estimation”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://arxiv.org/abs/1506.02438.

[6] Elie Kadoche et al. “On the importance of wind predictions in wake steering optimization”. In: *Wind Energy Science* 9.7 (2024), pp. 1577–1594. DOI: 10.5194/wes-9-1577-2024. URL: https://wes.copernicus.org/articles/9/1577/2024/.

[7] J. King et al. “Control-oriented model for secondary effects of wake steering”. In: *Wind Energy Science* 6.3 (2021), pp. 701–714. DOI: 10.5194/wes-6-701-2021. URL: https://wes.copernicus.org/articles/6/701/2021/.

[8] Evan Gaertner et al. “IEA Wind TCP Task 37: Definition of the IEA 15-Megawatt Offshore Reference Wind Turbine”. In: (Mar. 2020). DOI: 10.2172/1603478. URL: https://www.osti.gov/biblio/1603478.

A Appendix

A.1 Wind farm

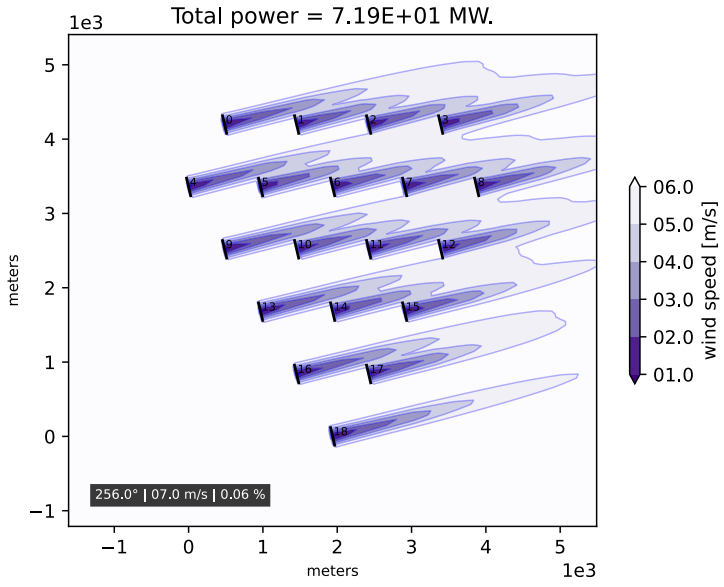


Figure 5: We consider a wind farm of $N = 19$ turbines and a custom diamond layout, with a distance of four turbines diameters between a turbine and its closest neighbors.

A.2 Detailed architectures

Due to the inherent symmetry in the WFFC problem, if an effective solution exists near the lower bound of the action space, a corresponding solution near the upper bound is often equally viable. By modeling actions with a circular distribution, like the von Mises, we ensure that the policy can explore these equivalent solutions efficiently. It promotes a more effective and balanced exploration.

FNN-based model We use 2 hidden layers for the shared branch, with output sizes of 1024 and 4096. The shared actor branch comprises two FC layers with output sizes of 2048 and 256. The $\bar{\mu}_t$ and $\bar{\kappa}_t$ actor branches each contain a single FC layer with output size of N. The critic branch consists of three FC layers with output sizes of 2048, 256, and 1. The FNN-based model has 22,091,047 trainable parameters.

Attention-based model In this work, each embedding layer has an output size of 256. The three attention blocks consist of a MHSA layer with three attention heads and an output size of 256, followed by two feed-forward layers: one increases the size four times and the other restores it. Both actor branches contain three feed-forward layers with output sizes of 128, 64, and 1. The critic branch follows the same structure, with three FC layers of output sizes 128, 64, and 1. The attention-based model has 21,600,771 trainable parameters.

A.3 Hyperparameters

We consider low wind speeds, i.e., between 3 and 10 m/s, because this is where WFFC is the most beneficial for energy production (wake losses have a greater impact). At each time step step $t \geq 1$, we use a simple ARMA process of order 1 to generate wind data. The direction is computed such that $K_t = (\epsilon_t + K_{t-1} + 0.1\epsilon_{t-1}) \mod 360$ with $\epsilon_t \sim \mathcal{N}(0, 9)$. The speed is computed such that $V_t = \epsilon'_t + V_{t-1} + 0.1\epsilon'_{t-1}$ with $\epsilon'_t \sim \mathcal{N}(0, 0.01)$. Noisy wind data is obtained by perturbing the original values with noise sampled from uniform distributions: $K'_t = K_t + \epsilon_K$, where $\epsilon_K \sim \mathcal{U}(-3, 3)$, and $V'_t = V_t + \epsilon_V$, where $\epsilon_V \sim \mathcal{U}(-0.1, 0.1)$. Numerical simulations are conducted with FLORIS [3], a steady-state, low-fidelity simulator developed by NREL. The default Gaussian-curl hybrid model [7] provided by FLORIS is used. The machines are International Energy Agency (IEA) 15 megawattss (MWs) wind turbines [8]. For the PPO, hyperparameters are listed in Table 1.

Table 1: PPO hyperparameters used for each model. Only the learning rate and gradient clipping parameters differ across models. At each training step, the learning rate is linearly interpolated between its initial and final values. Each model requires approximately two hours of training on an NVIDIA Grace CPU and an NVIDIA GH200 Hopper GPU.

Hyperparameter	**FNN-based model**	**Attention-based model**
Training steps	150	150
Discount factor	0.1	0.1
Learning rate (first)	1e-4	1e-5
Learning rate (last)	1e-6	1e-7
Gradient clipping	10	None
GAE λ parameter	0.95	0.95
Entropy coefficient	0.05	0.05
Clip parameter (actor)	0.01	0.01
Clip parameter (critic)	10	10
Value loss coefficient	0.1	0.1
Number of epochs	11	11
Train batch size	6480	6480
Mini batch size	360	360