Graph Convolution Reinforcement Learning for Active Wake Control in Windfarms

Application of a Multi-Agent Reinforcement Learning Algorithm

Jefferson Yeh[1]

Supervisor(s): Mathijs de Weerdt[1]**, Greg Neustroev**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Jefferson Yeh
Final project course: CSE3000 Research Project
Thesis committee: Mathijs de Weerdt, Greg Neustroev, Przemysław Pawełczak

Abstract

Wind energy, generated by windfarms, is playing an increasingly critical role in meeting current and future energy demands. windfarms, however, face a challenge due to the inherent flaw of wake-induced power losses when turbines are located in close proximity. Wakes, characterized by regions of turbulence and lower wind speed, are created as air passes through the rotors, reducing the efficiency of downstream turbines. Wake losses can be reduced by yawing upstream turbines to steer the wake away from downstream turbines. While there are power losses associated with turning turbines off-wind, the gains in the subsequent turbines can outweigh these losses. Yawing turbines to increase overall power output is known as Active Wake Control, and the literature shows that single-agent reinforcement learning algorithms can be used to learn such control policies. However, these approaches are limited to a small number of turbines and do not scale well to larger windfarms. Multi agent reinforcement learning algorithms do scale to larger windfarms and this paper investigates the application of the DGN algorithm for windfarm active wake control. DGN is a fully cooperative algorithm that utilizes a graph representation of agents to encourage collaboration among neighboring turbines. The use of DGN is particularly interesting because windfarms naturally have a topological structure, and depending on the modeling choices, graphs can capture a significant amount of information. This paper demonstrates that DGN can learn useful policies for wake control. Although it does not outperform the DQN single-agent algorithm on small windfarms, its advantage becomes apparent in larger windfarms, where its performance remains consistent while DQN's performance deteriorates.

1 Introduction

Nowadays, energy production is rapidly shifting away from fossil fuels towards renewable alternatives. The urgent need to reduce greenhouse gas emissions is driving technological advancements in renewable energy production to meet the projected global contribution target of 77% by 2050, as set by the Intergovernmental Panel on Climate Change (IPCC) [1]. Wind energy emerges as a serious solution in both the short-term and especially in the long-term, where it is projected to contribute 20% of total energy demand [2]. This can be achieved through the use of individual turbines or windfarms, consisting of a collection of turbines. Due to the infrastructural requirements of turbines, windfarms are more suitable for large-scale energy production. Consequently, it is relevant and important to study techniques that improve windfarm energy production.

1.1 Wake Problem

A significant issue with windfarms is the interaction between turbines in close proximity to each other. When wind passes through a turbine, it creates a wake behind it, which is a region of high turbulence and low wind speed. These conditions have a negative impact on the efficiency of subsequent turbines, diminishing the windfarm's overall output by up to 13% [3]. Turbines experience minimal wake effects when they are sufficiently spaced apart. This, however, induces additional costs and logistical challenges, such as acquiring more land and installing transmission lines. Consequently, turbines are usually placed within wake vicinities of each other [3]. Considering the importance of windfarms to global energy production, wake induced energy loss is a pressing and relevant problem.

Currently employed solutions can be categorized as passive or active. Passive solutions utilize historical wind data to customize turbine setups for specific locations [3]. The latter, and more efficient, approach is known as Active Wake Control (AWC). By controlling the yaw angle of a turbine, as demonstrated in Figure 1, it is possible to adjust the direction of its wake, thereby reducing wake losses on subsequent turbines. This method improves the overall output of the windfarm by sacrificing the production of some turbines to benefit those downstream [4].

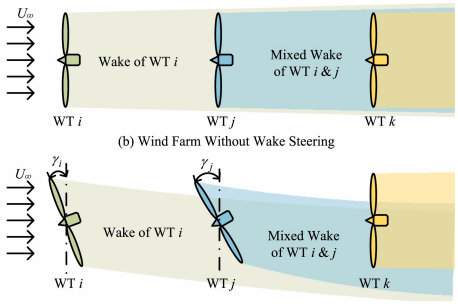


Figure 1: Turbine wake interaction with and without AWC. Figure obtained from Dong [5].

1.2 Existing Solutions

Numerous control techniques have been employed for AWC including classical control and reinforcement learning (RL) methods [6]. Model-based algorithms rely on an internal representation of the system to make control decisions based on its modelled dynamics and behaviour. Various classical control algorithms, such as model predictive control [7], employ feedback loop techniques that combine the observed system state, system model, and other control rules into a decision. Although these techniques have demonstrated performance improvements of up to 15% [6], the model-based approach also has its limitations. It depends on accurate system models to produce control policies that be equally effective on the actual system. Discrepancies often arise between simulation and deployment performances, due to the inherent challenge of accurately capturing the highly complex intricacies of turbine aerodynamics [6].

Model-free RL is a category of RL algorithms where a control policy is learned based on data from interactions with the environment. This is advantageous over the previously discussed model-based methods because it removes

the reliance on complex dynamics modelling and its associated limitations [5]. Existing research on applying RL to the AWC problem predominantly focuses on Single Agent RL (SARL) algorithms [6]. The SARL approach treats the entire windfarm as a single entity, considering all turbines together for learning and making control decisions. Applying SARL to AWC, which inherently involves multiple agents, has demonstrated performance improvements in small scale windfarms containing a limited number of turbines [8; 9]. The SARL paradigm, however, does not scale to larger windfarms with a higher number of turbines due to the combinatorial explosion of possible turbine actions and states.

1.3 Research Focus

This paper proposes model-free Multi Agent RL (MARL) algorithms as a promising alternative for AWC. Unlike SARL, each agent in MARL learns and makes control decisions independently of others [10]. This means that adding agents leads to a linear increases in the state and action spaces, rather than an exponential increase. Existing MARL algorithms can be categorized into different classes, each being suitable for different applications. The specific class of fully cooperative MARL algorithms with a fully observable critic appears to be suitable for AWC in windfarms. Agents in this class of algorithms work towards a common goal while sharing the same global reward [11]. This aligns well with the nature windfarms, where all turbines collaborate to collectively maximize the overall power output.

Graph Convolutional RL (DGN) is the algorithm of interest. It models agents as a graph and utilizes this representation to promote collaboration among nearest neighbors [12]. Moreover, the structure of windfarms naturally lends itself to graph modelling. When considering characteristics such as wake area and wind direction, representations have the potential for a variety of modeling approaches.

This paper focuses on the central question: *How can DGN be efficiently applied to the Active Wake Control of windfarms?* This question inspires the following sub-questions:

1. How does the learning rate influence the performance of DGN when applied to the AWC of windfarms?
2. How does the performance of DGN scale with increasing windfarm sizes?
3. How can the modelling of directionality in windfarm topological features be exploited by DGN to improve the windfarm performance?

The aim of the sub-questions, in chronological order, is to first establish a proof of concept for DGN before exploring potential improvements through modelling techniques.

This paper first formulates the AWC problem for MARL and the DGN algorithm. Second, it thoroughly motivates and describes the design and setup of the experiments. Finally, the results are presented, discussed and concluded.

2 DGN for Active Wake Control

2.1 Active Wake Control as a Multi Agent Problem

Multi agent RL for AWC problem can be modelled as a fully cooperative Markov Game, formally defined as $\left(\mathcal{N}, \{\mathcal{S}^i\}_{i\in\mathcal{N}}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \mathcal{R}\right)$ [10]. Here, $\mathcal{N}$ represents the set of agents, with each agent representing a turbine in the windfarm. $\mathcal{S}^i$ and $\mathcal{A}^i$ denote the state space observation and action space for each agent $i \in \mathcal{N}$. For the experiments conducted in this paper, the state space contains the yaw angle of the turbine, wind speed and wind direction. The action space for each turbine consists of reachable yaw angles, where the maximum clock and counterclockwise movement is defined by an angular rate of $1\,\mathrm{deg}$ per time step of 1 second. These angles are mapped to a range of [-1, 1] [8]. $\mathcal{P}$ is the transition probability from the current state to the next state, given the actions taken by all agents. Finally, the reward, denoted by $\mathcal{R}$, represents the windfarm power output at the specific time step. In cooperative MARL paradigm, each agent learns its own actions individually, but all agents share the same reward as they work towards a common goal [10].

The system wiwth which the agents interact is a windfarm simulator [8; 13]. It takes actions as inputs and determines the next state and reward. The turbines are simulated in discrete time steps, transitioning from one steady state to the next according to the inputted turbine actions. First, the yaws of the turbines are adjusted, then the wake aerodynamics and turbine power outputs are computed using the FLORIS [14] framework. Additionally, the windfarm simulator can simulate wind behaviour from stochastic processes [8]. For the scope of the paper, the wind profile will be static with speed 20 [m/s] and direction from West to East.

2.2 DGN Algorithm

The Graph Convolutional RL, also known as DGN, is a Q-learning based MARL algorithm. As its name suggests, DGN uses a graph representation to model agents in the environment. The algorithm makes use of edge traversals to incentivize each agent to collaborate with its nearest neighbours. The architecture of DGN, depicted in Figure 2, comprises of an Encoder, Convolutional layers, and a Q network for each agent. The Encoder layer is a Multi-Layer Perceptrons (MLP) that encodes the state of each agent into its respective feature vector [12]. The subsequent Convolutional layers extract latent features out of the encoded features. By utilizing an adjacency matrix graph representation, this layer ensures that agents only share information with their neighbouring agents. In the graph, agents are modelled by nodes, and edges exist only between agents that share information. The adjacency matrix enables the dynamic updating of agent graph representations to reflect non-static behaviour in real life scenarios [12]. Figure 2 illustrates two Convolutional layers, but more layers can be used. Each additional layer facilitates information flow through an additional edge traversal. Agents only have direct access to neighbouring agents but, additional Convolutional layers enables them to receive information from non-neighbouring agents through intermediate agents. Additionally, "higher order relation representation can be extracted, which effectively capture the interplay between agents and greatly help to make cooperative decision" [12]. Lastly, the latent features are fed into the Q-networks to learn action Q-values for each agent [12].

The implementation of the algorithm [15] uses discrete ac-

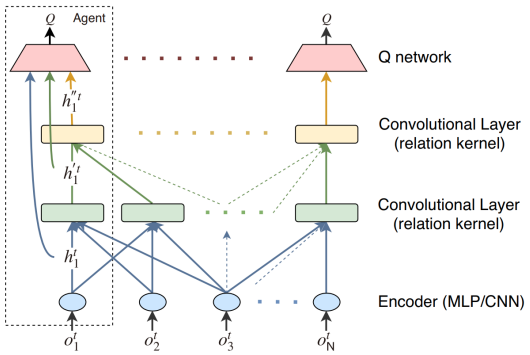


Figure 2: Diagram of the DGN architecture, obtained from Jiang [12].

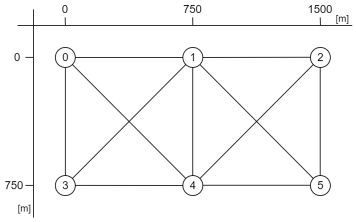
tion spaces. Since the turbine yaw angle is continuous, the action values are mapped to three discrete values: stationary, clockwise rotation at maximum speed, and counterclockwise rotation at maximum speed. During the training process, DGN explores the action space in two ways: randomly and by selecting the highest Q valued action. These methods are randomly selected using epsilon-greedy exploration. Initially, random actions are chosen with higher probability to encourage exploration. This probability gradually decreases as training progresses. DGN is trained episodically, where each episode contains a fixed number of steps or interactions with the environment. The learning takes place offline after the complete of environment interactions in each episode. Gradient descent is performed in batches over multiple epochs. At the start of the training process, learning is delayed by a predetermined number of episodes to ensure the replay buffer is sufficiently populated. Similar to other RL algorithms, DGN also includes several hyperparameters, some of which will be explored and tuned.

3 Windfarm Graph Representation

Inter-agent interaction in DGN is governed by the graph representation of the underlying agents [12]. In the used adjacency matrix representation, agent pairs that can directly communicate with each other are connected by an edge. Different graphs can therefore alter the information accessible to each agent within a single edge traversal, thereby influencing the overall performance of the algorithm. This section describes the different models that have been explored, referred to as undirected and directed, based on their respective representations of agent information exchanges.

3.1 Undirected Representation

Undirected, indicating bidirectional communication, is a naive representation achieved by assigning nodes represent turbines and adding an undirected unweighted edge between two turbines that share a wake interaction. Wake loss becomes negligible when turbines are sufficiently spaced, with a distance of at least 10 rotor diameter lengths apart [3]. Thus, an edge is included when the euclidean distance between two turbines does not satisfy the aforementioned condition. Intuitively, both turbines should exchange information since a turbine should be aware of the turbines it affects, and the downstream turbines should learn about the behaviour of upstream turbine(s). This relationship is captured by an undirected graph and its unweighted adjacency matrix. Figure 3 illustrates the graph representation of a 3×2 grid layout and its corresponding adjacency matrix.



(a) Nodes and undirected edges

	0	1	2	3	4	5
0	1	1	0	1	1	0
1	1	1	1	1	1	1
2	0	1	1	0	1	1
3	1	1	0	1	1	0
4	1	1	1	1	1	1
5	0	1	1	0	1	1

(b) Adjacency matrix

Figure 3: Undirected graph representation of a 3×2 grid windfarm layout.

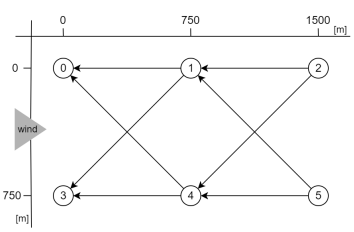
3.2 Directed Representation

DGN involves a trade-off between the cost of inter-agent communication capabilities and inter-agent cooperation [12]. However, this trade-off is not considered in the previous undirected representation, as it includes edges for turbine pairs that are oriented perpendicular to the wind and wake direction. Since the undirected representation still allows for a large number of edges, communication can be further reduced by incorporating directional information transfer based on the wind direction. This includes communication either in the upstream or downstream direction between turbines. Upstream directionality enables turbines to learn about the downstream turbines they affect, while downstream directionality allows downstream turbines to learn about the behaviour of their upstream counterparts. For both representations, turbine pairs that are neither downstream nor upstream of each other do not have an edge connection, even if they satisfy the distance condition. Figure 4 illustrates both directional modelling methods on a 3×2 windfarm layout.

4 Experiments

This section aims to investigate the research questions posed by the paper. It begins by presenting the setup up of hyperparameters and baselines employed in the experiments. Subsequently, each sub-research question is addressed by describing the experimental design, presenting the results, and conducting an analysis. These sub-research questions investigate the effects of the learning rate, the scalability of DGN, and the influence of different windfarm graph modelling techniques.

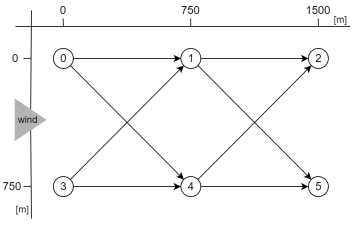
Each graph presented in this section plots the mean and 90th percentile and 10th percentile error bands of the experiment, each lasting 10,000 episodes and repeated three times. In every experiment, the training and evaluation rewards as well as training loss, were logged every 10 episodes. This data is aggregated as described and a centered simple moving average filter with window size five was ap-



(a) Upstream: Nodes and directed edges

	0	1	2	3	4	5
0	1	0	0	0	0	0
1	1	1	0	1	0	0
2	0	1	1	0	1	0
3	0	0	0	1	0	0
4	1	0	0	1	1	0
5	0	1	0	0	1	1

(b) Upstream: Adjacency matrix



(c) Downstream: Nodes and directed edges

	0	1	2	3	4	5
0	1	1	0	0	1	0
1	0	1	1	0	0	1
2	0	0	1	0	0	0
3	0	1	0	1	1	0
4	0	0	1	0	1	1
5	0	0	0	0	0	1

(d) Downstream: Adjacency matrix

Figure 4: Directed graph representation of a 3×2 grid windfarm layout .

plied. The filter calculates the mean of the values at episodes $(E-20, E-10, E, E+10, E+20)$, for $E \in [20, 9980]$.

4.1 Experimental Setup

The following subsections describe the setup used in the subsequent experiments. They outline the hyperparameter values used and present the baselines that will be used to evaluate the performance of DGN.

Hyperparameter Tuning

RL algorithm performance can be highly dependent on the values of hyperparameters used [16]. DGN encompasses a wide range of hyperparameters, with the most performance-sensitive ones being the learning rate (LR), batch size and epsilon. Please refer to Appendix C for the complete list hyperparameters. The LR significantly impacts performance as it controls the step size during gradient descent [17]. Additionally, the influence of batch size is closely related to the LR. Andrychowicz [17] argues that the LR and batch size should be tuned together, whereas Smith [18] claims that increasing batch size, instead of further decreasing LR, can yield improvements. Finally, the epsilon parameter in epsilon-greedy exploration influences the probability of exploring random actions, thereby expanding the exploration of the action space and diversifying the agent's learning experiences [19]. Moreover, Hariharan [19] highlights the impact of action space sampling schemes on learning performance, emphasizing the importance of providing the agent with a diverse collection of state and action combinations. Although this section summarized some potentially influential parameters, only the LR is tuned in the experiments conducted due to time constraints. All other hyperparameters will use the default values listed in Appendix C.

Baselines

The performance of DGN will be benchmarked against three different reference control methods: Static Yaw, FLORIS, and DQN. These references will provide insights into the suitability of applying DGN to the AWC problem. The values of these baselines are presented in Appendix B and will be used in the subsequent plots and analysis, where appropriate.

Static Yaw The static baseline is a non-active wake control methodology that aligns all the turbines to face the wind, where each turbine greedily maximizes its own power output. This baseline will be used to evaluate the policy learned by DGN in comparison to taking no yawing action. Policies that perform worse or comparably to this baseline may indicate an unsuitability for the AWC problem.

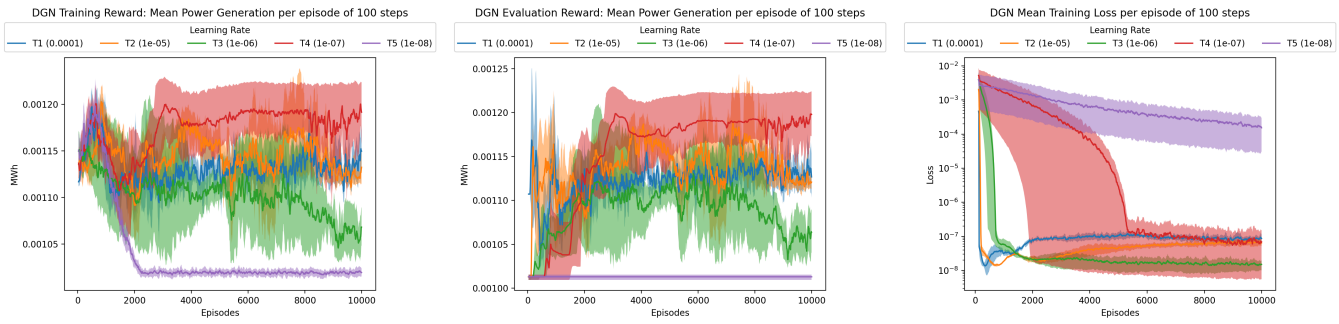
FLORIS This baseline uses FLORIS [14], a framework for wind turbine wake modelling and control. As mentioned in Section 2.1, FLORIS is employed by the windfarm environment to simulate airflow dynamics and power outputs. In this case, FLORIS is used to computationally determine an optimal yaw configuration per turbine using the Sequential Quadratic Programming method [20]. It is important to note that this method provides a good approximation but does not guarantee finding the global maximum. Therefore, achieving performances comparable to or exceeding the FLORIS baseline is an indication of a useful policy. FLORIS, however, is limited in its applicability to windfarms with a larger number of turbines, as it also suffers from the combinatorial increase in the search space, which impedes its convergence to a solution [20].

Deep Q-Network The final baseline is the SARL Deep Q-Network (DQN) algorithm [21; 22] trained on the windfarm. Since DGN utilizes Q-Networks, DQN serves as a comparable SARL algorithm that learns q-values in a similar manner. This comparison will be used to assess whether this particular MARL approach outperforms the SARL approach. The hyperparameter tuning and values used can be found in Appendix A.

4.2 Performance Impact of Learning Rate

The focus of this section is to investigate the influence of LR on the behaviour of DGN and, thereby, address the first sub-research question. This experiment explores a range of LR values, specifically 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, and 10^{-8}, which are chosen to provide a range that is a factor and divisor of 100 from the default LR value of 10^{-6}. The models corresponding to each LR value are denoted as *T1*, *T2*, *T3*, *T4*, and *T5*, respectively. The chosen range is hypothesized to showcase the impact of different LR values, where values that are too large can prevent convergence, while excessively small step sizes can converge to a local minimum too quickly. Large LR values may prevent the algorithm from learning any policy at all, and low LR values may learn a meaningless policy without much exploration. All other parameters in DGN and windfarm environment are kept constant. The models are trained on a basic 3×1 wind tunnel windfarm layout with an undirected graph representation.

The results of the experiments, including the training, evaluation, and loss curves, are presented in Figure 5. Analyzing



(a) Training reward. (b) Evaluation reward. (c) Training loss.

Figure 5: Training DGN on various learning rates.

the training and evaluation reward curves in Figure 5a and Figure 5b, it is immediately clear that only models *T4* and *T5*, which use LRs of 10^{-7} and 10^{-8} respectively, converge. The loss curve of *T5* in Figure 5c shows that it remains in its local minimum and gradually converges to the corresponding minimum due to its small LR. Interestingly, the converged reward value of *T5* is comparable to the Static Yaw baseline (0.001107, Appendix B), which coincides with the initial state of the windfarm environment. *T4*, on the other hand, learns a more useful policy, and this can be observed in its loss curve as well. Initially, it follows a shallow decrease, but once it discovers a better policy, it quickly converges and remains stable. Contrastingly, the LRs used by models *T1* and *T2* are too large, as evidenced by the considerable noise, spikes and non-convergence in the reward curves. More convincingly, the oscillations present in their loss curves are characteristic of the overshooting and undershooting caused by excessively large LR values. Although the reward curve of *T3* also does not converge within the specified episodes, the peaks and troughs in its loss curve are less pronounced, suggesting some inclination towards learning. Based on the results obtained with different LR values, it appears that the model can learn effectively within the range of $[10^{-6}, 10^{-8}]$, with a LR of 10^{-7} demonstrating the best performance. This LR value will be used in the subsequent experiments.

The conclusions drawn from the LR experiments align with the hypothesis, demonstrating the significant influences of LR on the learning behaviour of the algorithm and its ability to converge to a useful policy. When the LR is too large, the reward curve contains noisy fluctuations and fails to converge to a policy. This is consistent with the behaviour of large gradient descent steps, which tend to overshoot and undershoot the local minimum. As the LR decreases, the oscillations in the loss curve dampen out as the model adopts more appropriate step sizes. As expected, an appropriate LR enables the algorithm to learn a useful policy. However, further decreasing the LR prevents the exploration of different policies since the gradient descent step becomes insufficient to escape the immediate local minimum.

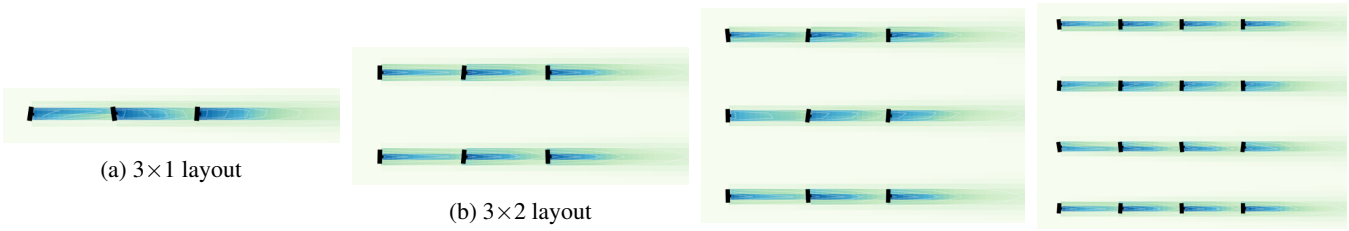
4.3 Performance Impact of Windfarm Size

The purpose of this section is to investigate the scalability of DGN in comparison to the SARL baseline, specifically in terms of increasing windfarm sizes. This aligns with the objectives of the second sub-research question. The experiment aims to explore windfarms of varying sizes, ranging from three turbines to six, nine, and up to a 16 turbine windfarm. These layouts are selected to gradually incorporate more turbines while maintaining wake interactions. The layout transitions from a simplified wind tunnel to a grid windfarm. The layouts are further explained below and visualized in Figure 6. The hypothesis posits that as windfarm size grows, the complexity of the optimal control rule also increases. Consequently, the SARL baseline is expected to significantly deteriorate, whereas DGN is anticipated to maintain its performance relative to the baselines.

> **Wind Tunnel:** The 3×1 wind tunnel setup features three turbines in a row, resulting in downstream turbines being in the wake of upstream turbines, as visualized in Figure 6a. This configuration allows for a small number of wind turbines while maximizing wake interactions.
>
> **Grids:** The windfarm layout is further expanded by adding additional rows, resulting in grid configurations of 3×2 (Figure 6b), 3×3 (Figure 6c) and 4×4 (Figure 6d) turbines. This gradual increase in the number of turbines introduces comparable wake effects, allowing for a gradual comparison between MARL and SARL scalability to larger windfarms.

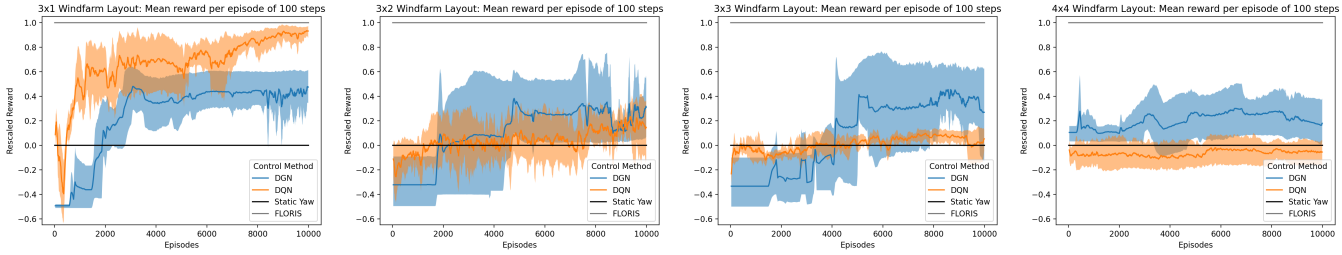
For these experiments, the layouts are modelled using undirected graphs. The windfarm environment settings and algorithm parameters for DGN and the other baseline methods are kept constant and outlined in Appendix A, B, and C. Since the windfarm power outputs cannot be directly compared across different layouts, the rewards are rescaled to enable direct comparison. In this rescaling, the Static baseline is assigned a value of zero, the FLORIS baseline is assigned a value of one, and all other reward curves are scaled accordingly. This rescaling facilitates the direct comparison and analysis of trends between different windfarm layouts.

The evaluation performance results for this experiment are presented in Figure 7, with the windfarm sizes increasing from left to right. The non-scaled results can also be found in Appendix D. Looking at the smallest windfarm in Figure 7a,



(a) 3×1 layout (b) 3×2 layout (c) 3×3 layout (d) 4×4 layout

Figure 6: Windfarm layout renders generated using the windfarm simulator [13].



(a) 3x1 windfarm layout. (b) 3x2 windfarm layout. (c) 3x3 windfarm layout. (d) 4x4 windfarm layout.

Figure 7: DGN and baseline performances on increasing windfarm sizes.

it is evident that DGN does not outperform FLORIS or DQN. In this case, DQN achieves higher performance more quickly and even approaches the FLORIS baseline. Regardless, this result still demonstrates that DGN learns a useful policy that outperforms the Static baseline.

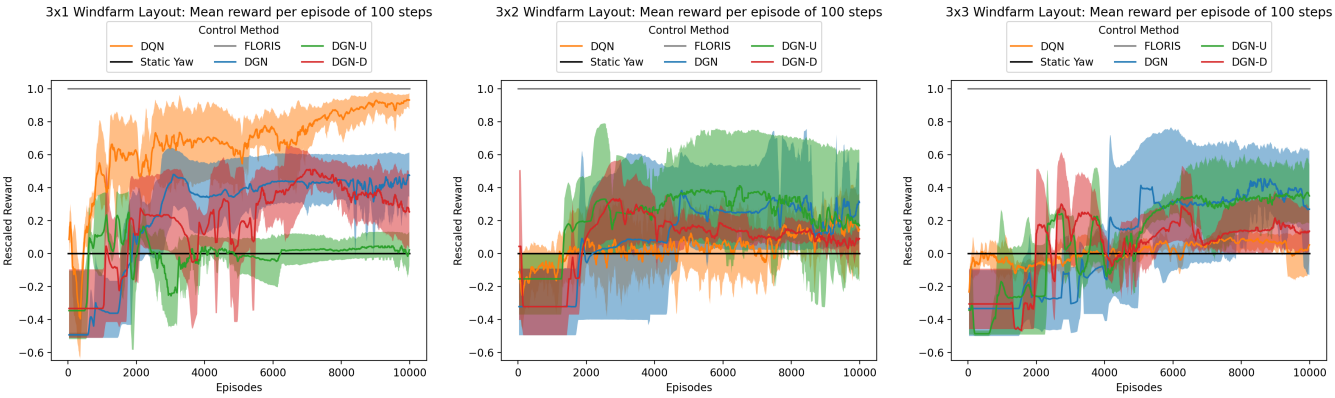
Moving on to the next windfarm, the 3×2 layout shown in Figure 6b clearly show that both DGN and DQN do not learn particularly useful policies when compared to FLORIS, but they do outperform the Static baseline. Towards the end of the training process at 10,000 episodes, the performance of both RL algorithms is comparable, with the upper band of DQN overlapping the lower band of DGN suggesting that the MARL algorithm may be slightly better. The result indicates that expanding the windfarm with an additional row of turbines brings the performances of DGN and DQN closer together. Additionally, the results show that DGN learns its policy earlier, at around 4,500 episodes, as opposed to DQN's gradual increase at 8,000 episodes. However,this is only an indication, as the dip in DGN's performance towards the end suggests that it may not have converged yet.

The results of the 3×3 windfarm layout shown in Figure 7c provide the first clear differentiation between the scalability of DGN to larger windfarms compared to the SARL method. The performance of DQN is comparable to the Static baseline, whereas the performance of DGN is at a level roughly halfway between FLORIS and the Static baselines. This same differentiation is also evident in the results of the largest windfarm layout in Figure 7d. Once again, this plot highlights the differences between MARL and SARL algorithms at scale. DGN learns a policy which is better than the Static baseline, albeit only slightly better. It is still a relatively useful policy compared to remaining stationary. In contrast, DQN learns a disadvantageous policy that consistently underperforms the Static baseline.

Analyzing the performance trends for increasing windfarm sizes further demonstrates the differences between MARL and SARL limitations. DQN produced an extremely useful policy in the smallest windfarm that was comparable to the numerical solution from FLORIS. However, as the windfarm sizes increase, DQN's performance degrades dramatically. In the six turbine windfarm, the performance fluctuates around the Static baseline before exhibiting a slight upwards trend at 8,000 episodes. Increasing the size by three again causes the performance to oscillate at the Static baseline for the entire duration. Finally, in the 16 turbine windfarm, DQN learns a policy worse than the Static baseline, further solidifying the limitations of SARL. On the other hand, DGN shows a completely different trend. Although it does not learn a policy as useful as DQN in the three turbine windfarm, its performance relative to the FLORIS and Static baseline remains stable as the windfarm size increases. While its performance appears to decrease slightly, the overall trend clearly demonstrates the ability of DGN as a MARL algorithm to scale with larger windfarms by maintaining the quality of its learned policy.

4.4 Performance Impact of Directionality in Windfarm Graph Representation

The goal of this section is to answer the third and final sub-research question by investigating the effects of the different windfarm graph representation techniques described in Section 3. Three models are considered: DGN for the undirected representation, DGN-U for the upstream directed representation, and DGN-D for the downstream directed representation. These models are tested on the layouts specified in Section 4.3, with the exception of the 4×4 layout, which is omitted due to time constraints of the lengthy training duration.



(a) 3x1 windfarm layout.

(b) 3x2 windfarm layout.

(c) 3x3 windfarm layout.

Figure 8: Performance of DGN using different directionality in windfarm graph modelling on various windfarm layouts.

The windfarm environment settings and algorithm parameters for DGN and the other baseline methods are outlined in Appendix A, B, and C. Additionally, the rewards in the graphs presented in this section are rescaled, as previously defined, to enable comparisons across different windfarm layouts. The non-scaled results can be found in Appendix E.

The results of these experiments can be found in Figure 8, which displays the performance of all control methods for the specified windfarm layouts in increasing size from left to right. For the smallest windfarm configuration in Figure 8a, the performance of DGN and DGN-D reaches comparable levels with a large overlap between their error bands. However, the power output of DGN-D appears to slope downwards from around 7,000 episodes, indicating that it has not yet converged. On the other hand, DGN-U learns a policy that is comparable to the Static baseline. Analyzing the entire training process, the reward curve of DGN-D and DGN-U contain more peaks and troughs. This uncertainty could be attributed to the reduced communication between agents. The graph suggests that transferring information downstream conveys more informative data for wake control than upstream communication.

The behavior observed in the smaller three turbine windfarm contradicts the behavior displayed in the six turbine windfarm. In Figure 8b, DGN-U converges to a level comparable to DGN, and it appears to perform better between the range of 2,000 and 7,000 episodes, where DGN eventually reaches the same level. Contrarily, DGN-D converges to a level comparable to DQN but drops slightly below it in the last 1,000 episodes. However, DGN-D does still perform better than the the Static baseline. Similar patterns are observed in the nine turbine windfarm in Figure 8c, where DGN-U performs equally well as DGN and DGN-D with DQN. On this layout, the differences between the two pairs are even more distinguishable.

These results demonstrate a discrepancy in the performance between DGN-D and DGN-U depending on the size of the windfarm. In smaller windfarms, modelling downstream directionality is more performant than upstream directionality and vice versa for the larger windfarms. Across all three windfarm sizes, the most optimal directional representations for each size performs comparably to the corresponding performance of DGN.

5 Discussion

This section serves to discuss the methodology and results of all the experiments conducted in Section 4, focusing on the learning rate, windfarm size and windfarm graph representation experiments. These will be discussed in the following order.

The LR experiments clearly demonstrated a correlation between the ability to learn and the LR. The chosen range of LR values was appropriate as it was able to showcase both extreme behaviors. In the lower range of the LR values, where the model converged immediately, the performance was comparable to the Static baseline, where all turbines face the wind. Considering that this is also the starting state of the windfarm environment, it could indicate an ineffective initial random action exploration methodology. The discretized actions of clockwise, counterclockwise, and no movement, sampled uniformly at each time step, result in a random walk behavior. The expectation of the turbine angle is to remain stationary, corresponding to the initial conditions. At each time step, random walk trinomially distributes the turbine positions around the starting position [23], which populates the replay buffer. Since RL algorithms learn from past experiences, having variety in the experiences helps explore a larger state space. The current exploration scheme restricts the exploration as all experience are centered around the starting position, which negatively impacts the learning process regardless of the chosen LR. Therefore, employing a different, more advantageous action sampling scheme could potentially improve the time to converge and quality of the learned policy.

The windfarm size experiments concretely demonstrate that DGN is capable of learning useful policies on all the tested windfarms. Although DGN's performance never reaches the FLORIS baseline, it performs consistently across all windfarm sizes whereas DQN only performs well on the three turbine windfarm. From an algorithmic perspective,

DQN, as an SARL algorithm, decides on turbine actions by using all available state information and learns a policy by taking all turbines into consideration. On the contrary, agents in DGN do not have access to all the information, and each agent learns its own policy independently. It is, therefore, explainable that on a scale where the state and action combinations are still manageable by the SARL algorithm, it yields better performance simply due to the access to information and centralized decision making. However, as windfarm sizes increase, there are simply too many turbine state combinations for the SARL algorithm to explore in a reasonable time. Since the SARL performances on large windfarms were comparable to the Static baseline, the previously discussed limitation of the action space exploration method could have furthered hindered its ability to explore. Nonetheless, DGN still demonstrates a useful policy despite utilizing the same ineffective sampling technique. It could still be a factor that prevented DGN from yielding better performance. Moreover, the lack of extensive hyperparamter tuning could have also suppressed the algorithm's true capabilities. Regardless, the experiments were sufficient to demonstrate DGN's scalability to larger windfarms and highlight the differences between DGN and DQN. The chosen windfarm sizes and layouts were appropriate as they revealed the performance trend in relation to windfarm size.

The windfarm modelling experiments showed that, with the current setup, the additional communication supported by undirected edge modelling does not appear to result in performance benefits over directed edges, as both approaches yield similar results. Since the undirected representations allows for more inter-agent communication, it suggests that the additional information transferred are not be significantly more useful for learning a policy. Surprisingly, the differences between upstream and downstream communication for AWC are inconclusive and requires further experiments, as outcomes are contradictory. Intuitively, one could argue that upstream communication should outperform downstream communication because a turbine should only steer its wake when there are downstream turbines. If there are no downstream turbines, it would always face the wind to maximize output. This knowledge is only achievable by communicating upstream. The results for larger windfarm sizes seem to support this intuition, however, at smaller sizes, the opposite behaviour is observed. Possible explanations could involve a relationship between the size and complexity of the windfarm and directionality of communication. Further investigation is required to draw further conclusions.

6 Conclusions and Future Work

Single agent RL algorithms have been successfully applied to the active wake control problem in windfarms with a limited number of turbines. However, these methods face challenges when scaling to real-life windfarms with a significantly larger number of turbines. The exponential growth in size of the state and action spaces makes it difficult for agents to learn useful policies within a reasonable time frame. Multi agent RL algorithms provide a solution to this problem by allowing each agent to learn its own actions independently. This paper presented DGN as a potential MARL algorithm suitable for AWC application for windfarms. It demonstrated that on small windfarms, DGN can be used for AWC but does not outperform DQN, the SARL benchmark. However, as the windfarm sizes increase, the results clearly show that DGN performance remains stable and outperforms the SARL algorithm, which suffers from scalability issues. Lastly, this paper explored the use of directional graph modelling of the windfarms. It was found that directed graphs perform comparably to undirected graphs, but the differences between upstream and downstream directionality are still unclear and require further experimentation.

The results establish DGN as a viable algorithm for AWC in windfarms. While it outperforms certain baselines, DGN doess not learn policies that are comparable to FLORIS numerical solutions. The setup of the experiments focused on tuning the learning rate to tailor the algorithm for AWC, but other influential factors such as epsilon-greedy exploration parameters and batch size were not explored. Further investigation into these hyperparameters, as well as potentially additional ones, could reveal the full capabilities of DGN. Additionally, the learning behavior of DGN could be studied by addressing the issue of the uniform random action space sampling scheme mentioned in Section 5. Alternative exploration schemes could be devised to generate states with diverse combinations of turbine angles. Approaches such as yawing turbines back and forth at different frequencies and directions to cover a wider range of turbine angle combinations. Furthermore, with more time budget, experiments should be repeated more times for reliability. Moreover, experiments should be run with more episodes as larger windfarm layouts may require more episodes to converge to a policy.

To further research the application of DGN to AWC in windfarms, experiments can be conducted using more realistic wind conditions and windfarm layouts. The experiments in this paper utilized grid layouts and static wind conditions to simplify the problem and focus on applying DGN. However, incorporating stochastic wind models that include variations in wind speed and direction would introduce more complexity to the state space and required policy, making it more representative of real-world windfarm conditions. Furthermore, grid layouts are seldomly employed in practice, as they inherently induce high wake interaction. Turbines are instead positioned using specialized optimization techniques [24]. It would be valuable to conduct experiments using layouts based on real-life windfarms, either using the entire windfarm or subsets of it. These extensions further test the viability of DGN for real world AWC deployment. Lastly, there are numerous possibilities for further investigation in windfarm graph modeling. These include exploring different approaches to graph modeling, such as the directionality graph modeling proposed in this paper, or dynamically updating graph representations to reflect changes in turbine and wind conditions. These avenues of research have the potential to enhance the understanding and capabilities of DGN in the context of active wake control in windfarms.

7 Responsible Research

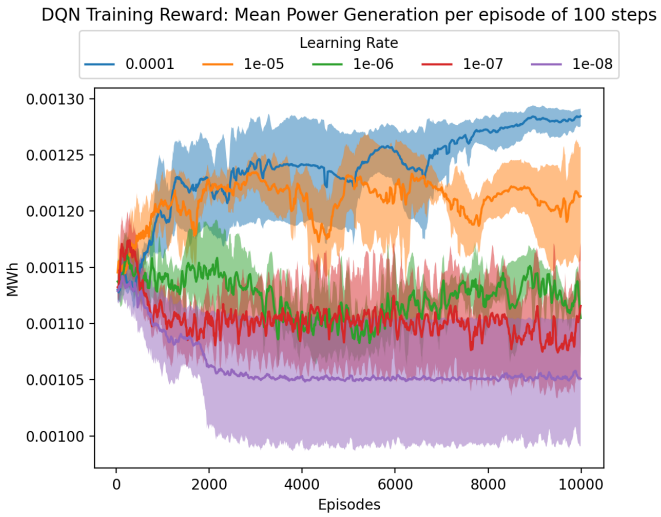
The focus of the research in this paper was the investigation of applying a specific method, namely the DGN MARL algorithm, to the AWC problem in windfarms. The goal of AWC is to improve the power efficiency and output of windfarms as a whole. From an AI perspective, the research of a crucial renewable energy source can be considered ethical for the potential societal benefits that stem from it. Especially, in this case, which is the use of RL to control a system where onsite human operation and maintenance are phasing out into monitoring roles [25]. However, ethical considerations still arise from the potential implications of such technologies. Increased autonomy raises a concern for the loss of human decision making. Although most ethical issues concern direct human interaction, the inherent black box nature of ML gives rise to risk in the explainability of decision making and its reliability in various potentially adversarial situations [26].

To ensure the reproducibility of the research, many considerations have been taken. Firstly, all the algorithm implementations and frameworks that have been used for the experiments are open source and have been referenced. The implementation for the experiments in this paper is also open sourced in Github[1]. Standardization in implementation is the first step towards achieving reproducibility. Given the same implementations, hyperparameters and seeds can drastically alter results. As previously discussed in Section 4.1, with the exception of tuning LR, all hyperparameters are kept constant for all experiments. These are also documented such that the exact model configurations can be reproduced. Seeds, on the other hand, control the aspects involving randomness in the experiments. An example of this is agent random action space exploration which can influence the policies that are ultimately learned. Although fixing the seed removes the stochasticity in random sampling, the sensitivity of seeds can negatively or positively skew the results and lead to potentially misleading conclusions [27]. Consequently, results presented in this paper are the mean over multiple random seeds such that they can be reproducible in the average case. Finally, all the methodology and results used for the experiments have been truthfully and fully reported so that conclusions drawn in this paper can also be reached in a reproducible manner.

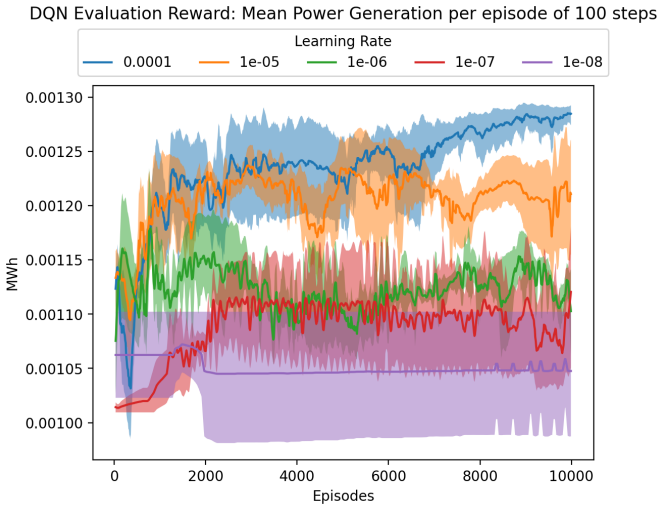
A Deep Q-Network Hyperparameters

Tuning the learning rate involved comparing performance of five values ranging from 10^{-4} to 10^{-8}. Experiments were repeated 3 times and the 10th and 90th percentile are used for error bands.

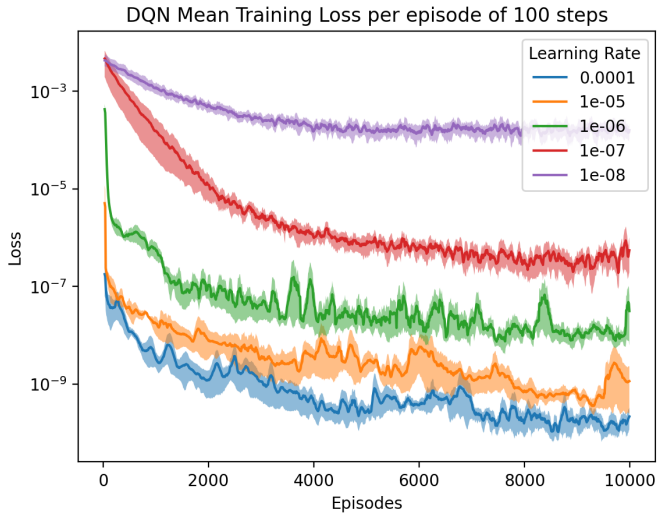
[1] https://github.com/JeffersonYeh/DGN-awc-windfarm



(a) Training Reward



(b) Evaluation Reward



(c) Training Loss

Figure 9: Tuning DQN on various learning rates.

The final setup of hyperparameters used are the following in Table 1.

Parameter	Value
ϵ_{start}	0.9
ϵ_{end}	0.05
ϵ_{decay}	1000
learning rate	10^{-4}
batch size	128
replay buffer size	10000
γ	0.99
τ	0.005
training episodes	10000
training steps per episode	100
evaluation steps per episode	100
windfarm action representation	yaw
action discretization step size	1

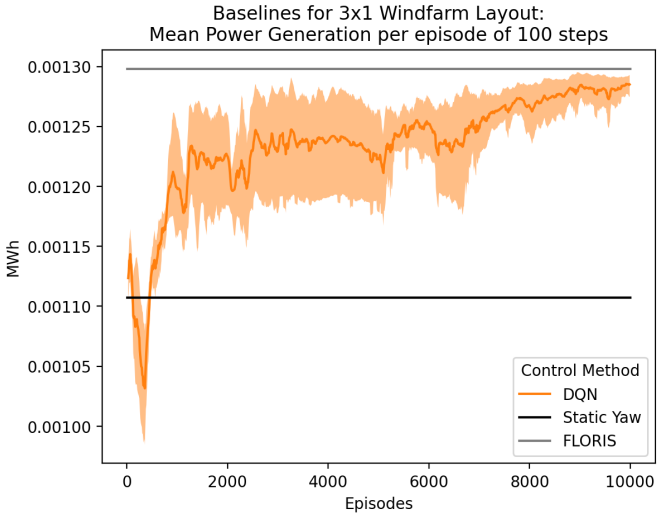
Table 1: DQN Hyperparameters

B Baseline Performance

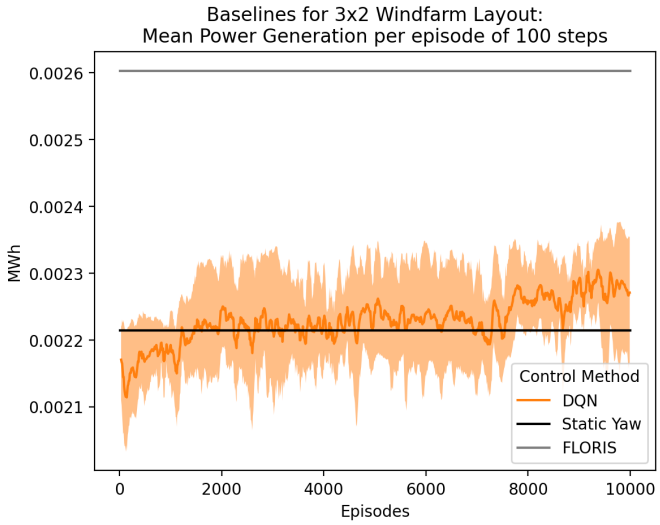
The baseline performances of FLORIS, Static Yaw, and DQN for all the layouts used in the experiments.

Windfarm Layout	Static Yaw	FLORIS
3×1 wind tunnel	0.001107	0.001298
3×2 grid	0.002214	0.002603
3×3 grid	0.003322	0.003909
4×4 grid	0.005471	0.006817

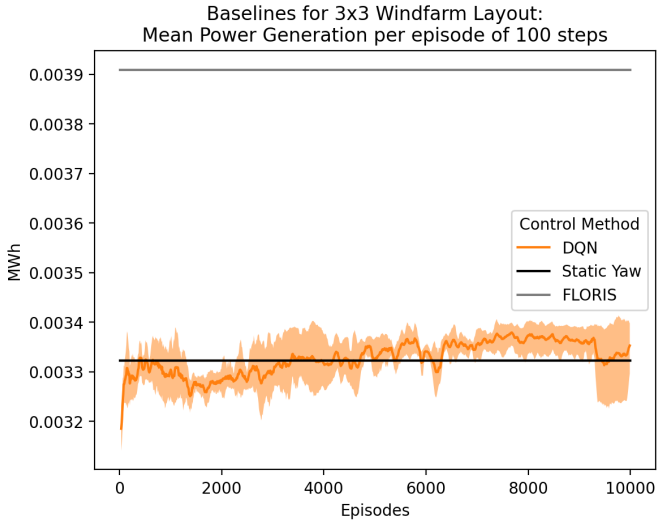
Table 2: Control method baseline reward per time step



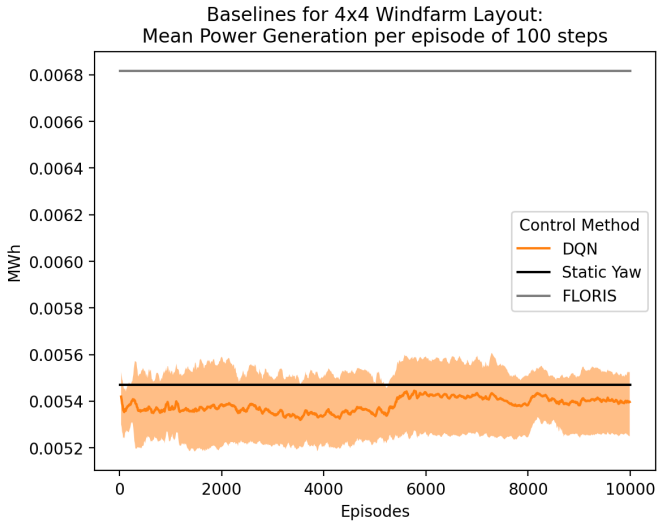
(a) 3×1 windfarm layout.



(b) 3×2 windfarm layout.



(c) 3×3 windfarm layout.



(d) 4×4 windfarm layout.

Figure 10: Baseline performance on various windfarm layouts

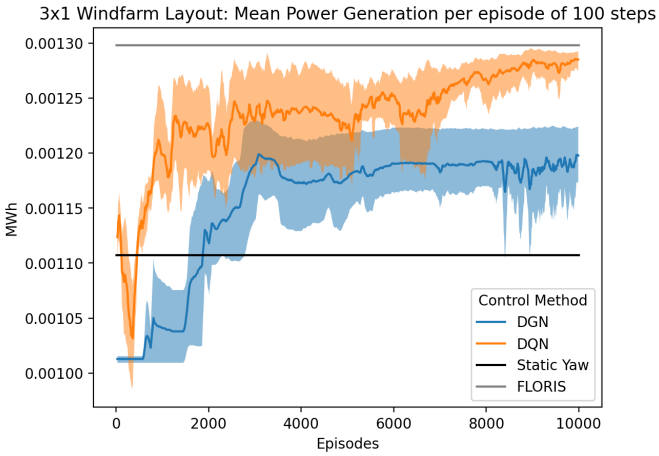
C DGN Hyperparameters

The final setup of hyperparameters used for DGN are summarized in Table 3.

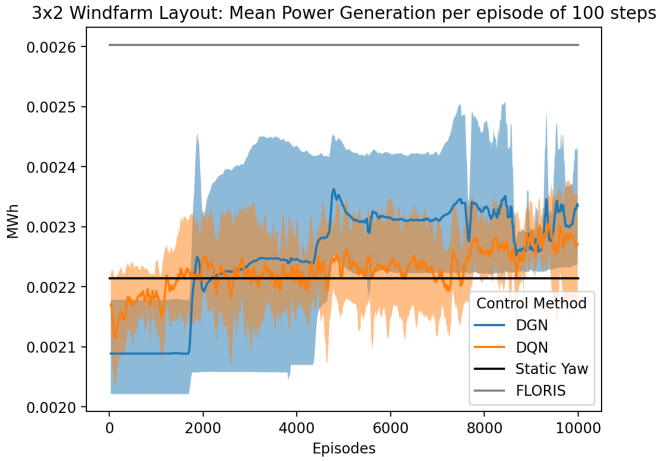
Parameter	Value
ϵ_{start}	0.9
ϵ_{end}	0.05
ϵ_{step}	-0.0004
learning rate	10^{-7}
batch size	128
replay buffer size	1000000
epoch	25
γ	0.99
hidden layer dimension	64
training episodes	10000
training steps per episode	100
evaluation steps per episode	100
learning start episode	100
windfarm action representation	yaw
action discretization step size	1

Table 3: DGN Hyperparameters

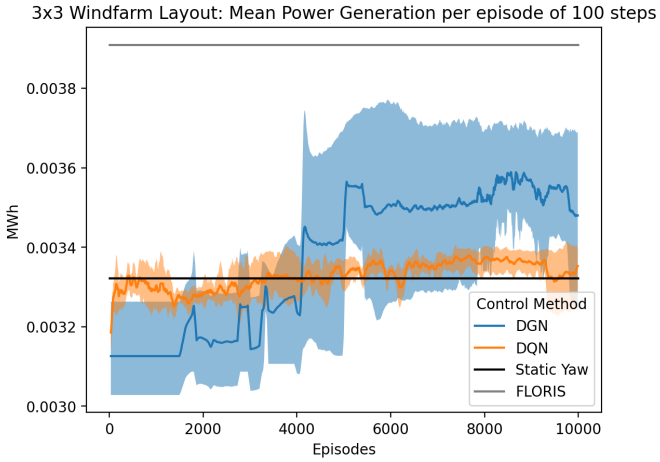
D Windfarm Size Experiments



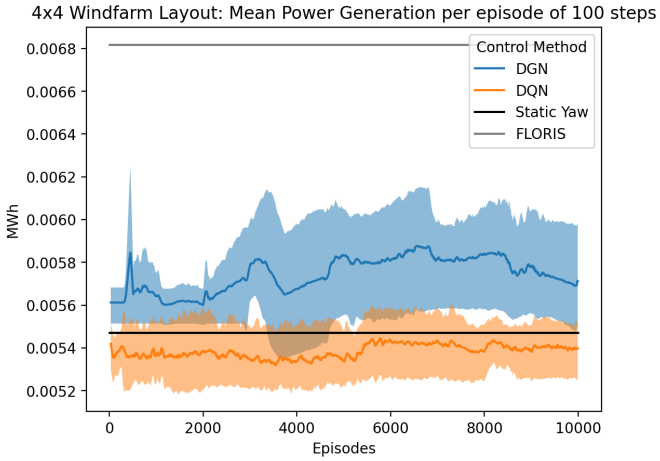
(a) 3×1 windfarm layout.



(b) 3×2 windfarm layout.



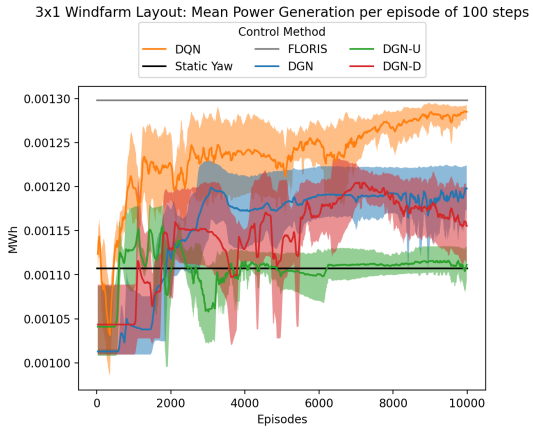
(c) 3×3 windfarm layout.



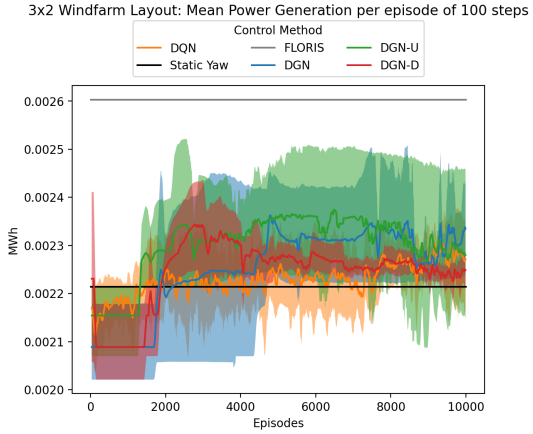
(d) 4×4 windfarm layout.

Figure 11: Non-scaled DGN and baseline performances on changing windfarm sizes.

E Windfarm Graph Representation Experiments



(a) 3×1 windfarm layout.



(b) 3×2 windfarm layout.

(c) 3×3 windfarm layout.

Figure 12: Non-scaled performance of DGN using different directionality in windfarm graph modelling on various windfarm layouts.

References

[1] O. Edenhofer, R. Pichs-Madruga, Y. Sokona, K. Seyboth, and et al., "Summary for policymakers," *Renewable Energy Sources and Climate Change Mitigation: Special Report of the Intergovernmental Panel on Climate Change*, p. 3–26, 2011.

[2] R. Wiser, Z. Yang, M. Hand, O. Hohmeyer, D. Infield, P. H. Jensen, V. Nikolaev, M. O'Malley, G. Sinden, A. Zervos, and et al., "Wind energy," *Renewable Energy Sources and Climate Change Mitigation: Special Report of the Intergovernmental Panel on Climate Change*, p. 535–608, 2011.

[3] M. F. Howland, S. K. Lele, and J. O. Dabiri, "Wind farm power optimization through wake steering," *Proceedings of the National Academy of Sciences*, vol. 116, pp. 14 495–14 500, 7 2019.

[4] P. Stanfel, K. Johnson, C. J. Bay, and J. King, "Proof-of-concept of a reinforcement learning framework for wind farm energy capture maximization in time-varying wind," *Journal of Renewable and Sustainable Energy*, vol. 13, p. 043305, 7 2021.

[5] 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, 6 2021.

[6] H. Dong, J. Xie, and X. Zhao, "Wind farm control technologies: from classical control to reinforcement learning," *Progress in Energy*, vol. 4, p. 032006, 7 2022.

[7] B. M. Doekemeijer, D. van der Hoek, and J.-W. van Wingerden, "Closed-loop model-based wind farm control using floris under time-varying inflow conditions," *Renewable Energy*, vol. 156, pp. 719–730, 2020.

[8] G. Neustroev, S. P. Andringa, R. A. Verzijlbergh, and M. M. De Weerdt, "Deep reinforcement learning for active wake control," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 2022, pp. 944–953.

[9] P. Stanfel, K. Johnson, C. J. Bay, and J. King, "A distributed reinforcement learning yaw control approach for wind farm energy capture maximization," in *2020 American Control Conference (ACC)*, 2020, pp. 4065–4070.

[10] K. Zhang, Z. Yang, and T. Başar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. Cham: Springer International Publishing, 2021, pp. 321–384.

[11] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Applied Intelligence*, 2022.

[12] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[13] G. Neustroev, M. de Weerdt, R. Verzijbergh, and S. Andringa, "Source code and data for the experiments

presented in deep reinforcement learning for active wake control," 2022. [Online]. Available: https://data.4tu.nl/articles/_/19107257/1

[14] NREL, "Floris wake modeling and wind farm controls software," v2.4. [Online]. Available: https://github.com/nrel/floris

[15] J. Jiang, "pytorch_dgn," 2021. [Online]. Available: https://github.com/jiechuanjiang/pytorch_DGN

[16] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 561–591.

[17] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, "What matters in on-policy reinforcement learning? a large-scale empirical study," *arXiv*, 2020.

[18] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *arXiv*, 2018.

[19] H. N and P. A. G, "A brief study of deep reinforcement learning with epsilon-greedy exploration," *International Journal of Computing and Digital Systems*, vol. 11, pp. 541–551, 1 2022.

[20] P. A. Fleming, A. P. J. Stanley, C. J. Bay, J. King, E. Simley, B. M. Doekemeijer, and R. Mudafort, "Serial-refine method for fast wake-steering yaw optimization," *Journal of Physics: Conference Series*, vol. 2265, p. 032109, 5 2022.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv*, 2013.

[22] PyTorch, "Reinforcement q learning." [Online]. Available: https://github.com/pytorch/tutorials/blob/main/intermediate_source/reinforcement_q_learning.py

[23] P. Consul, "On some probability distributions associated with random walks," *Communications in Statistics - Theory and Methods*, vol. 23, no. 11, pp. 3241–3255, 1994.

[24] N. Charhouni, M. Sallaou, and K. Mansouri, "Realistic wind farm design layout optimization with different wind turbines types," *International Journal of Energy and Environmental Engineering*, vol. 10, pp. 307–318, 2019.

[25] X. Chen, M. A. Eder, A. Shihavuddin, and D. Zheng, "A human-cyber-physical system toward intelligent wind turbine operation and maintenance," *Sustainability*, vol. 13, no. 2, 2021.

[26] S. L. Piano, "Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward," *Humanities and Social Sciences Communications*, vol. 7, p. 9, 6 2020.

[27] S. Bethard, "We need to talk about random seeds," *arXiv*, 2022.