

A MULTI-AGENT, MULTI-LAYER, ROBUST DRL-BASED WIND FARM CONTROL FRAMEWORK

Project Information	
Acronym - Title	ICONIC – Smart, Aware, Integrated Wind Farm Control Interacting with Digital Twins
Project no.	101122329
Call // Topic	HORIZON-CL5-2022-D3-03 // HORIZON-CL5-2022-D3-03-04
Type of action // Service	HORIZON-RIA // CINEA/C/02
Project duration // starting and end date	48 months // 1/12/2023 – 30/11/2027
Website	www.iconicwind.org
Deliverable Information	
Number & Title	D5-D2.2 A multi-agent, multi-layer, robust DRL-based wind farm control framework
WP number and title	WP2 - AI-based farm-level control and decision-making to improve farm-wide operating efficiency
Author	Hongyang Dong, Yubo Huang, Xiaowei Zhao
Description	A multi-agent multi-layer DRL-based WF control framework. It corresponds to T2.2: A multi-agent, multi-layer, robust DRL framework to maximise farm-wide power production.
Lead Beneficiary	University of Warwick
Type	R: Document, report
Dissemination Level	PU: Public
Status	Final
History of Changes	
Version 0.1	Draft created by Warwick (12 May 2025)
Version 0.2	Draft approved by the TC (23 May 2025)
Version 1	Approved for submission by the General Assembly (30.05.2025)

Copyright

This work is licensed under the [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).



**Funded by
the European Union**

Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the granting authority. Neither the European Union nor the granting authority can be held responsible for them.

TABLE OF CONTENT

1	Preface.....	3
2	Summary	4
2.1	Background and Position within ICONIC.....	4
2.2	Key Contributions Delivered	4
2.3	Simulation Highlights	5
2.4	Closing Remark	5
Annex: Wind Farm Control via Offline Reinforcement Learning with Adversarial Training		

1 PREFACE

The ICONIC project was established to advance the way wind farms are operated by integrating intelligence at farm-, turbine- and component-level through cutting-edge modelling, control, and digital-twin technologies, enabling more efficient operation and improved power production.

Within ICONIC, Work Package 2 (WP2) addresses farm-level decision-making and control. Its remit is to create an AI-driven framework that can coordinate all turbines in a wind farm. Deliverable D2.2, “A multi-agent, multi-layer, robust DRL-based wind-farm-control framework,” documents an important WP2 achievement: an intelligent wind-farm control strategy built on deep reinforcement learning (DRL).

The detailed algorithms, training procedure, and simulation results have already been **published in *IEEE Transactions on Automation Science and Engineering***. The paper is attached to this report. This report therefore serves as a cover note: it outlines the context, clarifies how the framework satisfies the objectives set out in the project plan, and directs readers to the published paper for full technical information. The aim is to give a clear overview of this deliverable while retaining complete traceability for anyone who wishes to examine the methodology and evidence in depth.

Overall, D2.2 shows that the advanced AI concepts envisioned for ICONIC WP2 have been translated into a wind-farm-control solution, ready for the next stages of development and validation within the project.

2 SUMMARY

2.1 Background and Position within ICONIC

The ICONIC project aims to place integrated intelligence at the heart of wind-farm operation, linking component, turbine and farm levels through cutting-edge modelling, digital twin and control technologies. Work Package 2 (WP2) contributes the farm-level layer of that vision: an AI system able to shape the collective behaviour of all turbines. Deliverable D2.2 marks an important output of WP2. It introduces a deep-reinforcement-learning (DRL) framework that realises farm-wide control. The deliverable matches well with the key features as indicated by its name: D2.2 A multi-agent multi-layer DRL-based WF control framework. Moreover, it underpins Milestone MS3 (First version of integrated wind-farm control system ready in simulation).

Full algorithmic details, hyper-parameter settings and validation results of D2.2 have been peer-reviewed and published in *IEEE Transactions on Automation Science and Engineering*. The article is attached and constitutes the technical core of this deliverable.

2.2 Key Contributions Delivered

A Multi-Agent Deep Reinforcement Learning (DRL) Architecture

Each turbine is modelled as an independent agent with its own observation and action spaces. Training follows a centralised-training, distributed-execution (CTDE) paradigm; at run time all decisions are taken locally. This reduces the impact of communication latency and still secures farm-wide coordination through a shared reward definition.

Layered Consideration

- *Training layer.* Offline DRL policies are initialised and refined using data generated by a suite of advanced control methods. These pre-computed datasets provide useful guidance, yet the final offline-trained policy can outperform all these algorithms.
- *Operational layer.* D2.2 delivers the farm-level controller. Its input-output structure is fully compatible with existing wind-farm operational logic and does not require hardware changes. The controller outputs reference signals (e.g., yaw offsets) that turbine-level loops track while considering their own local objectives.
- *Supervisory layer.* Standardised interfaces have been considered to allow higher-level energy-management systems (EMS) to be applied.

Robustness Consideration

The framework was trained and stress-tested on an extensive library of realistic operating scenarios. Scenario sampling covers a wide range of operational conditions; hard constraints on control actions prevent violation of mechanical/load limits; adversarial training episodes further improve generalisation.

Alignment with WP2 and the Corresponding Task.

The corresponding task (Task 2.2) and deliverable D2.2 aims to deliver a multi-agent, multi-layer DRL framework to maximise farm-wide performance while remaining compatible with

turbine-level control and supervisory objectives. The elements above correspond one-to-one with that:

- The multi-agent architecture fulfils the requirement for distributed, scalable control;
- The layered design supplies reference signals to turbine controllers and maintains interfaces for higher-level EMS;
- The robustness measures address the emphasis on dependable operation under realistic conditions.

No deviations from scope or timeline are reported; the deliverable provides the artefact foreseen for WP2 at this stage of the project.

2.3 Simulation Highlights

Below is a concise summary of the main simulation findings; complete data, figures and hyper-parameter info are available in the attached journal article.

- **Simulation setup.** Tests were run on three representative layouts: a 9-turbine grid, a 16-turbine irregular array and the full C-Power offshore farm, using stochastic inflow fields that cover the relevant environmental conditions.
- **Energy-capture improvement.** On average and across all layouts, the developed DRL-based wind farm control method delivered an approximately 16 % higher farm power than the greedy MPPT baseline.
- **Computational efficiency.** One inference step takes < 5 ms on a standard CPU, well inside the 50 ms SCADA cycle. During operation each turbine acts independently, so no inter-turbine communication is needed.

2.4 Closing Remark

Deliverable D2.2 delivers a novel DRL framework for wind-farm control—completed on schedule and aligned with the ICONIC work plan. Full technical details are provided in the attached paper; this cover note records the main outcome and its conformity to plan.

Manuscript version: Author's Accepted Manuscript

The version presented in WRAP is the author's accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:

<http://wrap.warwick.ac.uk/190330>

How to cite:

Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:

The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher's statement:

Please refer to the repository item page, publisher's statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.

Wind Farm Control via Offline Reinforcement Learning with Adversarial Training

Yubo Huang, and Xiaowei Zhao *Member, IEEE*

Abstract—In a wind farm, wakes produced by upstream wind turbines significantly diminish the wind capture of downstream ones, resulting in reduced power generation for the entire farm. Reinforcement learning (RL) control can alleviate the wake effect by enhancing coordination among turbines in arrays. However, training such a collaborative control policy through online RL necessitates millions of interactions with a computational fluid dynamics-based simulator, making it computationally expensive. Wind farm operators possess extensive datasets from past operations, which can tackle the sample generation difficulty. Nevertheless, online RL struggles with the direct use of these offline datasets due to covariate shift and biased value estimation. In this paper, we introduce an offline RL method named Multi-Agent Offline Behavior Imitation (MAOBI) to cope with this challenge. First, by employing f-GANs (Generative Adversarial Networks), MAOBI estimates the divergence between the learning policy and the behavior policy based on samples generated by them. By minimizing this divergence, MAOBI can mimic the behavior policy hidden in the offline datasets. The method then identifies state-action pairs that yield high returns, further improving the control policy. Results demonstrate that MAOBI-trained control policies achieve performance comparable to state-of-the-art online RL methods when deployed in a high-fidelity wind farm simulator.

Note to Practitioners—This study addresses a pressing demand in the wind industry: developing high-performance control systems for wind farms to maximize power generation. Although online RL is a popular approach for training collaborative control policies, its reliance on constructing wind farm environments consumes substantial computational resources, such as a supercomputer, to generate training samples. To overcome this limitation, we propose an offline RL algorithm. This algorithm leverages historical operational data from wind farms to search for optimal control policies, eliminating the need for a virtual environment. Our method offers advantages such as being model-free, communication-free, and capable of real-time response.

Index Terms—Reinforcement learning; Offline learning; Wind farm control; Wake effect

I. INTRODUCTION

VAST carbon emissions from various industrial activities have profoundly altered the climate, subsequently causing the frequent occurrence of a series of extreme weather events such as heatwaves, wildfires, severe droughts, and tropical storms. These natural disasters present significant

challenges to human survival and development. To address these challenges, the world has reached a consensus on carbon neutrality and established detailed timelines to achieve the net-zero objective. In this plan, wind energy plays a pivotal role, leading to a rapid increase in both the number and scale of wind farms globally.

By the end of 2023, the global installed capacity of wind turbines exceeded 1-terawatt (TW) threshold [1]. Following a span of over four decades to achieve the first one TW of installations, the wind industry is poised to accomplish the next TW of installations within the next eight years, demonstrating a significant surge in growth. Furthermore, the installed capacity of the largest wind turbine has reached 16 megawatts (MW), with an 18-MW turbine under construction. As wind farm scales and turbine sizes increase, maximizing the overall power generation efficiency becomes the primary concern since it directly impacts the net profitability of operators [2]. There are three main factors that can affect the power generation of wind farms, including wind farm site selection, wind farm layout optimization, and wind farm control. Among these, site selection and layout optimization jointly establish the theoretical ceiling of power generation for a wind farm, completed prior to its construction. During operation, wind farm control directly determines whether a wind farm can achieve or approximate its theoretical power generation ceiling. Ineffective control systems not only hamper maximum capacity attainment and reduce power output but also increase fatigue loads on blades, potentially damaging turbine structures. Therefore, it is significant to develop a high-performance, farm-level controller to improve the power generation efficiency of wind farms.

Mitigating the wake effect presents the primary challenge in designing a farm-level controller aimed at maximizing power generation [3], as the wake of upstream turbines can greatly reduce the wind capture of downstream turbines, decreasing overall farm output [2]. This also means that the yaw alignment method used for single turbine control is unsuitable for farm-level multi-turbine control because it ignores the wake interaction between turbines. To tackle this problem, wake steering is proposed, which redirects the wake direction of upstream turbines to reduce its effect on downstream ones. Initially, the yaw offset lookup table for each wind turbine in wake steering is obtained from analytical wake models or wind tunnel experimental results [4], making it easy to implement. However, this approach overlooks several important settings and constraints in wind farms, including the distance between wind turbines and the types of wind turbines used. Consequently, it suffers from weak scalability and suboptimal

This work has received funding from Horizon Europe Research and Innovation Actions under the grant agreement No 101122329, the UKRI Horizon Europe Guarantee scheme under the grant No 10095874, and the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 861398

Y. Huang and X. Zhao (corresponding author) are with the Intelligent Control & Smart Energy (ICSE) Research Group, School of Engineering, University of Warwick, Coventry, CV4 7AL, UK. Email: Xiaowei.Zhao@warwick.ac.uk

control results. To further improve wake steering, methods integrating computational fluid dynamics (CFD)-based large eddy simulation for wind and optimization were introduced to calculate the optimal yaw angles at each timestep [5]. In these methods, the optimization loss is associated with farm power output and constrained by yaw actuator response time, control energy cost, safety, and more. Farm controllers designed using this physics-based wind farm model deliver high performance. However, they require a high-precision model to maintain their effectiveness. Additionally, deploying the high-fidelity model demands substantial memory and processing hardware, which can be expensive. In rapidly changing wind scenarios, the slow reasoning speed of model-based optimization may prevent it from obtaining a solution within each control step when applying the predictive control regime. Recently, reinforcement learning (RL) [6] was introduced in wind farm control to optimize a collaborative control policy [7]. Specifically, by mapping the power output of a farm into the reward signal, RL can train agents to collectively explore the high-return control policy via trial and error, achieving the wind farm control objective. RL is powered by deep neural networks, which not only endow it with superior learning ability but are also friendly for controller deployment since they can respond in real-time and have very low memory and inference costs [8]. Moreover, wind farms often comprise diverse turbine types and capacities, posing a notable challenge for control algorithm adaptability. RL adeptly addresses this issue by learning a distributed control policy for each turbine in the farm, considering individual safety and control margins.

However, during the application of RL to wind farm control tasks, we encountered three intractable challenges: (1) Developing a comprehensive wind farm simulator involves considering the aerodynamics of the wind field (Navier-Stocks equations), the structural dynamics of wind turbines, and the electricity transmission module. Constructing such an extensive model consumes a significant amount of time and effort. (2) RL agents must engage in frequent online interactions with the simulator to gather millions of training samples for policy improvement [9]. This interaction process demands high-performance computing facilities due to the computational intensity required to solve complex fluid dynamics. (3) A simulation-to-reality (sim2real) gap exists between real-world wind farms and their digital counterparts [10]. Consequently, a control policy trained in a simulator, despite exhibiting strong performance, may yield subpar results when deployed in an actual farm.

The issues mentioned earlier stem entirely or in part from the development of a wind farm simulator for gathering training data in online RL. In reality, wind farms generate a substantial offline dataset through their regular operations [10], presenting an excellent alternative to replace samples generated by a virtual wind farm simulator. This dataset, directly recorded by various sensors on turbines, not only eliminates sim2real bias during RL agent training but may also encompass empirical collaboration information among turbines actively regulated by the control room. However, historical trajectories cannot be employed to train widely adopted online RL methods, such as Proximal Policy Optimization (PPO [11])

and Deep Deterministic Policy Gradient (DDPG [12]). This limitation arises because these methods require samples generated under the corresponding behavior policy, whereas real data is produced through various methods [13] like Model Predictive Control (MPC), Maximum Power Point Tracking (MPPT), and Wake Steering. In this case, offline RL, which utilizes previously collected data without additional online interactions, is proposed to tackle this problem. And this paper makes the following contributions:

- 1) Constructed an offline RL framework to tackle wind farm control tasks. This frame emphasises the coordination between turbines to mitigate the wake effect in wind farms. Moreover, it adopts the Centralized Training with Decentralized Execution (CDTE) principle for designing actors and critics to avoid the expensive communication among turbines during deployment.
- 2) Developed a novel offline RL algorithm named Multi-Agent Offline Behavior Imitation (MAOBI) to train an efficient farm-level control policy from offline datasets. MAOBI incorporates the distribution divergence measurement achieved by GAN into the policy loss to realize the goal of behavior cloning, while also drawing on the idea of PPO to implement policy improvement.
- 3) Open-sourced a wind farm training dataset¹ containing samples (trajectories) generated by MPPT, Wake Steering, MAPPO (Multi-agent PPO), HAPPO (Heterogeneous-Agent PPO), MAIPO (Multi-agent inductive policy optimization), and the random policy for offline RL. In comparison to the training curves of other RL algorithms, the proposed MAOBI achieved performance similar to the state-of-the-art online MAPPO that needs an accurate simulator during training. The test results of the control policy trained by MAOBI show the turbines have learned how to mitigate the wake effect in wind farms, further increasing the farm's power output.

II. LITERATURE REVIEW

A. Offline Reinforcement Learning

Offline RL, also known as batch RL or data-driven RL, is a paradigm that learns the control policy directly from a previously collected dataset rather than through the interactions with a simulator. In 2005, Ernst et al. [14] firstly introduced the batch mode in supervised learning to RL, explicitly building an offline dataset to train the Q function. In the same year, Riedmiller [15] proposed the core principle of offline RL: storing and reusing previous transition experiences in RL. Then, Fujimoto et al. [16] combined Q-learning with VAE (Variational Auto-Encoder) to facilitate the behavior clone ability of offline RL. Wu et al. [17] presented an offline RL method named Behavior Regularized Actor Critic (BRAC), which achieved excellent results in the Mujoco benchmark environment. Levine [18] demonstrated that many techniques in off-policy RL can be transferred to offline RL, such as importance sampling [19], policy constraint [20], uncertainty estimation [21]. Kidambi et al. [22] designed the MoRel

¹<https://github.com/Yubo-Huang/maobi/tree/main/data>

framework which aims to learn a model from the offline data set and then uses this model to train a policy.

B. Generative Adversarial Networks

Generative Adversarial Networks (GAN), were first proposed by Goodfellow [23] to solve the generative modeling problem via an adversarial process. Afterwards, InfoGAN [24] was presented to facilitate the disentangled representations of GAN. Meanwhile, Nowozin et al. revealed that the loss function of GAN essentially represents a divergence metric between the real probability distribution and the generated probability distribution, and then designed f-GAN to simplify the optimization procedure of classical GAN [25]. However, the convergence of GAN during training presents a great challenge, WGAN [26] was introduced to ease the vanishing gradient problem encountered in training. Zhu et al. [27] proposed CycleGAN, which achieved image-to-image translation. Fundamentally, the actor or policy network in RL serves as a generative model. The optimization process of GAN can train the actor to generate samples similar to the data stored in offline datasets, essentially allowing the actor to produce outputs akin to real samples.

C. Wind Farm Control

Wind farm control has truly emerged as a burgeoning research field in the last decade due to the rapid development and vast investment of renewable energy during this period. In the first decade of the 21st century, wind turbines were normally installed on onshore bases. The distribution of wind turbines was relatively sparse. In this case, the control for each turbine could be considered greedy, implying diminishing the turbine yaw misalignment with the wind direction to capture maximal energy. The combination of MPPT [28] and yaw correction [29] was the widely adopted method in controlling a single turbine. Additionally, plenty of studies focused on reducing the fatigue load of blades and mitigating the vibration of turbine rotor [30]. With the development of offshore wind energy, numerous large-scale wind farms have been constructed in near-coastal seas [31]. To reduce construction costs, floating wind turbines are installed in compact arrays. The wake effect of upstream turbines can impact the wind capture of their downstream counterparts, making the greedy control strategy unsuitable for offshore wind farms [3]. To address this issue, various wake steering methods [32] and power derating methods were proposed to mitigate the wake effect. The development of wind farm control can be categorized into the following four threads.

Wind farm controllers were initially developed based on analytic wake models such as the Jensen wake model and the Gaussian wake model. Representative code bases implementing these models include FLORIDyn [33], FLORIS, and others. In 2013, Marden et al. proposed using a game-theoretic algorithm to optimize a wind farm controller [34]. In 2016, Gebraad et al. modified this method and implemented a model-based version using FLORIS [35]. Additionally, leveraging FLORIDyn [36], Gebraad et al. designed a model predictive controller for the yaw system to achieve wake redirection.

In 2021, Eric et al. calculated a yaw offset lookup table based on FLORIS [4] for wake redirection using preview information. However, field tests conducted by Fleming et al. demonstrated that FLORIS was not tuned to predict near-wake region behavior [37], which may affect the performance of the designed controllers.

Researchers in wind energy have also proposed leveraging high-fidelity CFD models to optimize wind farm controllers, including SP-Wind, SOWFA [38], WFSim [39], and others. In 2015, Churchfield et al. began computing a yaw offset strategy for wake redirection using SOWFA [5]. In 2018, Munters and Meyers designed a centralized receding horizon optimal controller with SP-Wind [40]. In 2022, Broek et al. developed a model-based predictive wind farm controller using free-vortex wake models [41] to mitigate the wake effect while decreasing the yaw actuator usage. Due to the heavy computational load of CFD, these methods place significant demands on memory and computing resources.

To bridge the sim2real gap caused by the above model-based methods, wind field and wind tunnel experiments were integrated to design wind farm control systems. In 2016, Park and Law developed a game-theoretic-based controller to achieve both wake steering and power de-rating objectives [42]. This controller was further fine-tuned through wind tunnel experiments completed in 2017 [43]. Meanwhile, Fleming et al. conducted field tests for their wake steering methods at an offshore wind farm in China [32]. In 2022, Howland et al. used LiDAR wind data collected from field experiments to refine their analytic wake model and design a predictive wind farm controller [2]. These control systems have significantly boosted the power generation of real-world wind farms, but their development and deployment costs are quite high.

Recently, to reduce deployment costs, online RL, which learns control policy by interacting with a simulator or real environments, was introduced in wind farm control. In 2020, Zhao et al. trained a cooperative control policy via RL to mitigate the wake effect [44]. In 2022, Venkata leveraged RL to control a wind farm, maximizing energy while minimizing turbine fatigue damage [45]. In 2023, Dong et al. integrated RL and transfer learning to scale the policy trained for a small-scale farm to larger ones [46]. In 2024, Huang et al. designed a multi-objective RL control system by carefully tailoring the reward function [47]. However, online RL requires a high-fidelity wind farm model to optimize the control policy while building such a simulator is expensive [6]. Therefore, in this paper, we first introduce offline RL to address this problem.

III. PRELIMINARIES

A. Basic notions in wind farm control

For the operation of wind farms, the primary control objective is to maximize the power generation. And our task is to design the control system to achieve this objective. The hierarchical control system consists of two layers: the high-layer farm-level controller and the underlying turbine controller. The farm-level controller acts as a group leader, allocating control commands for the controllable substructures of each

turbine in the farm based on the wind farm state. Subsequently, the turbine controllers execute the control commands and provide feedback to the farm-level controller. In this paper, we focus on developing a model-free, offline RL-based farm-level controller to enhance power generation by mitigating the wake effect. Before delving into the detailed control regime, we'd like to explain several basic concepts in wind farms.

- **Environment state** s is a vector or matrix constituted by the information output by various sensors installed in wind farms, including the wind speed and direction measured by LiDAR, the rotation velocity and accelerated velocity of the rotor and generator, the state of the transmission, the load on the blade and beam, the frequency of the grid, the voltage of the connected bus, etc. The turbine observation o is a part of the entire environmental state and typically includes data measured by self-installed sensors. In this paper, we assume that each wind turbine has no communication with others, and thus it cannot obtain the observations of its counterparts.
- **Control action** a includes variables such as the pitch angle of blades, the yaw angle of the nacelle, the generator torque, and the brake torque of shafts. For offshore floating wind turbines, the spring constant, damping values, and the external force of the tuned mass damper (TMD) can also be controlled to reduce vibrations.
- **Reward** r is a signal that evaluates the control effect of action a under state s . Therefore, the reward function is designed based on our control objective. It encourages agents to explore the action space to find the best actions that can maximize the future return (where return is the total rewards obtained in a control episode).
- **Control policy** π is composed of a series of sub-policy π^i , each consuming turbine observations o^i and emitting optimal control actions (a^i) to maximize the return, thereby achieving the control objective. The purpose of this paper is to use offline RL to optimize such a farm-level collaborative policy π to mitigate the wake effect.

B. Online reinforcement learning-based wind farm control

From the aforementioned introduction, the core task in designing the controller system for wind farms is to determine the optimal control policy to achieve our control objective [48]. To utilize RL to solve this optimal control problem, we should first standardize the control task as a Markov decision process (MDP), denoted by a tuple $\langle \mathcal{M}, \mathcal{S}, \mathcal{A}, r, \rho_0, \gamma \rangle$, where \mathcal{S} and \mathcal{A} are the state space and action space ($s \in \mathcal{S}, a \in \mathcal{A}$), respectively. $\mathcal{M} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S} \in \mathbb{R}$ is the model (transition probability distribution) of the environment. r is the reward function, ρ_0 is the distribution of the initial state s_0 of the environment², and $\gamma \in [0, 1]$ is the discount factor used for calculating the cumulative reward (return).

Within the RL optimization framework, we regard each turbine in the wind farm as an agent. Each agent possesses a policy network (π^i , also known as the actor) and a value network (V^i , also referred to as the critic) used for estimating

the expected return at state s and thereafter following policy π . These networks are initialized randomly. At time t , the wind farm state is s_t and the private observation of turbine i is o_t^i . Based on o_t^i , the policy $\pi_t^i : \mathcal{O}^i \times \mathcal{A}^i \rightarrow [0, 1]$ samples an action a_t^i for turbine i . The underlying controller receives this control command and then execute it. Subsequently, the environment transitions to its next state s_{t+1} based on the transition model \mathcal{M} , while offering a reward signal r_t^i to turbine i . The task of RL algorithms is to search for a policy $\pi = \pi^1 \oplus \pi^2 \oplus \dots \oplus \pi^n$ (where n is the number of turbines in a wind farm) that can maximize the future return of the team

$$J(\pi) = \mathbb{E}_{(s_t, a_t) \sim \rho_0, \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (1)$$

During the k -th interaction between the wind farm and turbines, we can collect a batch of trajectories $\mathcal{D}_k = \{\tau_i = (s_i, a_i, r_i, s_{i+1}), i \in \mathbb{R}^+\}$ by running the policy π_k . Now, we can transform the optimization in Eq. 1 into maximizing

$$J(\pi_k) = \mathbb{E}_{(s_t, a_t) \sim \pi_k} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (2)$$

However, it is impossible to directly use Eq. 2 to optimize the policy π_t since it is of infinite horizon. To cope with this issue, Schulman et al. [11] proposed a surrogate objective to approximate it to optimize the control policy π .

$$J(\pi) = \frac{1}{|T|} \sum_{t=0}^T \min \left(\frac{\pi(a_t, s_t)}{\pi_k(a_t, s_t)} G(s_t, a_t), g(\epsilon, G(s_t, a_t)) \right) \quad (3)$$

Where T is the total time steps of an episode τ , G is the advantage function. and g is the clamp function. Eq. 3 is the single-agent optimization version. For multi-agent settings, the surrogate objective can be decoupled as

$$J(\pi^i) = \frac{1}{|T|} \sum_{t=0}^T \min \left(\frac{\pi^i(a_t^i, o_t^i)}{\pi_k^i(a_t^i, o_t^i)} G^i(s_t, a_t), g(\kappa, G^i(s_t, a_t)) \right) \quad (4)$$

The clamp function $g(\kappa, AG^i)$ is defined as

$$g(\kappa, G^i) = \begin{cases} (1 + \kappa)G^i, & G^i \geq 0 \\ (1 - \kappa)G^i, & G^i < 0 \end{cases} \quad (5)$$

In multi-agent settings, $\kappa = 1 - (1 - \epsilon^{\frac{1}{2}})^{\frac{1}{n}}$, where $\epsilon = 0.2$, and n is the number of agents, and G^i is defined as

$$G^i(s_t, a_t) = r^i(s_t, a_t) + \mathbb{E}_{s_{t+1}} \gamma V^i(s_{t+1}) - V^i(s) \quad (6)$$

The value network V^i of agent i is trained to minimize

$$J(V^i) = \frac{1}{|T|} \sum_{t=0}^T (r_t^i + \gamma V^i(s_{t+1}) - V^i(s_t))^2 \quad (7)$$

This frame is named Proximal Policy Optimization (PPO), which ensures the monotonous improvement of π^i after each update. It is worth mentioning that, in the above multi-agent setting, the input of the policy network—ultimately deployed in wind farms—is the local observation o of the corresponding turbine [49], while the input of the value network, which functions only during training, is the global state s of the wind

²In this paper, unless otherwise stated, the subscript and the superscript denote the time step and the index of an agent, respectively.

farm. This approach is known as Centralized Training with Decentralized Execution (CTDE) in RL, and it proves beneficial for deploying the control policy as it avoids real-time communication between turbines during operation. Derived from the preceding introduction, we can outline the strengths of RL over alternative approaches for optimal control: (1) model-free, (2) communication-free, and (3) agent collaboration.

C. Why is online RL far from sufficient?

RL employs the data collected from the interaction between agents and the environment (simulator) to train a control policy. However, the construction of such a simulator is time-intensive, and its execution proves costly due to the high requirements for computation and memory resources. We will delve into this matter in-depth by introducing the wind field and turbine models, while also estimating their associated computational and memory expenses.

1) *Wind field dynamics*: To describe the dynamics of velocity \mathbf{v} , pressure p , temperature T , and density ρ of the moving fluid in a wind farm, we employ the renowned three-dimensional Navier-Stokes (N-S) equations in this paper—a set of coupled differential equations

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} + \nabla \cdot \tau_M + \frac{1}{\rho} \nabla p - \mathbf{F} = 0 \quad (8)$$

where $\mathbf{v} = (v_x, v_y, v_z)^T$, p , ρ , and τ_M represent the wind velocity, the air pressure, the air density, and the subgrid-scale model at each cell, respectively; $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$. $\mathbf{F} = (f_x, f_y, f_z)^T$ is the force exerted by turbines installed in the wind farm.

2) *Wind turbine servo dynamics*: For the i -th wind turbine, the dynamics of the rotor speed depend on its rotor inertia J_r^i , rotor torque T_r^i , low-speed shaft torque T_{ls}^i and rotor external damping B_r^i

$$\dot{\omega}_r^i = \frac{1}{J_r^i} (T_r^i - T_{ls}^i - B_r^i \omega_t) \quad (9)$$

And the low-speed shaft torque is determined by

$$T_{ls}^i = K_{ls}^i (\delta_t^i - \delta_{ls}^i) + B_{ls}^i (\omega_t^i - \omega_{ls}^i) \quad (10)$$

where K_{ls}^i and B_{ls}^i represent the low-speed shaft stiffness and damping, respectively.

The generator dynamics are governed by the difference between the high-speed shaft torque T_{hs}^i and the braked electromagnetic torque $B_g^i \omega_g^i + T_{em}^i$

$$\dot{\omega}_g^i = \frac{1}{J_g^i} (T_{hs}^i - T_{em}^i - B_g^i \omega_g^i) \quad (11)$$

where ω_g^i is the generator speed.

For the gearbox with a ratio n_g^i , the following relation holds

$$n_g^i = \frac{T_{ls}^i}{T_{hs}^i} = \frac{\omega_g^i}{\omega_{ls}^i} = \frac{\delta_g^i}{\delta_{ls}^i} \quad (12)$$

The output power of turbine i is

$$p^i = T_{em}^i \omega_g^i \quad (13)$$

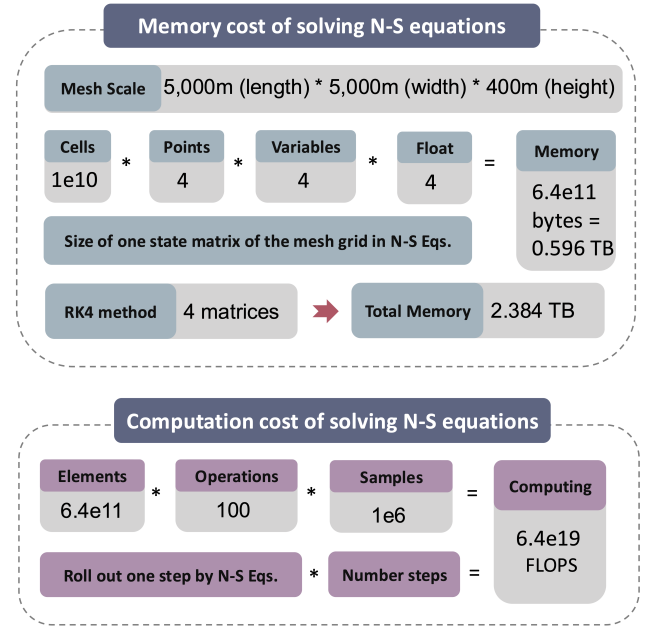


Fig. 1. Memory and computation expenditure for solving the wind farm dynamics based on N-S equations. In the top, cell, point, variables, and float denotes the number of cells in the mesh, the number of solution points in one cell, the wind speed and pressure variables, and the float32 data type, respectively. In the bottom, elements and operations represents the size of the state matrix and the approximate number of operations of one byte should be completed in one time step, respectively.

3) *Expenditure of operating the wind farm model*: Running the presented high-fidelity wind farm model (Eqs. 8-13) to calculate a high-performance control policy is extremely expensive in both calculation and memory. Let consider a wind farm with dimensions of 5,000m in length, 5,000m in width and 400m in height (Fig. 1). We use the grid cell with a size of 1m * 1m * 1m to mesh this wind farm, resulting in $1e^{10}$ cells. According to N-S equations, there are four variables (velocity and density) representing the state of one point within a cell. Suppose that we use a second-order scheme, and there are 4 characteristic points in a cell. For one variable (single-float type), we need to allocate 4 bytes of memory space to represent it. The state matrix of this wind farm grid needs to occupy $1e^{10} * 4 * 4 * 4 = 6.4e^{11}$ bytes (equivalent to 0.64TB) memory. When we utilize RK4 to solve the N-S equations (Eq. 8), there are more than four matrices of this size. This implies that the memory requirement is in the 3 TB tier, while current industrial-grade GPUs generally offer less than 100GB of memory (Table I). A common strategy to address this problem is to partition the meshed grid into smaller pieces, but this inevitably leads to a significant increase in the computation time.

To collect enough samples for training, RL agents necessitate to interact with the wind farm simulator more than $1e7$ times. The total computational workload for this process, measured in terms of arithmetic operations, amounts to approximately $6.4e^{11} * 100 * 1e6 = 6.4e^{19}$ FLOPS. For A100 and H100 GPUs, the time cost is approximately $6.4e^{19} / 19.5e^{12} \approx 3.27e^6$ seconds ≈ 909.3 hours and 355.6 hours, respectively

TABLE I
SPECIFICATIONS OF NVIDIA A100 AND H100 GPUS

GPU	FP32 (TFLOPS)	FP64 (TFLOPS)	Memory (GB)	Bandwidth (TB/sec)
A100 PCIe	19.5	9.7	40	1.6
H100 PCIe	51	26	80	2.0

(Fig. 1). Please note that we ignore the factors of IO cost, communication delay, and data checks in our calculation. Thus it is a conservative estimation.

IV. OFFLINE REINFORCEMENT LEARNING-BASED WIND FARM CONTROL

In Subsection III-C3, we demonstrated the expensive modeling, memory, and computation expenditures associated with leveraging online RL to train a wind farm control policy. The rationale behind incurring such a substantial cost is to generate the requisite number of training samples for online RL. In real-world scenarios, sensors installed on turbines record vast amounts of data during the operation of wind farms controlled by various methods. A straightforward idea is to use this data to train RL agents, rather than relying on trajectories gained from expensive interactions. Levine et al. [18] concluded that directly using online RL algorithms to train control policies based on offline data leads to poor results. To tackle this problem, this section will propose an offline model-free RL algorithm to optimize a low-cost and effective control policy for wind farms.

A. Offline behavior cloning

Let $\mathcal{D} = \{\tau_i = (s_i, a_i, r_i, s_{i+1}), i \in \mathbb{R}^+\}$ be the offline dataset collected during the operation of a wind farm manipulated by various controllers such as MPPT and MPC. The merged control policy contained within \mathcal{D} is called the behavior policy $\pi_b = \bigoplus_i^n \pi_b^i$ that we can access to the samples generated by it but its specific parameters are not available. In the offline behavior cloning stage, the learning objective of our target policy $\pi_\theta = \bigoplus_i^n \pi_{\theta^i}$ is to imitate the behavior policy π_b . To this end, we can use the distance or divergence function D_f to measure the similarity of the two policies³, and behavior cloning can be accomplished by minimizing D_f

$$\begin{aligned} \min \quad & \mathbb{E}_{s_t \sim b} D_f(\theta^i(\cdot|o_t^i), b^i(\cdot|o_t^i)) \\ & = \mathbb{E}_{s_t \sim b} \mathbb{E}_{a_t^i \sim \theta^i} D_f(\theta^i(a_t^i|o_t^i), b^i(a_t^i|o_t^i)) \end{aligned} \quad (14)$$

The corresponding pseudocode of behavior cloning is shown in Algorithm 1 in Supplementary V. During the training process, we can acquire the parameters θ^i of the target policy π_{θ^i} , but those of the behavior policy are invisible. In this scenario, we cannot directly use Eq. 14 for behavior cloning. Instead, based on the samples generated by the target and behavior policies, GAN is capable of estimating their divergence (Fig. 2), which is a key component in this paper. Specific details are introduced in Subsection IV-C.

³For convenience, we short π_{θ^i} and π_b^i as θ^i and b^i , respectively.

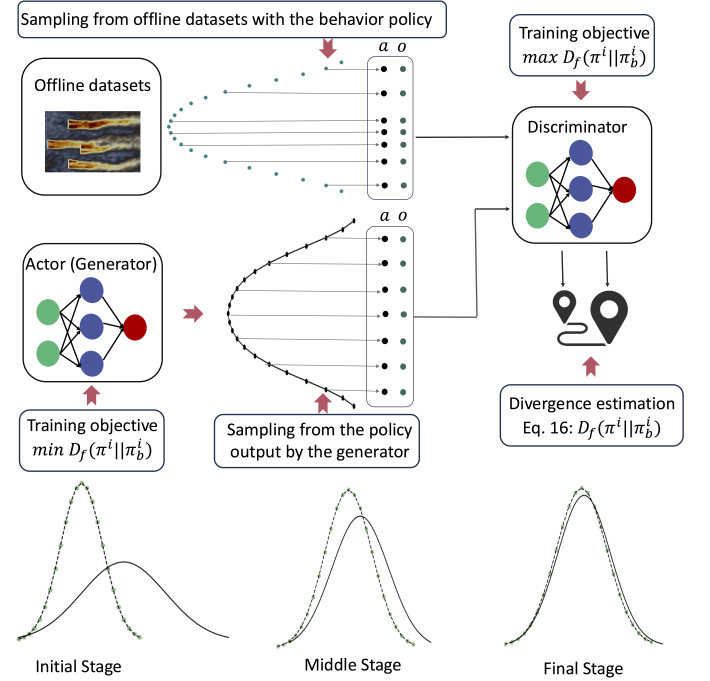


Fig. 2. GAN-based behavior cloning in offline RL. Left: utilizing GAN to estimate the divergence between two probability distributions based on samples generated by them. Right: the change in divergence between the generated policy distribution and the ground-truth policy distribution during training.

B. Offline behavior selection

As mentioned earlier, samples in the offline dataset are generated by various controllers, with the possibility that some are high-performance, while others may have lower quality. During the behavior cloning stage, when RL encounters two behavior pairs (o_t^i, \bar{a}_t^i) and (o_t^i, \hat{a}_t^i) with discrepant rewards⁴, under the gradient calculated by them, agents tend to adopt a compromise. However, we expect them to learn the high-reward behavior while avoiding the lower-quality behavior if there is a conflict (behavior selection). For the behavior selection purpose, we can borrow the idea presented in PPO, implying selecting the action a_t^i that has high advantage at s_t . Then the objective L of policy π_{θ^i} for behavior selection is:

$$\begin{aligned} L(\tau_t) &= \frac{1}{|T|} \sum_{t=0}^T J_{\theta^i}(s_t, a_t) - \alpha D_f(\theta^i(\cdot|o_t^i), b^i(\cdot|o_t^i)) \\ J_{\theta^i}(s_t, a_t) &= \min \left(\frac{\pi_{\theta^i}(a_t^i, o_t^i)}{\pi_{\theta_{old}^i}(a_t^i, o_t^i)} G^i(s_t, a_t), g(\kappa, G^i(s_t, a_t)) \right) \end{aligned} \quad (15)$$

where α is the temperature coefficient used for balancing policy improvement and cloning, which can be linearly decaying during training. At each iteration, this policy gradient increases the probability $\pi(a^i|o^i)$ if the advantage value G^i associated with the state-action pair (o^i, a^i) is positive. Otherwise, $\pi(a^i|o^i)$ is minimized, reducing the likelihood of selecting action a^i at observation o^i in subsequent iterations. With

⁴please note this is only an example used to demonstrate our idea, and we normally utilize a batch of samples for training.

sufficient training, the action with the highest advantage becomes dominant in the policy distribution, achieving the policy selection goal. Please refer to Algorithm 2 in Supplementary V for the implementation details of behavior selection.

C. GAN-based policy divergence estimation

One key issue we remained in previous subsections is to use GAN to estimate the divergence between a behavior policy and a learning (target) policy based on samples generated by them. In GAN, there is a generator (generative network) used for mapping from a latent space to a data distribution of interest, and a discriminator (adversarial network) used for distinguishing samples produced by the generator from the real data distribution (Fig. 2). Sebastian et al. [25] pointed out that the loss function of GAN essentially serves as a divergence measure between the fake and real data distributions. The generator's objective is to minimize the divergence to closely emulate real data, while the discriminator aims to maximize this divergence to effectively identify fake samples. Therefore, we can regard the actor (policy network) π in RL as the generator in GAN, which aims to diminish the divergence between the action distribution generated by it and the action distribution contained in the mix dataset, achieving the behavior cloning purpose (Fig. 2). The discriminator is used for improving the performance of the generator (actor) via adversarial learning.

Let the neural network Y be the variational function in f-GAN, we can leverage Y estimate the KL divergence between two probability distributions P and Q based on samples x generated by them:

$$D_{KL}(P||Q) = \mathbb{E}_{x \sim P} Y(x) + \mathbb{E}_{x \sim Q} [\log(-Y(x))] + 1 \quad (16)$$

The optimization equation for Y is

$$\arg \max_{\psi} \mathbb{E}_{x \sim P} [Y_{\psi}(x)] - \mathbb{E}_{x \sim Q} \log[-Y_{\psi}(x)] - \frac{\lambda}{2} I^2(\psi) \quad (17)$$

where ψ is the weight matrix of Y , I is a non-negative function used for measuring the complexity of g , and $\lambda > 0$ is a regularization parameter. Please refer to Supplementary II for the derivation of using Y for divergence estimation, as well as the training objective of Y .

In RL scenarios, x is the observation-action pair (o, a) , and P and Q are the distributions of the behavior policy θ_b and the target policy θ_t , respectively. Therefore, the optimization objective of Y and the divergence estimator become

$$\arg \max_{\psi} \mathbb{E}_{o, a \sim \theta_b} [Y_{\psi}(o, a)] - \mathbb{E}_{o, a \sim \theta_t} \log[-Y_{\psi}(o, a)] - \frac{\lambda}{2} I^2(\psi) \quad (18)$$

$$D_{KL}(P||Q) = \mathbb{E}_{o, a \sim \theta_b} Y(o, a) + \mathbb{E}_{o, a \sim \theta_t} [\log(-Y(o, a))] + 1 \quad (19)$$

The pseudocode for the proposed method is provided in Algorithm 3 of Supplementary V. In our approach, the objective of the variational network, akin to the discriminator in the vanilla GAN, is twofold: to accurately estimate \hat{y} and to furnish a divergence measurement between two probability distributions based on samples generated by them. Subsequently, the policy network (generator) undergoes optimization to minimize this

divergence. This leads to a reduction in the disparity between the output probability distribution $\pi_{\theta^i}^i(\cdot|o^i)$ and the ground-truth $\pi_{\theta^i}^i(\cdot|o^i)$ contained in the mixed dataset. In the case of f-GAN, the learning objective of the generator shares similarities with the objective utilized in this paper. However, in f-GAN, the variational network also seeks to maximize the divergence between two distributions. Our approach, on the other hand, centers on precisely estimating the divergence rather than altering it. We posit that once the samples and divergence function are determined, the divergence between the two distributions remains constant. This crucial distinction sets our method apart from f-GAN. Please refer to Supplementary III for the proof of the convergence of MAOBI.

V. RESULTS

A. Simulation settings

The offline dataset \mathcal{D} used in this paper is collected during the operation of well-planned controllers (MPPT, Wake Steering) or fine-tuned controllers (MAPPO, HAPPO, and MAIPO) deployed in wind farms. Each sample in the mixed dataset \mathcal{D} is in the form of $\langle o_t^i, a_t^i, r_t^i, o_{t+1}^{i+1} \rangle$, where $i = 1, 2, \dots, n$ is the turbine ID, and t is the time step during operation. The observation o^i is a 24-dimensional vector constituted by the wind velocity on the rotation plane of turbine i . The action vector, denoted as a^i , is comprised of the yaw angle and CT prime (CT: thrust coefficient, determined by the pitch angle and the rotor speed). The reward function r^i is proportional to the power output of the turbine generator.

$$r_{g1} = \begin{cases} 0.15 * \left(\frac{P}{P_r}\right)^2 & \text{if } P \leq P_r \\ 0 & \text{if } P > P_r \end{cases} \quad (20)$$

where $P_r = 5e6W$ denotes the rated power of the turbine.

The testing environments include three wind farms with a grid layout (Fig. 6), a random layout (Fig. 7) and a real-world layout (C-Power wind farm, please refer to Fig. 8 and Supplementary III), respectively, all equipped with NREL 5MW wind turbines. Within the farm domain, the top surface is characterized as an inviscid (slip) wall, while the bottom surface functions as a no-slip wall. The front and back surfaces are treated with far-field-type (Riemann-invariant characteristic) boundary conditions, imposing the specified values for the velocity vector and pressure in a weak manner. The left surface serves as the inflow layer, allowing for the adjustment of inflow wind and the introduction of turbulence, and the right surface functions as the outflow layer.

Regarding the NREL 5MW turbine, its gross properties are shown in Supplementary IV. The longitudinal separation between two adjacent turbines is set at $3d$, while the lateral spacing is $5d$. All NREL 5MW wind turbines have a same gearbox ratio of 97. The rated generator torque is $15,000 N \cdot M$, with a maximum reaching $47,402.91 N \cdot M$. The rated blade rotation speed and generator speed are specified as $1.254 rad/s$ and $121.6805 rad/s$, respectively. The control intervals for sub-modules of these turbines are set as $[0, 1.57] rad$ for blade pitches, $[-1.57, 1.57] rad$ for nacelle yaw, and $[0, 47,402] N \cdot M$ for generator torque. To obtain the state

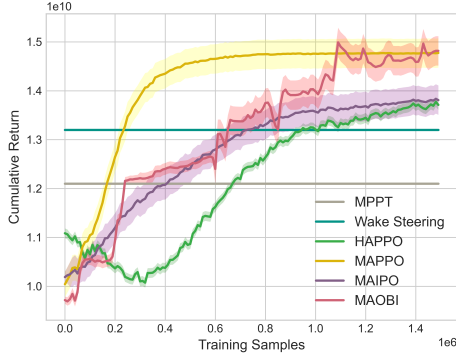


Fig. 3. Comparison of training curves calculated by online RL methods, MPPT, Wake steering, and MAOBI.

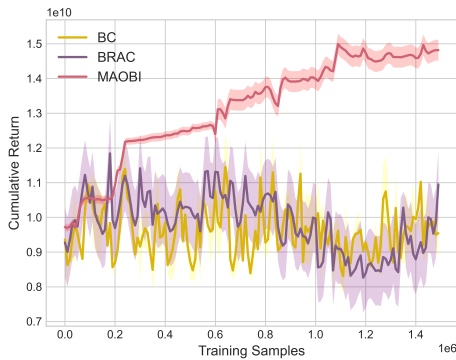


Fig. 4. Comparison of offline RL methods, BC, BRAC, and MAOBI.

vector, we should first normalize the above variable using their maximum values to make them dimensionless, and then concatenate them. For the implementation of our algorithms and detailed parameters for training and testing, please refer to our GitHub repository.

B. Training results of MAOBI

Fig. 3 illustrates the training curve of MAOBI in comparison with online RL algorithms and MPPT, while Fig. 4 showcases the comparison with other offline RL methods. The central solid lines represent the fitted test results of the control policy at each training step, offering insights into the improvement ratio of the corresponding method during training. The shadow area surrounding the curve depicts the standard deviation of five trials, providing a glimpse into the training stability under random initial conditions.

Upon analyzing Fig. 3, it becomes evident that the performance of the proposed MAOBI method surpasses not only the greedy MPPT but also a range of online RL methods. Furthermore, its endpoint closely approaches that of the state-of-the-art MAPPO under conditions without online feedback. Notably, the improvement ratio of MAOBI is relatively low in the initial training stage, as the primary focus lies on learning a robust discriminator during this phase, where the divergence measurement may be imprecise. In the subsequent

stage, the offline methods including Behavior Cloning (BC) and Behavior Regularized Actor Critic (BRAC [17]) struggle to improve the control policy from the offline dataset, highlighting the inherent high complexity of wind farm control tasks. In contrast, MAOBI continues to enhance the policy, attributed to the configuration of its policy selection process.

In above tasks, online RL agents require a high-fidelity wind farm simulator that discretizes the wind field into fine-grained cells, employing CFD methods to accurately compute wind velocities for each cell. Fig. 1 shows that when using the RK4 method to solve the wind dynamics in a computational domain with $1e^{10}$ cells, the minimum memory usage is 2.384 TB. In our scenario, we use the standard wind farm ($3\text{km} \times 3\text{km} \times 1\text{km}$) with $3000 \times 3000 \times 1000 = 9e^9$ cells. Thus, the minimum memory requirement is approximately $2.384 \times 9e^9 / 1e^{10} = 2.147$ TB. Additionally, the model requires a block matrix to store distances between computational cells and turbine elements, consuming an additional 0.683 TB of memory. Consequently, the theoretical minimum memory usage for online interaction between RL agents and the high-fidelity wind farm model is $2.147 + 0.683 = 2.83$ TB. Given the limited memory of individual GPUs (e.g., 24GB memory), at least $2.83 \text{ TB} \times 1024 / 24 \text{ GB} \approx 121$ GPUs are required to meet the memory demands. Other operations, such as the interpolation of aerodynamic parameters further increases memory demands. To ensure stability and avoid memory overflow, it is prudent to allocate a computational node with 128 GPUs (each with 24 GB of memory). Under this configuration, the computation time for a single timestep (corresponding to the collection of one training sample) is $128 \times 0.03776 \text{ second} = 0.0013425$ GPU hours, and the time for collecting $1e^6$ samples is approximately $0.0013425 \times 1e6 = 1342.5$ GPU hours.

Offline RL training does not rely on a high-fidelity wind farm simulator. In offline RL, memory usage is primarily devoted to storing batches of training samples, along with the current weights and architecture of the deep networks employed, resulting in significantly lower memory requirements. During the training process, the peak memory usage recorded was approximately 64.512 GB (0.063 TB) with a batch size of 256. The total training time for offline RL with the H100 GPU configuration was approximately 16.4 GPU hours.

C. Testing results of MAOBI

Fundamentally, the fine-tuned wind farm control policy is a composite of neural networks (NN). The forward reasoning of NN is swift, enabling real-time responsiveness. Additionally, the localized input of the sub-control policy for each turbine eliminates the need for communication during operation. Fig. 5 showcases the test results of the proposed MAOBI control compared to other cutting-edge methods. The testing outcomes align consistently with the endpoints of the training curves, suggesting that the offline MAOBI control policy attains comparable performance to the state-of-the-art MAPPO control policy. Notably, the power output of the wind farm controlled by MAOBI remains stable under fluctuating wind speeds, indicating high power quality.

Training curves and deployment results unequivocally demonstrate that the trained offline control policy significantly

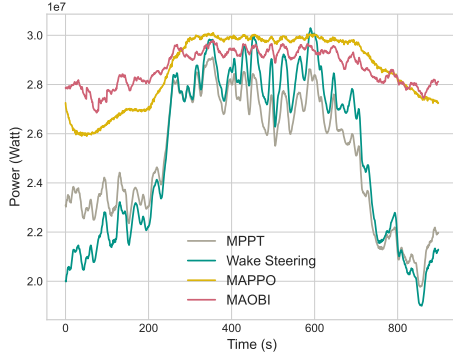


Fig. 5. Power outputs of the SOWFA wind farm controlled by MPPT, Wake Steering, MAPPO, and MAOBI policies.

TABLE II
CONTROL RESULTS OF MPPT AND MAOBI

Method	Index	Unit	T1	T2	T3	Total
MPPT	Power generation	MW	3.148	0.937	0.866	4.951
	Generator speed	rad/s	110.5	82.1	81.7	/
	Rotor torque	MN·M	2.75	0.91	1.43	/
	Yaw angle	rad	0	0	0	/
MAOBI	Power generation	MW	2.348	1.932	2.656	6.936
	Generator speed	rad/s	100.1	93.81	107.6	/
	Rotor torque	MN·M	2.19	1.97	2.38	/
	Yaw angle	rad	0.523	0.524	0.099	/

enhances the power generation of wind farms. However, the control strategy learned by MAOBI from offline datasets remains elusive. Hereupon, we leverage the control actions of turbines under specific wind conditions to showcase the learned control policy. Fig. 6 illustrate the wind speed on the horizontal turbine hub plane in the wind farm where the turbines are controlled by MPPT and MAOBI, respectively. In the MPPT diagram, upstream turbines rotate at their maximum allowable speed, avidly extracting wind energy from the field, leading to a pronounced decrease in wind speed on the rotation plane of downstream turbines due to the wake effect (Fig. 6). For instance, the hub wind speeds of downstream turbine 2 and turbine 3 are 3.21 and 3.34 m/s respectively, whereas this value for upstream turbine 1 is 7.8 m/s (Fig. 6 left lower), causing the output power of turbine 2 (0.937 MW) and turbine 3 (0.866 MW) to be much lower than turbine 1. Consequently, the overall power generation of the wind farm keeps at a relatively low level, limiting its capacity. In comparison, the MAOBI control policy adopts a collaborative approach to alleviate the negative effect of upstream wakes to facilitate power generation. Specifically, upstream wind turbines redirect their wakes via yaw control (see yaw angles in Table II). In this case, although the wind speed on downstream turbines does not fully recover to the inflow level, it surpasses that in the MPPT counterpart (Fig. 6 right lower), contributing to a farm-level increase in power. Simultaneously, the rotation speed of upstream turbines is reduced to degrade their wake intensity (see generator speeds in Table II). This joint control action can further enhance the wind energy available from downstream turbines (see power generations in Table II).

TABLE III
PERFORMANCE COMPARISON OF WIND FARM CONTROL METHODS

Method	Training time (GPU Hour)	Execution time (Second)	Memory usage (Byte)	Power generation (MW)	Power fluctuation (MW)
MPPT	N/A	real time	MB-level	106.69	20.75
PWS	N/A	real time	MB-level	114.16	22.32
MPYO	N/A	1.27	GB-level	117.64	15.87
FVWC	N/A	2.63	TB-level	119.34	12.11
MAPPO	3475.6	real time	TB-level	121.87	14.62
MAOBI	23.2	real time	GB-level	121.63	9.98

Fig. 7 and Fig. 8, showcasing the turbine yaw actions in wind farms with a random layout and a real layout obtained from the C-Power wind farm, further strengthen the conclusions drawn from the grid-layout wind farm. Specifically, the learned control policy primarily redirects the wakes of upstream wind turbines into the array gaps to decrease their effect on downstream turbines. If there are no downstream turbines, the control strategy for upstream turbines is to minimize yaw misalignment. For turbines located on the edges of the wind field, the farm controller redirects their wakes outside of the turbine arrays, increasing the wind capture of the entire wind farm. Fig. 9 shows the detailed growth in power generation of three methods with the MPPT baseline, where PWS (Preview wake steering [4]) is the Gaussian wake model-based wind farm control method and FVWC (Free-Vortex Wake Controller [41]) is the CFD-based controller is the control method modified by field experiments. Results demonstrate that all three wind farm controllers can significantly increase the power generation of wind farms by mitigating the wake effect. The proposed MAOBI method achieves the best performance, with an average increase of 16.16%, exceeding that of PWS (6.95%) and FVWC (12.16%).

D. Comparison analysis

This subsection compares the proposed MAOBI with cutting-edge wind farm control methods in the C-Power wind farm under five evaluation indicators listed in Table III, where MPYO (Maximize Power through Yaw Optimization [2]) is the control method modified by field experiments. From Table III, we observe that the average power outputs of the C-Power wind farm, when controlled by learning-based methods (MAPPO and MAOBI), are the highest compared to other methods. The primary drawback of MAPPO is its high memory and computational requirements due to the need for online interaction with the CFD-based simulator. Although MPYO and FVWC also demonstrate excellent performance, their execution times exceed the control timestep (0.05s) of wind turbines because they require online optimization at each step. While MPPT and PWS are capable of real-time control, their power generation capacity could be further improved. Therefore, considering training costs, hardware constraints, and control performance, the proposed MAOBI method is the most suitable choice for developing a low-cost, real-time, high-performance wind farm controller.

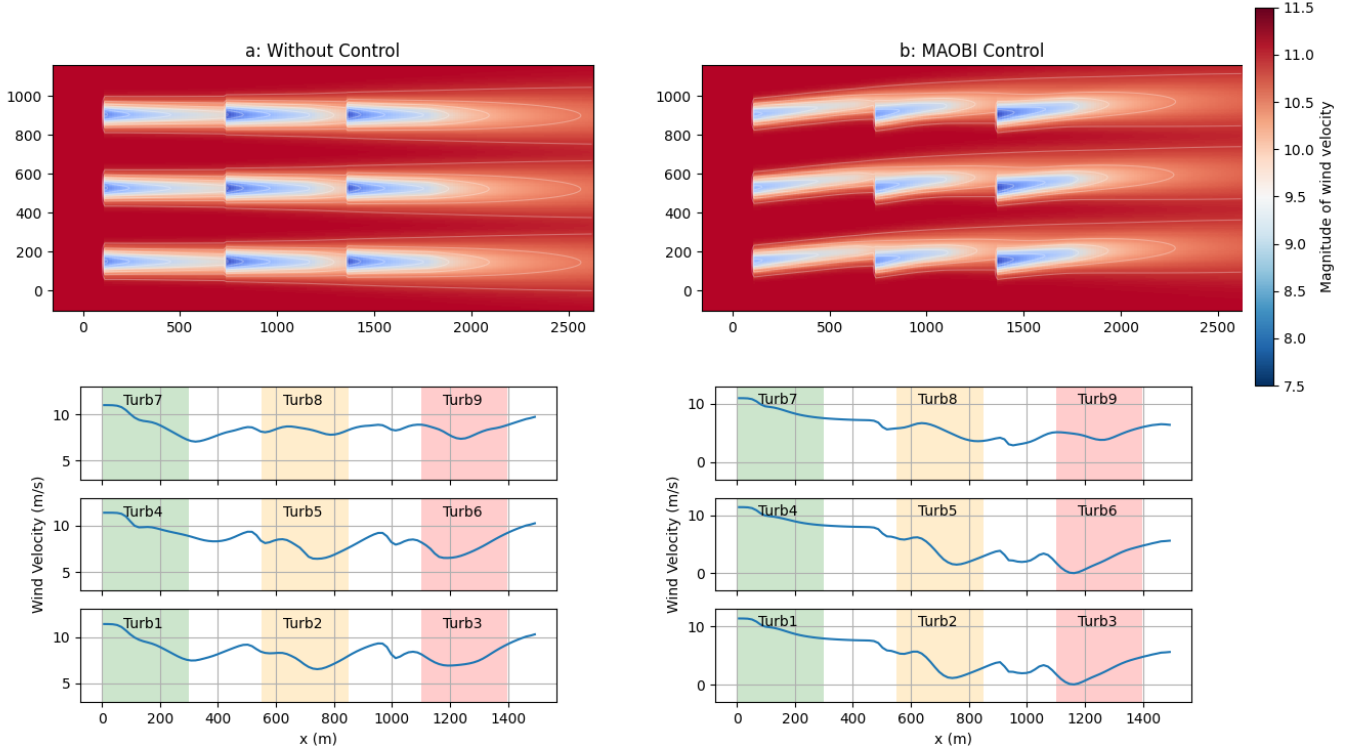


Fig. 6. Turbine wakes in a 9-turbine farm controlled by MPPT (a, without yaw control) and MAOBI (b), and the corresponding wind speed on three hub lines (Bottom). Wind direction: Left, $260^\circ \pm 10^\circ$. Inflow wind speed: $10 \pm 2 \text{ m/s}$.

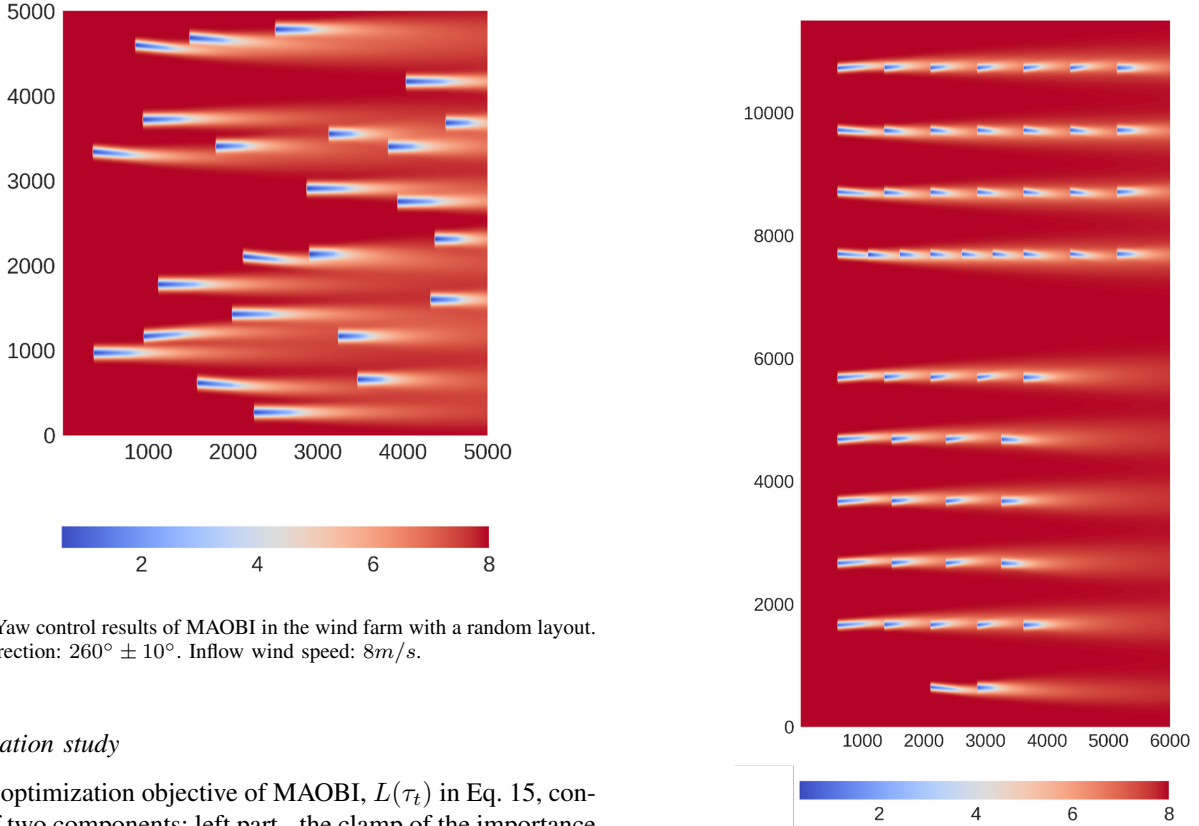


Fig. 7. Yaw control results of MAOBI in the wind farm with a random layout. Wind direction: $260^\circ \pm 10^\circ$. Inflow wind speed: 8 m/s .

E. Ablation study

The optimization objective of MAOBI, $L(\tau_t)$ in Eq. 15, consists of two components: left part - the clamp of the importance sampling ratio $g(\kappa, G^i(s_t, a_t))$, ensuring the monotonically improving policy during training; right part - the distance D_f between the target policy and the behavior policy estimated via

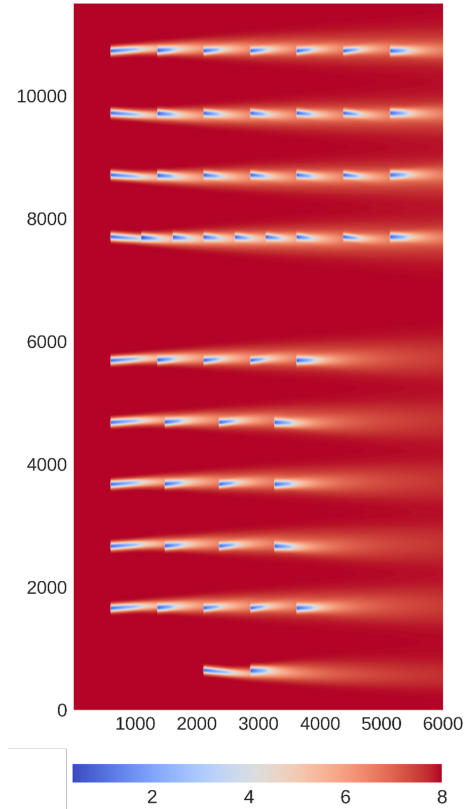


Fig. 8. Yaw control results of MAOBI in the C-Power wind farm. Wind direction: $260^\circ \pm 10^\circ$. Inflow wind speed: 8 m/s .

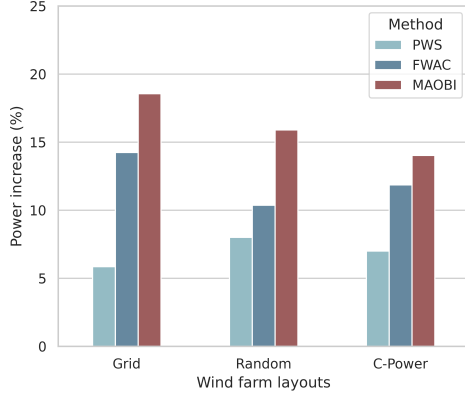


Fig. 9. Power increase of three controllers compared with the MPPT baseline.

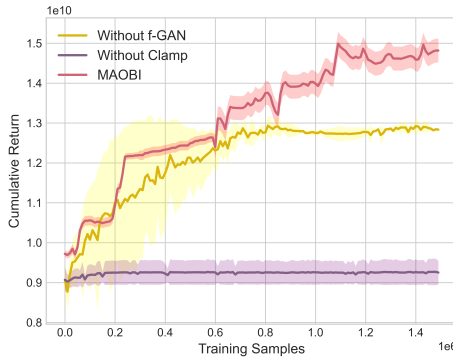


Fig. 10. Results of ablation studies.

f-GAN, achieving the policy imitation function. In our ablation test, we remove the first clamp part (without clamp) and the second f-GAN distance part (without f-GAN), respectively. Ablation results (Fig. 10) show that the policy performance cannot be improved at all without the clamp function, as it ensures the convergence of the policy in the training process. Without the distance constraint calculated by f-GAN, the learning speed in training is slower, and the final optimized control policy is sub-optimal, which supports the necessity of introducing f-GAN in offline training for policy improvement.

VI. CONCLUSION AND DISCUSSION

To avoid expensive online interactions with the high-fidelity wind farm simulator, this paper proposes an offline RL method named MAOBI to learn a collaborative farm-level control policy using previously collected trajectories generated by various controllers. MAOBI first mimics the behavior policy by minimizing the divergence between the real policy distribution contained in the offline dataset and the distribution estimated by f-GAN. Then, by selecting the high-return state-action pair contained in the dataset, MAOBI can further improve the cloned control policy. Simulation results show that the trained control policy can effectively mitigate upstream wakes through the joint control of nacelle yaw (redirecting the wake direction), blade pitch, and generator (slowing the rotation

speed to lower the wake intensity), achieving the purpose of increasing farm power generation. Meanwhile, based on the CTDE principle, the trained policy is deployed in a distributed manner, meaning that it can sidestep real-time communication between turbines in operation. Additionally, it is feasible to customize multiple control objectives by modifying the reward function in MAOBI.

Furthermore, in this paper, the proposed model-free, purely data-driven controller aims to achieve excellent performance across multiple dimensions. In terms of power generation alone, it may be weaker than state-of-the-art model-based controllers, which can access precise model information and interact with the model, allowing them to improve performance through immediate feedback on their control actions. If the performance our controller is lower than that of others, we can always fine-tune it to enhance its performance. Specifically, we can collect high-quality operational trajectories generated by the advanced controller and leverage them to further train our controller, enabling it to learn new control strategies. Therefore, our controller has substantial potential for growth, as it can continuously enhance its performance through fine-tuning. Moreover, this fine-tuning method also can be applied to improve its transferability.

REFERENCES

- [1] V. Lebbon, M. Thomson, S. Kerr, A. Markwell, and K. Baxter, "Global wind energy to top 1 tw threshold by the end of 2023," Wood Mackenzie, Tech. Rep., 2023.
- [2] M. F. Howland, J. B. Quesada, J. J. P. Martínez, F. P. Larrañaga, N. Yadav, J. S. Chawla, V. Sivaram, and J. O. Dabiri, "Collective wind farm operation based on a predictive model increases utility-scale energy production," *Nature Energy*, vol. 7, no. 9, pp. 818–827, 2022.
- [3] Y. Huang, S. Lin, and X. Zhao, "Multi-agent reinforcement learning control of a hydrostatic wind turbine-based farm," *IEEE Transactions on Sustainable Energy*, 2023.
- [4] E. Simley, P. Fleming, J. King, and M. Sinner, "Wake steering wind farm control with preview wind direction information," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 1783–1789.
- [5] M. J. Churchfield, P. Fleming, B. Bulder, and S. M. White, "Wind turbine wake-redirecting control at the fishermen's atlantic city windfarm," in *Offshore Technology Conference*. OTC, 2015, pp. OTC-25644.
- [6] O. E. Egbomwan, H. Chaoui, and S. Liu, "A physics-constrained td3 algorithm for simultaneous virtual inertia and damping control of grid-connected variable speed dfwg wind turbines," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [7] H. Dong, J. Zhang, and X. Zhao, "Intelligent wind farm control via deep reinforcement learning and high-fidelity simulations," *Applied Energy*, vol. 292, p. 116928, 2021.
- [8] Y. Huang, X. Wang *et al.*, "Soft policy optimization using dual-track advantage estimator," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1064–1069.
- [9] S. Cayci, S. Satpathi, N. He, and R. Srikant, "Sample complexity and overparameterization bounds for temporal difference learning with neural network approximation," *IEEE Transactions on Automatic Control*, 2023.
- [10] Y. Zhang, L. Qiao, and M. Zhao, "Fault diagnosis for wind turbine generators using normal behavior model based on multi-task learning," *IEEE Transactions on Automation Science and Engineering*, 2023.
- [11] J. Schulman, F. Wolski *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
- [13] L. E. Andersson, O. Anaya-Lara, J. O. Tande, K. O. Merz, and L. Imsland, "Wind farm control-part i: A review on control system concepts and structures," *IET renewable power generation*, vol. 15, no. 10, pp. 2085–2108, 2021.

- [14] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, 2005.
- [15] M. Riedmiller, "Neural fitted q iteration—first experiences with a data efficient neural reinforcement learning method," in *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005*. Springer, 2005, pp. 317–328.
- [16] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International conference on machine learning*. PMLR, 2019, pp. 2052–2062.
- [17] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," *arXiv preprint arXiv:1911.11361*, 2019.
- [18] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.
- [19] D. Precup, "Eligibility traces for off-policy policy evaluation," *Computer Science Department Faculty Publication Series*, p. 80, 2000.
- [20] E. Todorov, "Linearly-solvable markov decision problems," *Advances in neural information processing systems*, vol. 19, 2006.
- [21] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *Advances in neural information processing systems*, vol. 29, 2016.
- [22] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "Morel: Model-based offline reinforcement learning," *Advances in neural information processing systems*, vol. 33, pp. 21 810–21 823, 2020.
- [23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.
- [24] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems*, vol. 29, 2016.
- [25] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," *Advances in neural information processing systems*, vol. 29, 2016.
- [26] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.
- [27] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.
- [28] G. Moor and H. Beukes, "Maximum power point trackers for wind turbines," in *2004 IEEE 35th Annual Power Electronics Specialists Conference*, vol. 3. IEEE, 2004, pp. 2044–2049.
- [29] L. Gao and J. Hong, "Data-driven yaw misalignment correction for utility-scale wind turbines," *Journal of Renewable and Sustainable Energy*, vol. 13, no. 6, 2021.
- [30] S. J. Johnson, J. P. Baker, C. Van Dam, and D. Berg, "An overview of active load control techniques for wind turbines with an emphasis on microtabs," *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 13, no. 2-3, pp. 239–253, 2010.
- [31] S. Lu, Z. Gao, P. Zhang, Q. Xu, T. Xie, and A. Zhang, "Event-triggered federated learning for fault diagnosis of offshore wind turbines with decentralized data," *IEEE Transactions on Automation Science and Engineering*, 2023.
- [32] P. Fleming, J. Annoni, J. J. Shah, L. Wang, S. Ananthan, Z. Zhang, K. Hutchings, P. Wang, W. Chen, and L. Chen, "Field test of wake steering at an offshore wind farm," *Wind Energy Science*, vol. 2, no. 1, pp. 229–239, 2017.
- [33] M. Becker, D. Allaerts, and J. Van Wingerden, "Floridyn—a dynamic and flexible framework for real-time wind farm control," in *Journal of Physics: Conference Series*, vol. 2265, no. 3. IOP Publishing, 2022, p. 032103.
- [34] J. R. Marden, S. D. Ruben, and L. Y. Pao, "A model-free approach to wind farm control using game theoretic methods," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1207–1214, 2013.
- [35] P. M. Gebraad, F. W. Teeuwisse, J. Van Wingerden, P. A. Fleming, S. D. Ruben, J. R. Marden, and L. Y. Pao, "Wind plant power optimization through yaw control using a parametric model for wake effects—a cfd simulation study," *Wind Energy*, vol. 19, no. 1, pp. 95–114, 2016.
- [36] P. M. Gebraad and J.-W. van Wingerden, "Maximum power-point tracking control for wind farms," *Wind Energy*, vol. 18, no. 3, pp. 429–447, 2015.
- [37] P. Fleming, J. King, K. Dykes, E. Simley, J. Roadman, A. Scholbrock, P. Murphy, J. K. Lundquist, P. Moriarty, K. Fleming et al., "Initial results from a field campaign of wake steering applied at a commercial wind farm—part 1," *Wind Energy Science*, vol. 4, no. 2, pp. 273–285, 2019.
- [38] M. Churchfield, S. Lee, and P. Moriarty, "Overview of the simulator for wind farm application (sowfa)," *National Renewable Energy Laboratory*, 2012.
- [39] S. Boersma, B. Doekemeijer, M. Vali, J. Meyers, and J.-W. van Wingerden, "A control-oriented dynamic wind farm model: Wfsim," *Wind Energy Science*, vol. 3, no. 1, pp. 75–95, 2018.
- [40] W. Munters and J. Meyers, "Towards practical dynamic induction control of wind farms: analysis of optimally controlled wind-farm boundary layers and sinusoidal induction control of first-row turbines," *Wind Energy Science*, vol. 3, no. 1, pp. 409–425, 2018.
- [41] M. J. van den Broek, M. Becker, B. Sanderse, and J.-W. van Wingerden, "Dynamic wind farm flow control using free-vortex wake models," *Wind Energy Science Discussions*, vol. 2023, pp. 1–28, 2023.
- [42] J. Park and K. H. Law, "A data-driven, cooperative wind farm control to maximize the total power production," *Applied Energy*, vol. 165, pp. 151–165, 2016.
- [43] J. Park, S.-D. Kwon, and K. Law, "A data-driven, cooperative approach for wind farm control: a wind tunnel experimentation," *Energies*, vol. 10, no. 7, p. 852, 2017.
- [44] H. Zhao, J. Zhao et al., "Cooperative wind farm control with deep reinforcement learning and knowledge-assisted learning," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 6912–6921, 2020.
- [45] V. R. Padullaparthi, S. Nagarathinam, A. Vasan, V. Menon, and D. Sudarsanam, "Falcon-farm level control for wind turbines using multi-agent deep reinforcement learning," *Renewable Energy*, vol. 181, pp. 445–456, 2022.
- [46] H. Dong and X. Zhao, "Reinforcement learning-based wind farm control: Toward large farm applications via automatic grouping and transfer learning," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 12, pp. 11 833–11 845, 2023.
- [47] Y. Huang and X. Zhao, "Reinforcement learning-based multiobjective control of grid-connected wind farms," *IEEE Transactions on Industrial Informatics*, 2024.
- [48] A. Roy, V. Borkar, A. Karandikar, and P. Chaporkar, "Online reinforcement learning of optimal threshold policies for markov decision processes," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3722–3729, 2021.
- [49] Y. Zhang and M. M. Zavlanos, "Cooperative multi-agent reinforcement learning with partial observations," *IEEE Transactions on Automatic Control*, 2023.



tions in offshore renewable energy systems, smart grids, and autonomous systems.



Yubo Huang received his B.S. degree in automation from Northwestern Polytechnical University, Xi'an, China in 2018, his M.S. degree in control engineering from Shanghai Jiao Tong University, Shanghai, China in 2021, and his Ph.D. degree in engineering from the University of Warwick, Coventry, UK in 2024, respectively. He was a Marie-Curie Early Stage Researcher and then Research Assistant at the University of Warwick. His main research interests include reinforcement learning, control theory and computational fluid dynamics (CFD), with applica-

Xiaowei Zhao received the Ph.D. degree in control theory from Imperial College London, London, U.K., in 2010. He was a Postdoctoral Researcher with the University of Oxford, Oxford, U.K., for three years before joining the University of Warwick, Coventry, U.K., in 2013. He is Professor of control engineering with the School of Engineering, University of Warwick. His main research interests include control theory and machine learning with applications in offshore renewable energy systems and smart grids.