Imperial College London
Department of Earth Science and Engineering
MSc in Environmental Data Science and Machine Learning

Independent Research Project
Final Report

# Deep Reinforcement Learning to Enhance Carbon Abatement Potential of Battery Energy Storage Systems

by

Christopher Saad

Email: christopher.saad22@imperial.ac.uk
GitHub username: edsml-cs1622
Repository: https://github.com/ese-msc-2022/irp-cs1622

Supervisors:
Alexander Abou-Jaoude
Dr. Pablo Brito-Parada
Dr. Jacqueline Edge

Company:
Adaptogen Capital
68 Dalling Road, London, England, W6 0JA

September 2023

**Table of Contents**


Abstract 3

I. Introduction 3

II. Deep RL Background 5

III. Project Objectives 6

IV. Methodology 6

i. Calculating the Carbon Abated by BESS 6

ii. Optimizing the Charging of BESS for Carbon Abatement 7

iii. Assumptions 10

V. Results and Discussions 11

i. DRL Battery Agent Training 11

ii. Carbon Abated by Real BESS and Battery Agent 13

VI. Conclusion 16

References 17

Appendix 19

A1. Notable Test Setups and Results 19

A2. Notable Agent Test Plots 20

A3. Energy Output Profiles for UK BESS in the Last Year 22

**Abstract**

Battery Energy Storage Systems (BESS) contribute significantly to the decarbonization of the UK's electricity grid by balancing the supply and demand of electricity during times of varying renewable energy output. This project aims to establish a carbon abatement benchmark for BESS in the UK under current trading strategies and explore machine learning-optimised trading strategies to enhance carbon abatement. The project's methodology includes an analysis of historical battery trading data and grid carbon intensity data, and the construction and optimisation of a deep reinforcement learning model. The proposed approach is model-free, meaning it is adaptable to batteries with different characteristics and constraints. The final objective is to seek an optimal charging strategy to maximise the carbon abated by a battery storage system while staying within its operational limits. Despite the complexities of energy market modelling and forecasting, this research aims to provide insights and strategies to further the UK's transition towards carbon neutrality.

## I. Introduction

Battery Energy Storage Systems (BESS) have been increasingly integrated into the UK energy system, facilitating the transition to net zero by 2050, and the decarbonization of the electricity system by 2035 [1]. The popularity of BESS has grown significantly due to its ability to balance the fluctuations in renewable energy generation, and hence its ability to balance the grid as it increasingly relies on renewables. They do this by time-shifting renewable energy generation, and displacing more carbon-intensive energy sources at times when renewables are unavailable, and thereby displacing carbon emissions. Figure 1 [1] shows the storage volumes needed to meet the goals set out by the National Grid Electricity System Operator (NGESO), as intermittent renewables accelerate and base load coal, nuclear, and combined cycle gas turbine generation reduce in the next few years.

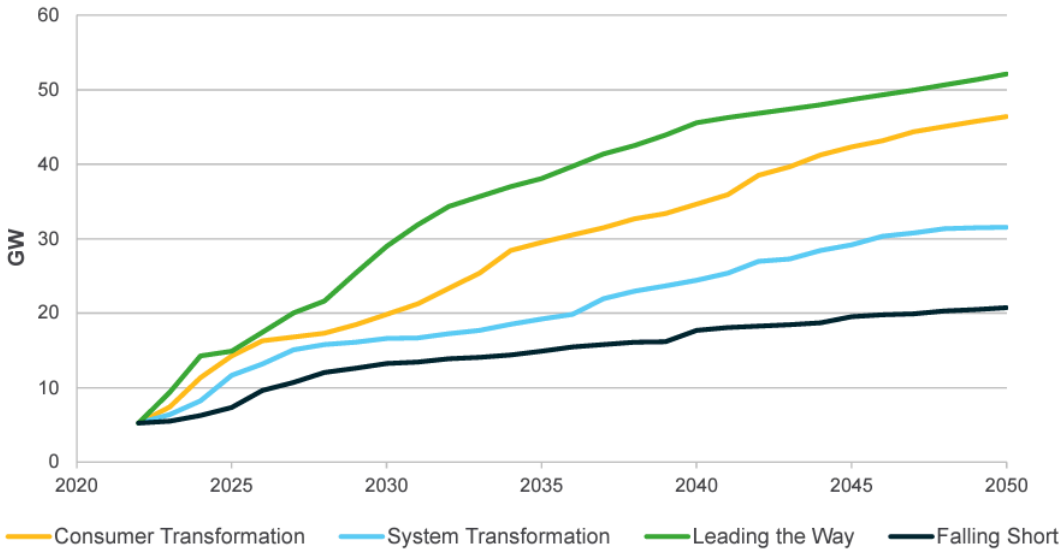



***Figure 1:*** *FES 2023 Great Britain Total Electricity Storage Needs* [1]

BESS primarily operate across the wholesale (year-head, day-ahead, intra-day), balancing mechanism (BM), and ancillary service (AS) electricity markets. As displayed in Figure 2, the wholesale market covers trades made from years ahead up to one hour ahead of the delivery period. The BM takes place in the hour before delivery, in which NGESO balances the supply and demand of electricity by paying generators to increase or reduce their generation. Final Physical Notifications (PNs) are delivered by the grid prior to each settlement period. Each PN describes the expected time of delivery and level of energy

to be imported or exported from the grid by each energy supplier for the following settlement period (SP), a 30-minute period in which these trades are made. Additionally, AS is used to maintain the stability of the grid on a sub-second basis [2]. AS is increasingly dominated by BESS, leading to diminishing carbon emissions in this sector [3]. This study examines longer duration batteries, which are becoming more common [4] and are less focused on AS and more on energy arbitrage, and hence carbon abatement. The carbon offset by a battery in wholesale markets is typically measured by multiplying the amount of energy imported or exported from the battery by the carbon intensity (CI) of the grid during every SP [5]. More complex methods can be used to estimate the carbon abated in AS, but since this quantity is approaching zero, we will focus on estimating carbon abated in wholesale markets using PN data.

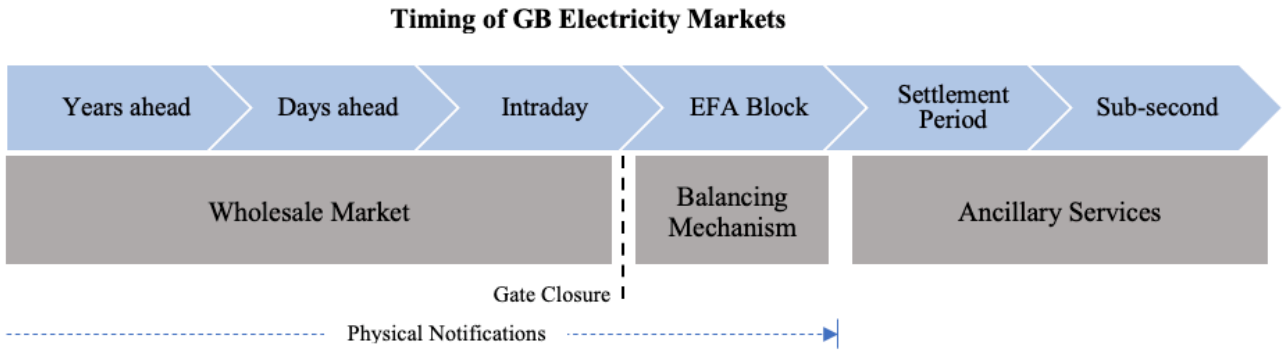



***Figure 2:*** *Timings of UK electricity markets in which BESS primarily operate*

BESS typically profit by buying and charging energy during times of oversupply, and selling and discharging during times of peak demand. Times of oversupply and peak demand typically align with times of high renewable output and low renewable output, respectively. Therefore, we hypothesise that maximising profitability should nearly maximise carbon abatement. Academic research and commercial efforts to enhance the profitability of BESS are growing, but explicitly increasing carbon abatement in these systems is still unexplored. To investigate this potential, we develop an approach to quantify the carbon abated by a battery storage system and a model to enhance a battery's charging and discharging schedule for this parameter.

To create a realistic model of BESS, we consider their operational constraints and characteristics [33]. The most obvious constraints are that a battery's power output cannot go beyond its power capacity, and its stored energy cannot exceed its energy capacity. BESS also receive warranties to charge and discharge up to a certain number of cycles per day, and those that cycle more aggressively can degrade faster. They also suffer from inefficiencies when storing the energy, preventing them from discharging as much as they charge [6].

While explicitly optimizing BESS' charging for carbon abatement is unexplored, many approaches exist for modelling and optimising BESS, electric vehicle (EV) batteries, and other energy storage systems for various objectives. A classical approach to optimisation is multi-integer programming (MIP), in which an objective function is minimised. Several studies modelled BESS bidding strategies using this approach, considered the battery's activation period [7], degradation rate [8], cycle life [9], power capacity, energy capacity, state of charge [10], and other chemical features [11] as separate constraints in the model. While less flexible for different systems and environments, MIP provides more control of the constraints of the system. Others have used evolutionary algorithms to optimise trading strategies [12], constraints of PV batteries [13], and charging times of EVs [14]. This heuristic-based approach is generalizable to most optimisation problems, but it might not find optimal solutions in dynamic environments [15].

Recently, deep reinforcement learning (DRL) has gained traction for optimising BESS and EV charging, and energy and stock trading. [16] used DRL to create strategic day-ahead bidding strategies for BESS

and [17] used it to schedule EV charging using real-time price signals. DRL, specifically a model-free approach, demonstrates flexibility when dealing with dynamic, uncertain environments [15]. In this paper we choose a model-free DRL approach to optimise BESS charging and discharging strategies for maximising carbon abatement in wholesale markets.

## II. Deep RL Background

Reinforcement learning (RL) solves problems framed as a Markov Decision Process (MDP), in which an agent interacts with an environment through successive observations, actions, and rewards. When an agent encounters a state $s$, it samples an action from a policy, receives a scalar reward $r$, and transitions to the next state $s'$ [18]. The agent learns to select an action that maximises the expected discounted reward, which is represented by the recursive bellman equation [19]:

$$Q^*(s,a) = E_{s'\sim P}\left[r(s,a) + \gamma \max_{a'} Q^*(s',a')\right] \quad (1)$$

where $a'$ is the next action taken in $s'$, $s' \sim P$ denotes that $s'$ is sampled from a probability distribution $P$, and $\gamma$ is the discount factor. A larger $\gamma$ will give more weight to long-term rewards. $Q^*(s,a)$ is usually approximated by a look-up table in RL, but for high dimensional, continuous state spaces, this table can become too large. In this case, an artificial neural net (ANN) can be used to approximate $Q^*(s,a)$. Sometimes, a recurrent neural network (RNN) is used to consider historical data, such as electricity prices, when selecting an action [17, 20, 21]. A key to optimally solving MDPs is finding a balance between exploration of the environment and exploitation of known information. To do this, MDPs typically use an ε-greedy strategy, where ε is the exploration rate.

In this paper, we will consider model-free DRL approaches. Model-based DRL techniques simulate experiences based on the model of the environment to inform decision making. This method can suffer if the model of the environment is inaccurate, and it is typically computationally expensive [22]. Model-free DRL techniques learn directly from experiences gathered through interactions with the environment, and do not rely on an explicit model of the environment. This approach leverages historical data and captures complex environment dynamics but may require many samples [23].

Within model-free DRL, there are policy-based and value-based algorithms. Policy-based, or actor-only, algorithms directly optimise a policy and perform gradient ascent to maximise the reward. Policy-based methods may sample inefficient, but they directly optimise the parameter of interest [18]. In value-based, or critic-only, methods the goal is to approximate a value function based on the bellman equation, with the idea that an optimal value function will lead us to a near-optimal policy [24]. Value-based methods are more sample efficient because they can learn from previous experiences, but they can be unstable when approximating the value function [18]. To build on the strengths of both policy and value-based methods, an actor-critic approach was introduced [24], where a critic network is used to approximate the value function and update the actor network's policy parameters with gradient ascent.

DRL algorithms can be further classified into off-policy and on-policy algorithms. Off-policy algorithms can learn from data collected using older policies. Some of these algorithms leverage an idea called experience replay, where the agent's experiences are stored in a replay buffer. During learning, the agent takes a minibatch of random samples from the replay buffer to perform Q-learning updates [25]. On the other hand, on-policy algorithms only make updates using the latest policy. These algorithms suit rapidly changing environments where quick updates may be necessary. They can be less computationally expensive than off-policy algorithms, but they may be less sample efficient. In this paper we explore both off-policy and on-policy actor-critic algorithms that support continuous action and state spaces. State-of-

the-art algorithms for continuous control that we will consider are DDPG (Deep Deterministic Policy Gradient) [26], SAC (Soft Actor-Critic) [27], and TD3 (Twin-Delayed DDPG) [28].

## III. Project Objectives

1. Benchmark carbon abated by BESS assets using status quo trading strategies: The first objective is to develop a quantification method to calculate carbon abated by BESS in the UK wholesale market.
2. Develop an optimisation model: The second objective is to develop a DRL model that will simulate optimal trading strategies for BESS, resulting in increased carbon abatement.
3. Compare different optimisation strategies: The final objective will be to compare the implemented model's results with the benchmarked carbon abatement values and draw conclusions.

## IV. Methodology

### *i. Calculating the Carbon Abated by BESS*

Using NGESO's CI API [29] and Elexon's PN Stream API [30] we can create the data necessary to calculate the carbon abated by BESS. In order to calculate the carbon abated between two dates, the CI data and PN data are retrieved from their respective APIs for that time frame. The APIs have limitations on retrieving up to one month of data at a time, so each month of data from both datasets are retrieved in parallel, and their results are combined. We also use a local caching mechanism to prevent redundant API calls. The PN data is more granular than the CI data, so the corresponding CI value for each PN is appended to its row. Additional modifications had to be made to the data to account for missing values and time gaps in the PN data. If the end of one PN does not lead directly to the start of the next PN, we assume a linear change in level between both PNs and create a new PN for the missing time.

We derive the following equation to calculate the carbon abated by a battery:

$$CarbonAbated = (CI_D - CI_C) * E_D - (E_C - E_D) * CI_C \quad (2)$$

where $CI_D$ and $CI_C$ are the weighted averages of the grid's CI while the battery was charging and discharging, respectively, and $E_C$ and $E_D$ are the total amounts of energy charged and discharged by the battery in that time frame. We can separate this equation into two main parts:

1) $(CI_D - CI_C) * E_D$ quantifies the carbon abated when discharging. The battery offsets carbon by discharging at a higher CI than when it charged, or it contributes carbon by discharging at a lower CI than when it charged.

2) $(E_C - E_D) * CI_C$ accounts for differences in total energy charged and discharged. If the battery charges much more than it discharged, it misses the opportunity to abate carbon, and contributes carbon with the CI it used during charging. Theoretically a battery should never discharge more than it charges, and this term would not increase $CarbonAbated$.

To solve this equation, we first calculate the total energy charged and discharged using the PN data. Each PN's energy output is calculated by integrating the power output, assuming a linear rate of charge:

$$energyOut = 0.5 * (levelFrom + levelTo) * time \quad (3)$$

where $levelFrom$ is the power level at the start of the PN, and $levelTo$ is the power level at the end of the PN. After this calculation, we have new feature $energyOut$, which is the total energy outputted by

the battery for each PN. We denote all negative and positive values of $energyOut$ as $energyCharged$ and $energyDischarged$, respectively. Using these values, we can calculate $E_C$, $E_D$, $CI_C$, and $CI_D$:

$$E_C = \sum_{t=start}^{t=end} energyCharged_t \tag{4}$$

$$E_D = \sum_{t=start}^{t=end} energyDischarged_t \tag{5}$$

$$CI_C = \sum_{t=start}^{t=end} \frac{CI_t * energyCharged_t}{E_C} \tag{6}$$

$$CI_D = \sum_{t=start}^{t=end} \frac{CI_t * energyDischarged_t}{E_D} \tag{7}$$

where $CI_t$ is the CI of the grid at time t. We plug all the calculated values back into (2), apply scalars to adjust the units, and are given the carbon abated by the asset over the time analysed. The outputs generated show a comparison of the results of all the analysed assets, along with a comparison of $E_C$, $E_D$, $CI_C$, and $CI_D$. The code was developed as a standalone python API for Adaptogen Capital and can analyse hundreds of BESS at a time without any special hardware requirements.

### *ii. Optimizing the Charging of BESS for Carbon Abatement*

We formulate this problem as an MDP with discrete timesteps. Each timestep *t* represents one SP, matching the granularity of the CI dataset. At each timestep, the agent, representing a battery, receives an observation of its current state $s_t$, takes an action $a_t$, and receives a scalar reward $r_t$. The objective of the agent is to determine the optimal action-value function, described by (1).

The agent is defined and trained using OpenAI's open-source libraries Gymnasium and StableBaselines3. Gymnasium is used to define the environment, state space, action space, state transition function, and reward function. For training, we use StableBaselines3's implementations of SAC, DDPG, and TD3, modify their underlying ANN architecture, and tune their hyperparameters. We define a class $BatteryEnv$, a subclass of $gymnasium.Env$, to represent a battery. This environment can be described in four parts:

1) Observation: The observation space of our agent at time $t$ is made up of seven values:

$$s_t = [SoC_t, CI_t, f_{t+n}^{min}, f_{t+n}^{max}, f_{t+n}^{mean}, C_t^D, C_t^C]$$

where $SoC_t$ is the battery's state of charge at time $t$ as a percentage of its energy capacity, $CI_t$ is the national grid's CI at time $t$, $f_{t+n}^{min}, f_{t+n}^{max}, f_{t+n}^{mean}$ are the minimum, maximum, and mean, respectively, of the forecasted CI values from timesteps $t+1$ to $t+n$, and $C_t^C$ and $C_t^D$ are a percentage of the maximum cycles charged and discharged, respectively, by the battery in the current day. We use the forecast values from NGESO's CI API to simulate the data a battery would access in real-time, replacing the common

use of RNNs. Additionally, the environment tracks the current SP, the battery's power and energy capacity, and the current timestep's energy output.

2) Action: The action space is defined as a continuous space $A \in [-1,1]$. Each action $a_t \in A$ represents the power output of the battery at time $t$, as a percentage of its power capacity. Action $a_t$ is positive when the battery is discharging and negative when the battery is charging. To get the energy output of the battery in MWh for $a_t$, we multiply $a_t$ by the power capacity and the time of the SP, which is 0.5 hours:

$$energyOut_{a_t} = 0.5 * a_t * P_{max} \tag{8}$$

where $P_{max}$ is the power capacity of the agent. We assume the agent maintains the power level of $a_t$ throughout the SP. If $a_t$ causes the agent to exceed any constraints, the action is clipped to reach the constraints' maximum values, ensuring they are never actually exceeded.

3) Transition Function: At each step, the transition to the next state can be defined by

$$(s_{t+1}, r_t, d) = step(s_t, a_t) \tag{9}$$

where $d$ is a flag denoting whether $s_{t+1}$ is a terminal state, and $r_t$ is the reward for taking $a_t$ in $s_t$. When a terminal state is reached, the current episode ends, a reset transition is triggered, and a new episode begins. We define a state to be terminal if any of the following conditions are met:

$$SoC_t > 1$$
$$SoC_t < 0$$
$$C_t^C > 1$$
$$C_t^D > 1$$
$$SP = 48$$

If a reset transition is triggered, we begin the next episode. If the next episode is the start of a new day, we reset $C_t^C$ and $C_t^D$ to 0. We don't modify $SoC_t$ from its previous value, unless it is out of bounds, then we clip the value to the closest boundary.

4) Reward: The agent is given an immediate reward at each timestep and an additional reward at the end of an episode, described by:

$$r_t^C = \begin{cases} energyOut_{a_t} * -5 & CI_t - f_{t+n}^{mean} > 0 \\ energyOut_{a_t} & CI_t - f_{t+n}^{mean} \leq 0 \\ energyOut_{a_t} * 5 & CI_t < f_{t+n}^{min} \end{cases} \tag{10}$$

$$r_t^D = \begin{cases} energyOut_{a_t} * -5 & CI_t - f_{t+n}^{mean} < 0 \\ energyOut_{a_t} & CI_t - f_{t+n}^{mean} \geq 0 \\ energyOut_{a_t} * 5 & CI_t > f_{t+n}^{max} \end{cases} \tag{11}$$

$$r_{end} = \begin{cases} -penalty & SP \neq 48 \\ C_t^{C\,10} + C_t^{C\,10} & SP = 48 \end{cases} \tag{12}$$

where $r_t^C$ and $r_t^D$ are the immediate rewards when a charging or discharging action is taken, respectively, $r_{end}$ is the terminal reward, and $penalty$ is a scalar penalty for exceeding a constraint. The rewards are mainly concerned with where the current CI is relative to the forecasted CI. For example, if the agent

charges at a CI that is higher than the mean of the forecasted CI, the reward is negative and weighted by the amount of energy charged. The second part of (12) motivates the agent to charge and discharge as much as possible within its cycle limits. The reward function was initially defined by an equation derived from (3), however this led the agent to focus too much on immediate rewards. With the adapted reward structure, the agent still learns to discharge at time of high CI and charge at times of low CI, but it also determines whether there will be a more optimal time to act in the future.

To initialise a $BatteryEnv$ instance for training, we must first pre-process NGESO CI data. We first collect the CI data and forecasts for the last four years, which is about 91000 timesteps. We use a rolling window of size $n$ to get $f_{t+n}^{min}, f_{t+n}^{max}, and\ f_{t+n}^{mean}$. We also calculate the SP of each row using the index of the data. We then create an 80-20 train-test split, leaving over 1 year of test data. The mean and standard deviation of training CI values are used to standardise all the CI and forecast values. We initialise $BatteryEnv$ with a power capacity of 25MW, an energy capacity of 40 MWh, and an initial state:

$$SoC_t = 0.5,\ C_1^D = C_1^C = 0\ ,$$

An initial charge of 50% allows room for initial exploration. The other state variables are taken from the first row of training data.

Our final solution uses DDPG due to its balance between sample efficiency and stability. It adapts the idea of Deep Q-Learning [25] to continuous actions spaces by using off-policy data to learn a Q-function and the Q-function to learn a deterministic policy. DDPG also introduces the idea of action noise to control the exploration rate of the agent.

Algorithm 1 [19] shows how the agent is trained using DDPG. The algorithm begins by initializing a random critic network $Q_\Phi$, actor network $\mu_\theta$, and empty replay buffer $\mathcal{D}$. A critic target network $Q_{\Phi_{targ}}$ and actor target network $\mu_{\theta_{targ}}$ are then created with the same weights. At each step, the agent encounters the current state, selects an action from the main actor network, or policy, and adds noise $\mathcal{N}$ for exploration. When an action is taken, the transition $(s, a, r, s', d)$ is stored in $\mathcal{D}$. Afterwards a random minibatch of transitions $B$ is sampled from $\mathcal{D}$ to make network updates. The goal of these updates is to minimise the critic's loss with respect to the target, and maximise the actors gain with respect to the critic. The critic estimates the Q-value of taking an action in a state, and each update aims to minimise the difference between the estimated Q-value and target Q-value. The target can be described by

$$y(r, s', d) = r + \gamma(1 - d)\, Q_{\Phi_{targ}}(s', \mu_{\theta_{targ}}(s')) \tag{13}$$

The critic's loss function is defined as the mean-squared Bellman error (MBSE), and tells us how close $Q_\Phi$ is to approximating the bellman equation:

$$L(\Phi, \mathcal{D}) = E_{(s,a,r,s',d)\sim\mathcal{D}}[(Q_\Phi(s, a) - y(r, s', d))^2] \tag{14}$$

In order to learn a deterministic policy, the algorithm uses gradient ascent on the actor's parameters to maximise the expected Q-value determined by the critic:

$$\max_\theta E_{s\sim\mathcal{D}}[Q_\Phi(s, \mu_\theta(s))] \tag{15}$$

So, we have two updates, one for the critic and one for the actor. After performing gradient descent and ascent for each one, respectively, the target networks are updated. In order to stabilise the training, the target networks do not directly copy the weights of the main networks. Instead, they are slightly delayed

through soft updates using polyak averaging. For details of the training process of other algorithms, refer to SB3's guide on SAC [31] and TD3 [32].

**Algorithm 1** Deep Deterministic Policy Gradient

Input: initial policy parameters $\theta$, Q-function parameters $\phi$, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ}} \leftarrow \phi$
**repeat**
  Observe state $s$ and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
  Execute $a$ in the environment
  Observe next state $s'$, reward $r$, and done signal $d$ to indicate whether $s'$ is terminal
  Store $(s, a, r, s', d)$ in replay buffer $\mathcal{D}$
  If $s'$ is terminal, reset environment state.
  **if** it's time to update **then**
    **for** however many updates **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets

$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

      Update Q-function by one step of gradient descent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_\phi(s, a) - y(r, s', d)\right)^2$$

      Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \mu_\theta(s))$$

      Update target networks with

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1 - \rho)\phi$$
$$\theta_{\text{targ}} \leftarrow \rho\theta_{\text{targ}} + (1 - \rho)\theta$$

    **end for**
  **end if**
**until** convergence

***Algorithm 1:*** *Open AI Stable Baselines 3 implementation of DDPG* [19]

We optimise the training over different values for the learning rate ($\alpha$), $\gamma$, $Q_\Phi$, $\mu_\theta$, and $\mathcal{N}$. To track the results of different experimental setups and plot the model's training curves, we use the open-source library Weights & Biases the mean reward per episode, mean episode length, actor loss, and critic loss over all timesteps. Using these outputs, we get a general idea of which setups are optimal.

To test the agent's learning, we initialise a test instance of $BatteryEnv$ with an initial state:

$$SoC_t,\ C_1^D, C_1^C = 0$$

to emulate a real battery's initial state. The power and energy capacities for the test environment can be changed to represent different sized batteries, despite the model being with different parameters. For each row of the test data, we make a prediction from the model and take a step to the next state, and we log the state variables. By appending these values to each row of the test data, we can generate the same outputs as we did for the real BESS.

### *iii. Assumptions*

Due to this project's time constraints, we made several assumptions to simplify the battery model. We assume no cycling inefficiencies, allowing the agent to discharge 100% of the energy it charges. We

assume the battery's power level is constant through each SP, meaning that each PN of the battery agent is 30 minutes in length. We also assume no battery degradation. Despite these assumptions, we assert that the battery operates within its power and energy capacity limits, and that it charges and discharges at most 2 cycles daily.

## V. Results and Discussions

### *i. DRL Battery Agent Training*

Figure 3 shows the training plots of each algorithm's best setup. DDPG converged to an optimal episode length and reward, and its actor and critic loss converged much faster and to a lower value than the other algorithms. It is possible that TD3 required more samples to learn an optimal policy, however the data we had was limited. SAC also performs well and achieves carbon abatement values slightly below those of DDPG, but tends to end episodes early by exceeding constraints.

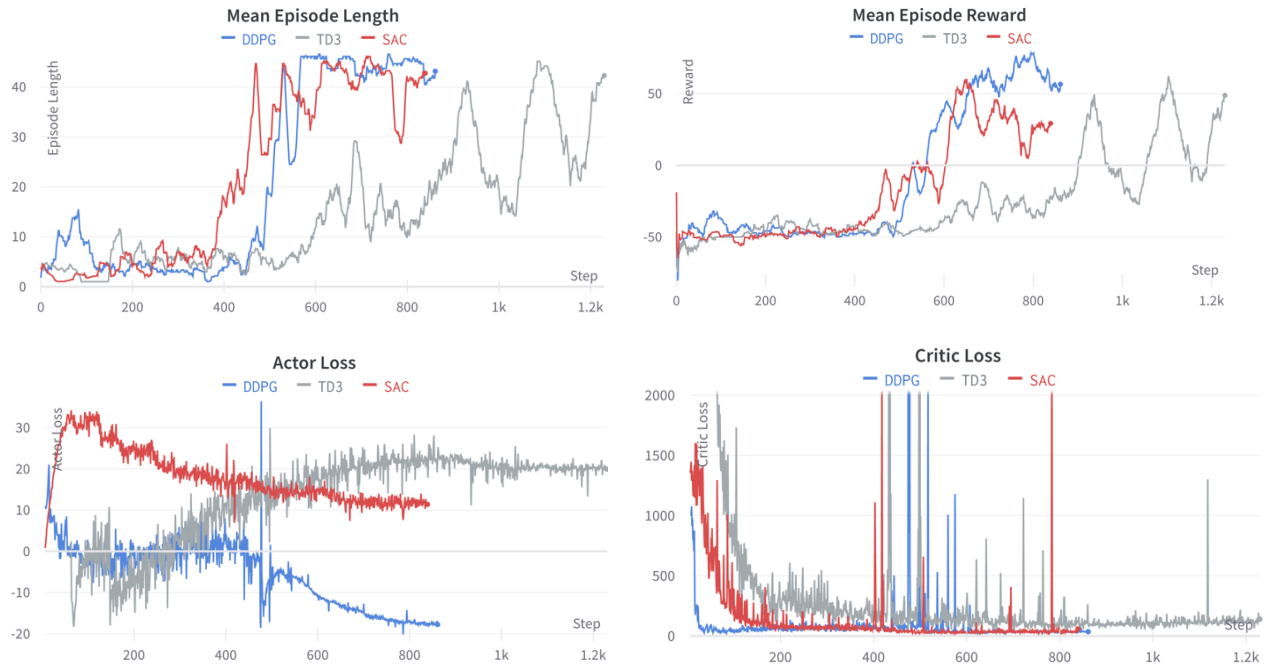



***Figure 3:*** *DRL Training Plots: Mean Episode Length, Mean Episode Reward, Actor Loss, Critic Loss*

Table 1 shows the optimal penalty value, forecast window, and DDPG hyperparameters. Penalty values less than 50 would lead the agent's policy to converge to ending the episode early and accepting the penalty. Larger values would prevent the agent from getting close to the boundaries. A window size of 24 gives forecasts for the next 12 hours, providing a good balance between seeking immediate and long-term rewards. A smaller $\gamma$ allowed the agent to focus more on the immediate rewards of charging or discharging in its current state. Higher $\gamma$ values led the agent to focus too much on the terminal reward, resulting in a less optimal charging strategy. The actor and critic both have the same ANN architecture suggested by [28], and no action noise was added as it prevented the agent from learning the charging boundaries. If the environment and state space are made more complex, action noise may become helpful. Other notable setups and results are described in A1.

***Table 1:*** *Optimal Battery Agent Environment and Model Parameters*

| Parameter | Description | Value |
|---|---|---|
| $penalty$ | Scalar penalty for exceeding constraints | 50 |
| $n$ | Window size for getting forecast features | 24 |
| $\gamma$ | Expected return discount rate | 0.91 |
| $\mathcal{N}$ | Action noise (exploration rate) | none |
| $Q_\Phi$ | Critic architecture: *[hidden layers], activation* | [400, 300], ReLU activation |
| $\mu_\theta$ | Actor architecture: *[hidden layers], activation* | [400, 300], ReLU activation |
| $\alpha$ | Learning rate | 0.001 |

Figure 4 shows tracking of $SoC_t$, $C_t^D$, and $C_t^C$ while testing the agent on the last year of CI data. The agent only charged and discharged an average of about 0.7 cycles per day, but it consistently completed episodes without a penalty. Finding the balance between managing constraints and maximising the reward was complicated, and given the time we had, this was the best solution. Other combinations of different reward structures and hyperparameters led the agent to cycle more every day, reaching a maximum average of about 1.6 cycles per day, but these setups were not charging at the optimal times.

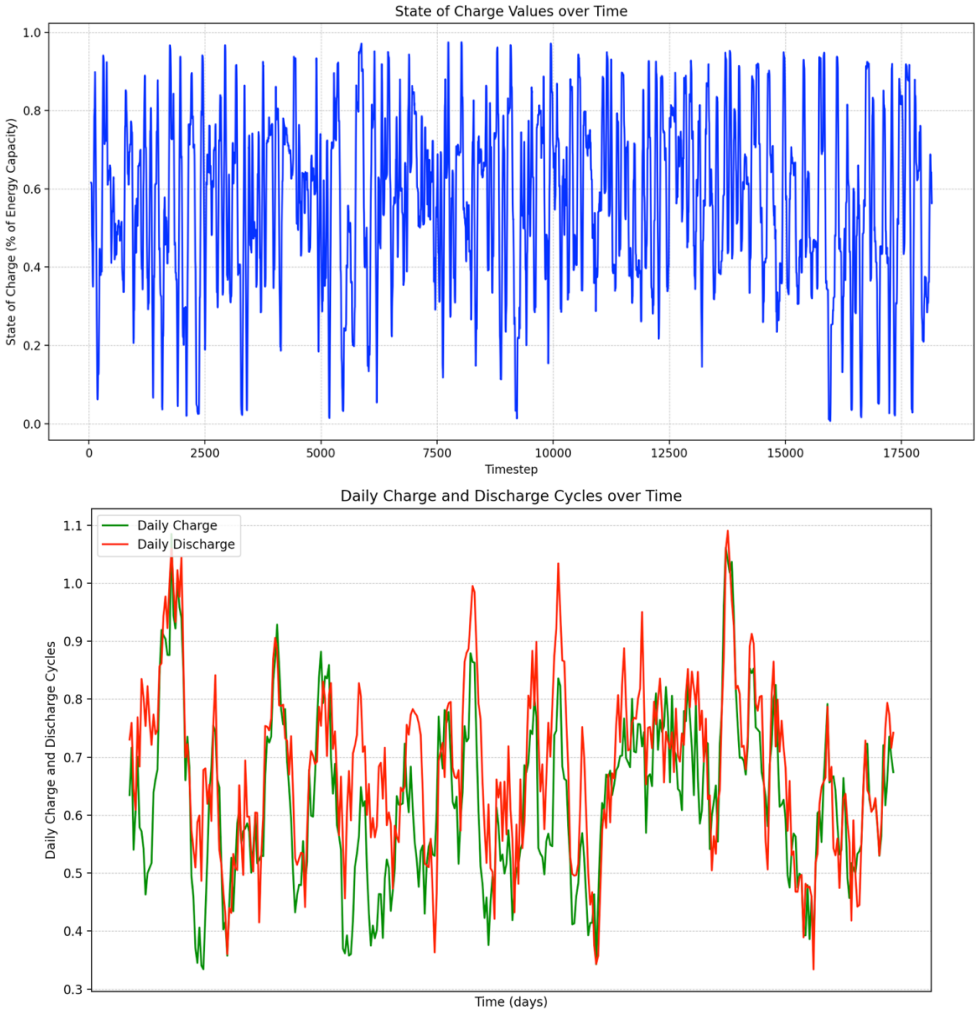


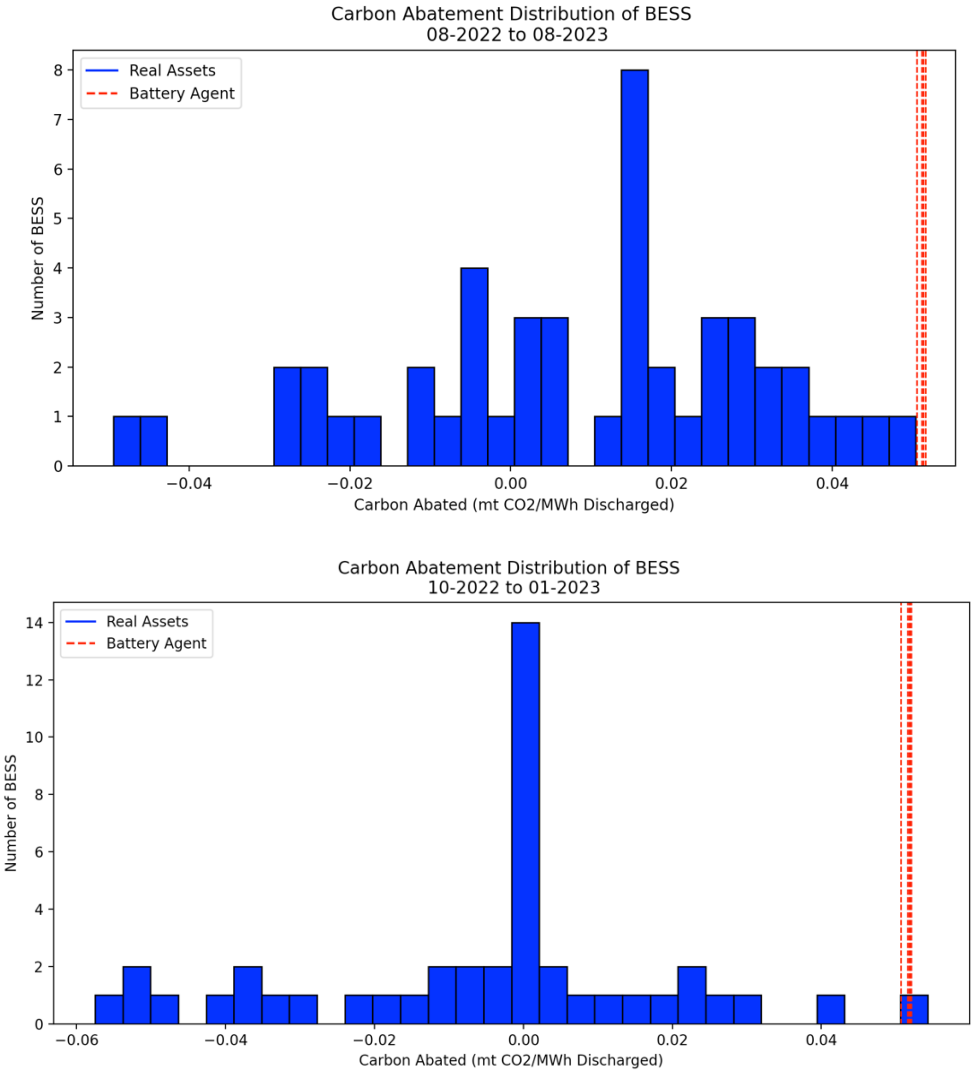

***Figure 4:*** *Battery Agent constraint values during testing. (a) Battery Agent state of charge during test run. (b) Batter Agent daily cycles charged and discharged per day of test run*

Training the agent did not require any special hardware requirements. Training 74000 timesteps took less than 7 minutes on a M1 Mac, and 41 minutes using the default Google Colab CPU runtime.

### *ii. Carbon Abated by Real BESS and Battery Agent*

In this section, we compare the results of the agent and those of real UK BESS. We set up 6 test agents with the same power and energy capacity as some prominent UK BESS: Bloxwich, Capenhurst 1, Creyke Beck, Holes Bay, Pillswood 1, and Red Scar. These assets were chosen due to their diverse durations, locations, and sizes, and their relevance in the UK.

To get an overall picture of BESS performance, we first look at the carbon abated by all BESS in the UK with a duration of 1 hour or greater and compare the results with those of our 6 agents. We use (1) to calculate the carbon abated for each asset, but because the assets are of different sizes, and have been operational for different amounts of time, we divide the result by $E_D$ for each asset. The resulting value is in metric tons of carbon abated per MWh discharged. In figure 5, we see the distribution of the carbon abated by these assets and the agents over different test periods. The generation mix of the grid changes with seasons, as renewable output fluctuates, so we ran the analysis for Q4 2022 and Q2 2023 and for the last year. The agent outperformed the real assets in all time frames.



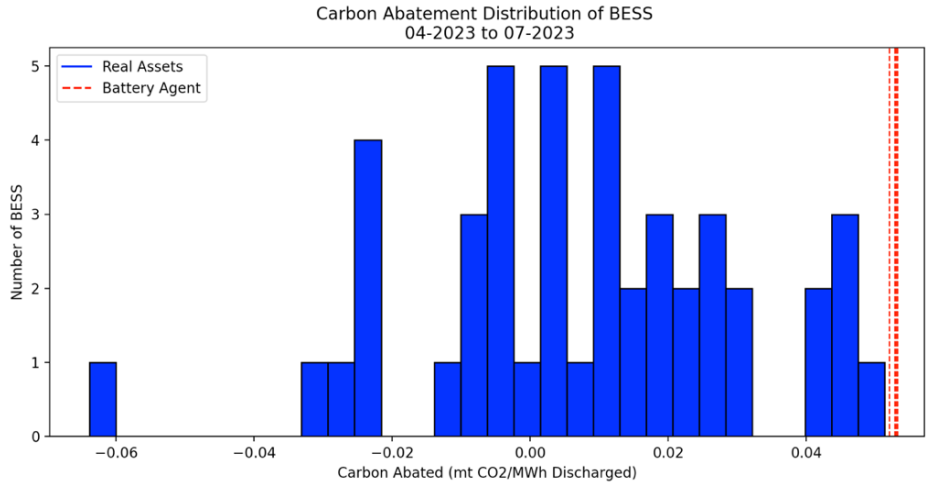



***Figure 5**: Carbon abated by all UK BESS and the Battery Agents in (a) the last year, (b) Q4 2022, and (c) Q2 2023*

Because longer duration batteries participate more in energy arbitrage and less in AS, we also analysed the carbon abated versus the duration of the batteries, as shown in figure 6. The agent outperformed batteries with all different durations. It offset about the same amount of carbon regardless of its duration.

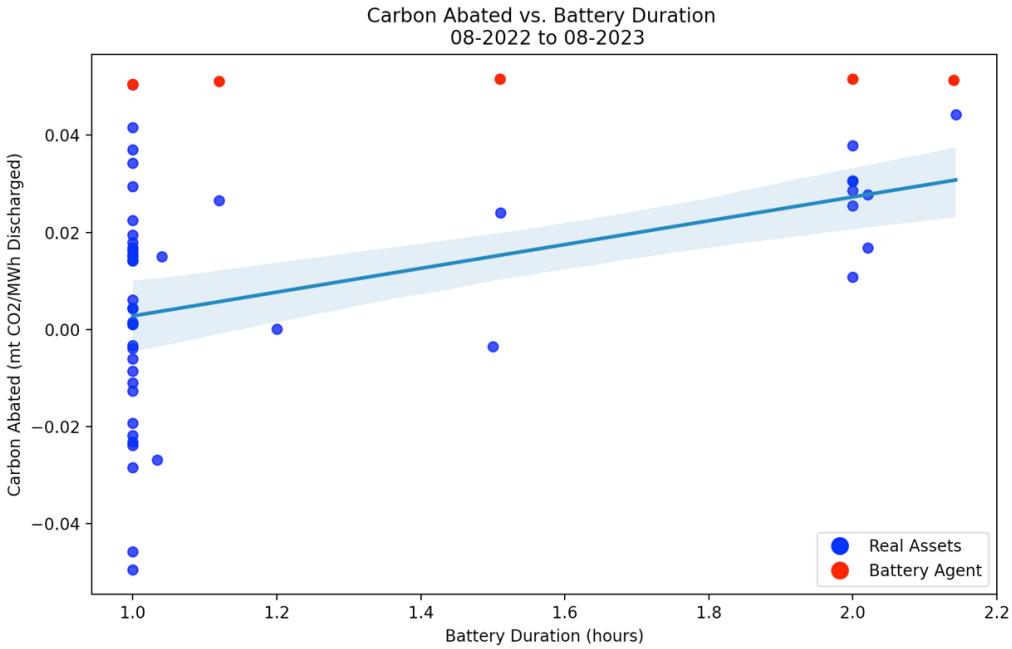



***Figure 6:** Carbon Abated by all UK BESS and the Battery Agents in the last year vs. their durations*

We decided to take a closer look at the results of the agents and the assets they are emulating. Some other assets were analysed, such as a shorter duration battery like Tynemouth. We could see that this asset contributed carbon in the last year, however considering it was a short duration battery, it is possible most of its activity was in AS, and not in the PN data. See the results of Tynemouth in A3 to see how its energy output in the wholesale market does not align optimally with the grid's CI.

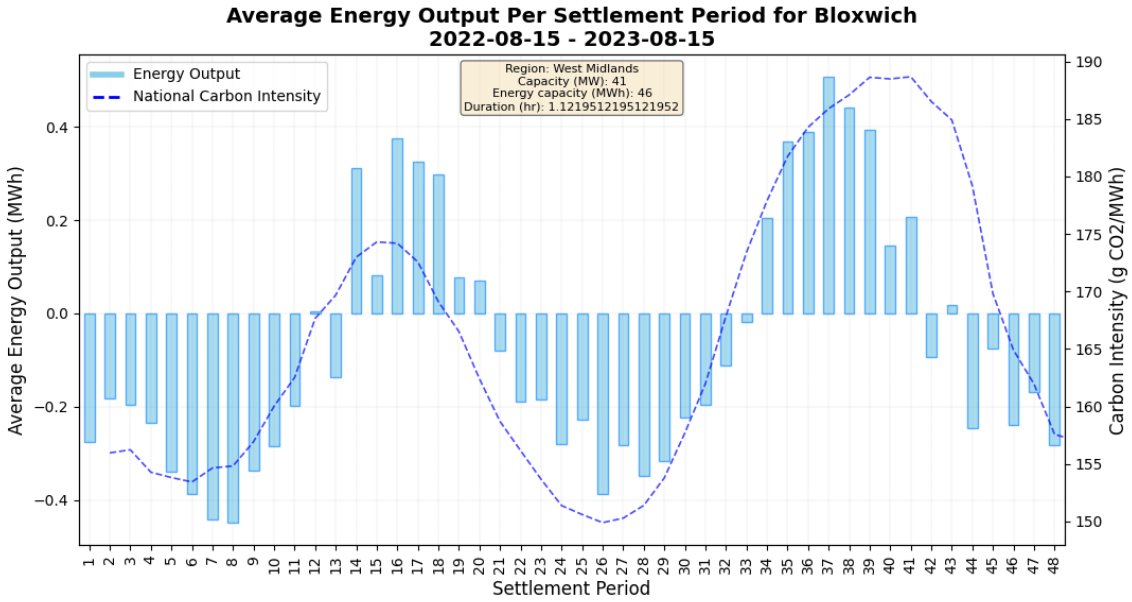
Table 2 shows the results of the 6 assets analysed, along with their corresponding agents. The agents' results are shown in the green columns. We see that the agent abated more carbon per MWh discharged

compared to each asset, mainly because the agent learned to maximise the difference between $CI_C$ and $CI_D$, resulting in an immediate positive reward. Additionally, the agent maintained a very minimal positive difference in the energy charged and discharged, reflecting realistic behaviour.

***Table 2**: Real BESS vs Battery Agent Carbon Abatement results from 15-08-2022 to 15-08-2023*

| Asset | Duration (h) | $CI_D - CI_C$ $(\frac{mt\ CO2}{MWh})$ | $E_C - E_D$ $(MWh)$ | Carbon Abated $(\frac{mt\ CO2}{MWh\ Discharged})$ | $CI_D - CI_C$ $(\frac{mt\ CO2}{MWh})$ | $E_C - E_D$ $(MWh)$ | Carbon Abated $(\frac{mt\ CO2}{MWh\ Discharged})$ |
|---|---|---|---|---|---|---|---|
| **Bloxwich** | 1.12 | 0.02798 | 4038.52 | -0.06317 | 0.05134 | 27.26 | 0.05111 |
| **Capenhurst 1** | 1 | 0.02286 | 301.84 | 0.01534 | 0.05085 | 24.34 | 0.05053 |
| **Pillswood 1** | 2 | 0.02801 | -266.68 | 0.03061 | 0.05202 | 59.88 | 0.05158 |
| **Holes Bay** | 2.14 | 0.03862 | -103.28 | 0.04421 | 0.05178 | 9.14 | 0.05131 |
| **Red Scar** | 1.51 | 0.03346 | 1114.35 | 0.02407 | 0.05194 | 48.17 | 0.05160 |
| **Creyke Beck** | 1 | 0.02055 | 2337.43 | 0.00439 | 0.05085 | 46.81 | 0.05053 |

We will take a closer look at the performance of the real Bloxwich asset and the corresponding agent. Figure 7 shows the average energy output per SP over the last year for Bloxwich and the agent. The profile of the other assets can be seen in A3. We can see that the agent waited to charge and discharge most of its energy at the peak demand hours of the day, when CI reached its half-daily maximum and minimum. The agent also charged and discharged a lot more energy overall. In the last year, the agent offset about 863 metric tons of carbon, while Bloxwich contributed about 444 metric tons of carbon. The agent's main strategy, driven by the reward structure in the environment, was to charge and discharge as little as possible until we reach the 12-hour peaks of CI, where most of the charging and discharging took place. We can see from the plot that the agent still tended to act prematurely, but in the peak hours from SPs 38 to 42, the agent sustained its positive power level and discharged more than Bloxwich. The reward function and forecast features may require more careful tuning to get the timing just right.



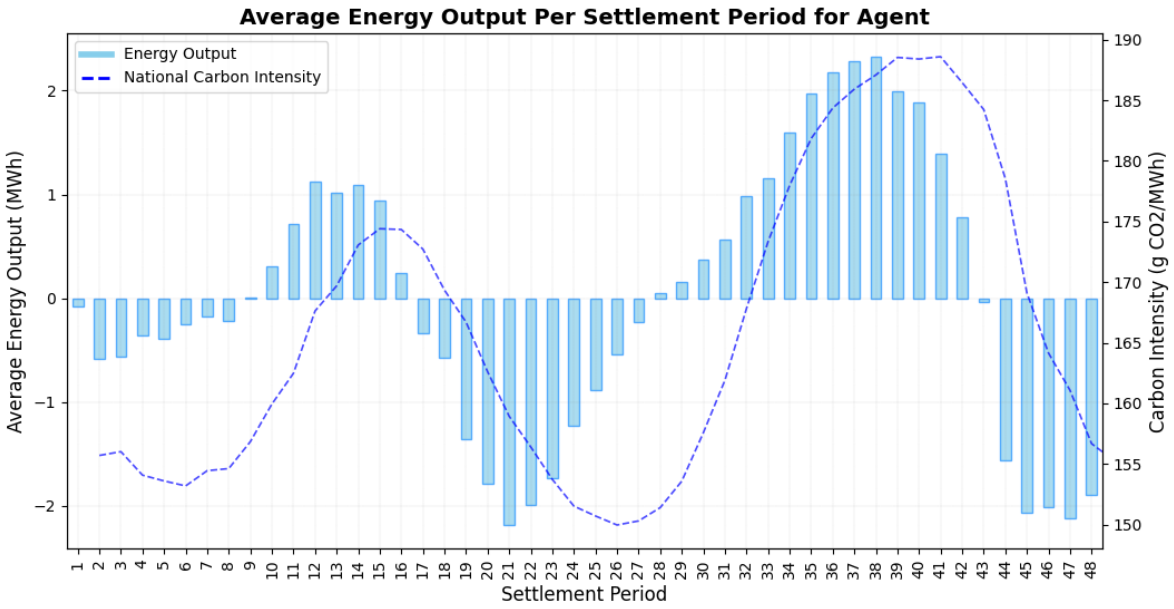



***Figure 7:*** *Energy output profiles in the last year. (a) Average energy output per SP by Bloxwich (b) Average energy output per SP by Battery Agent*

## VI. Conclusion

In this paper, we introduced a methodology to calculate the carbon abated by BESS in UK wholesale markets and developed a DRL solution to optimise a battery's trading, or charging schedule, to offset as much carbon as possible. Our findings indicate that while many BESS contribute carbon to the atmosphere, longer duration batteries show a trend of increased carbon abatement. Our DRL agent, trained using the DDPG algorithm, demonstrates the potential to increase carbon abatement of BESS in the UK and decrease overall emissions. The results, run on the last year of carbon intensity data, show that the agent offsets at least 0.05 metric tons of CO2 per MWh discharged, compared to the real-world average of about 0.0 to 0.02 metric tons depending on the time frame. The agent operated rather conservatively given its constraints, and still charged and discharged more energy than real assets.

While this study demonstrates potential for reducing carbon emissions, it also highlights areas for future research. A more complex and realistic representation of a battery storage system must be developed, covering characteristics like round trip inefficiency, battery degradation, and market operations. As the models become more complex, more training data may be required, and other algorithms, reward structures, and environments may be better suited. TD3, for example, is an adaptation of DDPG generally suited for more complex problems.

As BESS optimisation technology continues to advance, the deployment of a carbon abatement optimisation algorithms could substantially accelerate the transition to a net-zero electricity system. This work lays the foundation for more sustainable energy storage solutions and offers a promising opportunity for reducing carbon emissions.

**References**


1. Future Energy Scenarios 2023 Report, National Grid ESO (2023). https://www.nationalgrideso.com/future-energy/future-energy-scenarios
2. MODO Energy Academy https://platform.modo.energy/phase/channels?wchannelid=wy5tng1x36
3. D. Duncan, “GB batteries confront ancillary saturation,” Timera Energy, Oct. 31, 2022. https://timera-energy.com/gb-batteries-confront-ancillary-saturation/
4. D. Duncan, “Battery storage duration is lengthening,” Timera Energy, Dec. 06, 2021. https://timera-energy.com/battery-storage-duration-is-lengthening/
5. The Carbon Benefit of Battery Energy Storage in Great Britain. MODO Energy (2023). https://platform.modo.energy/phase/article/8973/carbon-benefit-battery-energy-storage-avoided-co2-emissions
6. A. Mey, “Utility-scale batteries and pumped storage return about 80% of the electricity they store - Today in Energy - U.S. Energy Information Administration (EIA),” www.eia.gov, Feb. 12, 2021. https://www.eia.gov/todayinenergy/detail.php?id=46756]
7. D. Schweer, A. Maaz and A. Moser, "Optimization of frequency containment reserve provision in M5BAT hybrid battery storage," 2016 13th International Conference on the European Energy Market (EEM), Porto, Portugal, 2016, pp. 1-5, doi: 10.1109/EEM.2016.7521335.
8. J. Cao, D. Harrold, Z. Fan, T. Morstyn, D. Healey and K. Li, "Deep Reinforcement Learning-Based Energy Storage Arbitrage With Accurate Lithium-Ion Battery Degradation Model," in IEEE Transactions on Smart Grid, vol. 11, no. 5, pp. 4513-4521, Sept. 2020, doi: 10.1109/TSG.2020.2986333.
9. G. He, Q. Chen, C. Kang, P. Pinson and Q. Xia, "Optimal Bidding Strategy of Battery Storage in Power Markets Considering Performance-Based Regulation and Battery Cycle Life," in IEEE Transactions on Smart Grid, vol. 7, no. 5, pp. 2359-2367, Sept. 2016, doi: 10.1109/TSG.2015.2424314.
10. H. Aaltonen, S. Sierla, V. Kyrki, M. Pourakbari-Kasmaei, and V. Vyatkin. Bidding a Battery on Electricity Markets and Minimizing Battery Aging Costs: A Reinforcement Learning Approach. Energies. 2022; 15(14):4960. https://doi.org/10.3390/en15144960
11. K. Volkan, H. Holger, S. Michael, T. Anshuman, W. Youyi, and J. Andreas. Energy Arbitrage Optimization With Battery Storage: 3D-MILP for Electro-Thermal Performance and Semi-Empirical Aging Models. IEEE Access. 8. (2020).
12. O. Salman, M. Kampouridis, and D. Jarchi, “Trading Strategies Optimization by Genetic Algorithm under the Directional Changes Paradigm,” IEEE Xplore, Jul. 01, 2022. https://ieeexplore.ieee.org/abstract/document/9870270
13. D. Magnor and D. U. Sauer, “Optimization of PV Battery Systems Using Genetic Algorithms,” Energy Procedia, vol. 99, pp. 332–340, Nov. 2016, doi: https://doi.org/10.1016/j.egypro.2016.10.123
14. S. Karakatič, “Optimizing nonlinear charging times of electric vehicle routing with genetic algorithm,” Expert Systems with Applications, vol. 164, p. 114039, Feb. 2021, doi: https://doi.org/10.1016/j.eswa.2020.114039
15. S. Zhang and O. R. Zaiane, “Comparing Deep Reinforcement Learning and Evolutionary Methods in Continuous Control,” arXiv.org, Mar. 07, 2018. https://arxiv.org/abs/1712.00006
16. Y. Dong, Z. Dong, T. Zhao, and Z. Ding. A Strategic Day-ahead Bidding Strategy And Operation For Battery Energy Storage System By Reinforcement Learning. Electric Power Systems Research. Jul. 2021, doi: https://doi.org/10.1016/j.epsr.2021.107229
17. Z. Wan, H. Li, H. He, and D. Prokhorov, “Model-Free Real-Time EV Charging Scheduling Based on Deep Reinforcement Learning,” IEEE Transactions on Smart Grid, vol. 10, no. 5, pp. 5246–5257, Sep. 2019, doi: https://doi.org/10.1109/tsg.2018.2879572.

18. O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the Gap Between Value and Policy Based Reinforcement Learning," arXiv:1702.08892 [cs.AI], Nov. 2017, https://arxiv.org/abs/1702.08892
19. "Deep Deterministic Policy Gradient — Spinning Up documentation," Openai.com. https://spinningup.openai.com/en/latest/algorithms/ddpg.html
20. F. Chang, T. Chen, W. Su, and Q. Alsafasfeh, "Control of battery charging based on reinforcement learning and long short-term memory networks," Computers & Electrical Engineering, vol. 85, p. 106670, Jul. 2020, doi: https://doi.org/10.1016/j.compeleceng.2020.106670
21. Y. Liu, Q. Liu, H. Zhao, Z. Pan, and C. Liu, "Adaptive Quantitative Trading: An Imitative Deep Reinforcement Learning Approach," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 02, pp. 2128–2135, Apr. 2020, doi: https://doi.org/10.1609/aaai.v34i02.5587
22. H. Shengren, E. M. Salazar, P. P. Vergara, and P. Palensky, "Performance Comparison of Deep RL Algorithms for Energy Systems Optimal Scheduling," arXiv.org, Aug. 01, 2022. https://arxiv.org/abs/2208.00728
23. X. Chen, G. Qu, Y. Tang, S. Low, and N. Li. "Reinforcement Learning for Decision-Making and Control in Power Systems: Tutorial, Review, and Vision," DeepAI, Jan. 27, 2021. https://deepai.org/publication/reinforcement-learning-for-decision-making-and-control-in-power-systems-tutorial-review-and-vision
24. V. Konda and J. Tsitsiklis, "Actor-Critic Algorithms," Apr. 2001. Accessed: Aug. 21, 2023. https://www.researchgate.net/publication/2354219_Actor-Critic_Algorithms
25. V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," arXiv.org, Dec. 19, 2013. https://arxiv.org/abs/1312.5602
26. T. P. Lillicrap et al., "Continuous Control with Deep Reinforcement Learning," arXiv:1509.02971 [cs.LG], Jul. 05, 2019. https://arxiv.org/abs/1509.02971
27. T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," arXiv:1801.01290 [cs.LG], Aug. 2018. https://arxiv.org/abs/1801.01290
28. S. Fujimoto, van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," arXiv:1802.09477 [cs.AI], Oct. 22, 2018. https://arxiv.org/abs/1802.09477
29. "Carbon Intensity API," Carbonintensity.org.uk. https://carbonintensity.org.uk/
30. Elexon BSC Insights Solution," Elexon.co.uk. https://bmrs.elexon.co.uk/api-documentation/endpoint/datasets/PN/stream
31. "Soft Actor-Critic — Spinning Up documentation," Openai.com. https://spinningup.openai.com/en/latest/algorithms/sac.html
32. "Twin Delayed DDPG — Spinning Up documentation," Openai.com. https://spinningup.openai.com/en/latest/algorithms/td3.html
33. T. Bowen, P. Denholm, and I. Chernyakhovskiy, "Grid-Scale Battery Storage Frequently Asked Questions," Sep. 2019. https://greeningthegrid.org/news/new-resource-grid-scale-battery-storage-frequently-asked-questions-1

**Appendix**

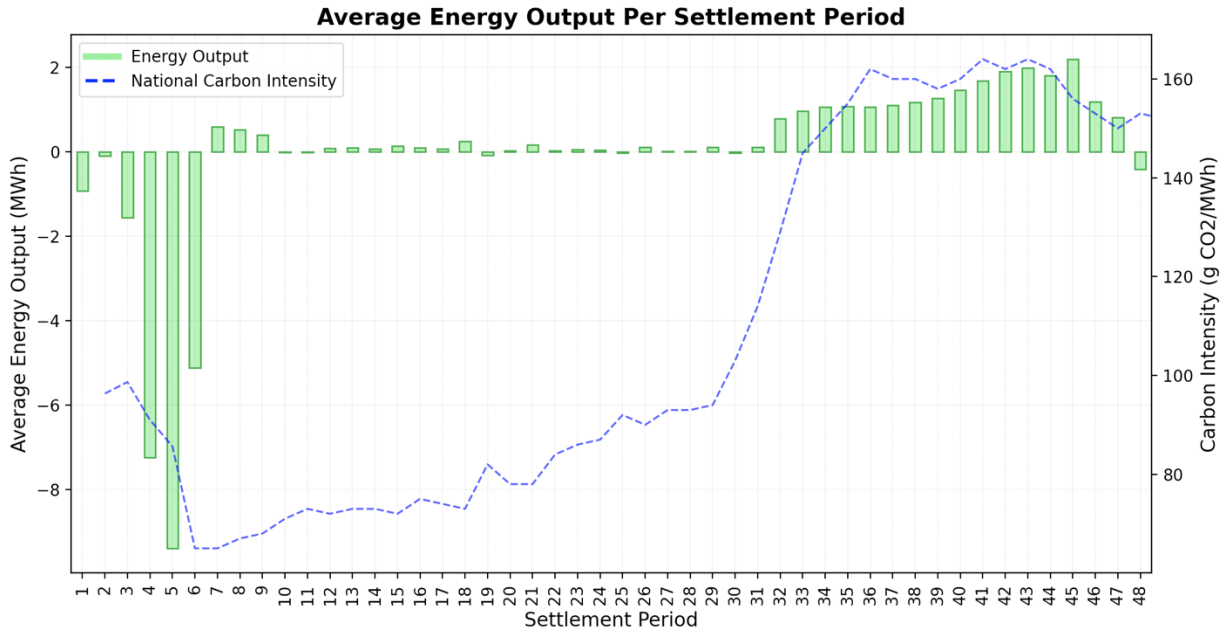
## A1. Notable Test Setups and Results

All the experiments are shown in https://wandb.ai/irp-cs1622/projects , but the following are the most notable results from each algorithm.

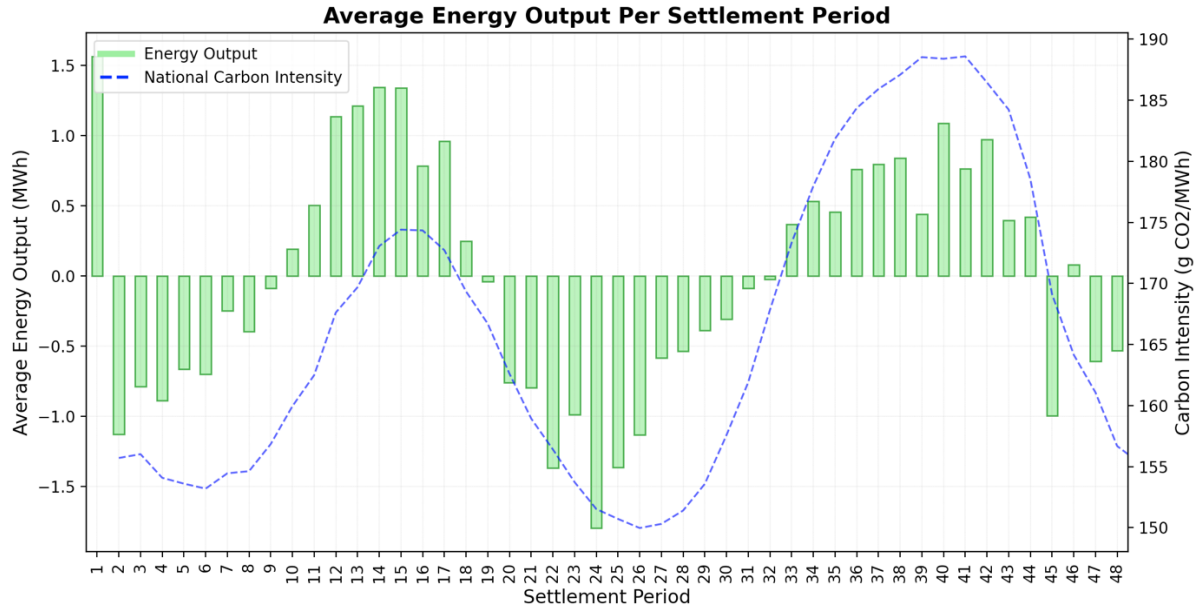
| Algorithm | Parameters changed from default values | Carbon Abated ($\frac{mt\ CO2}{MWh\ Discharged}$) | Comments |
|---|---|---|---|
| SAC_1 | $\gamma$= 0.91<br>$\alpha = 0.0003$ | 0.04996 | Good result, but episode lengths tend to be shorter than DDPG |
| SAC_2 | $\gamma$= 0.92<br>$\alpha = 0.0003$ | 0.04863 | |
| SAC_3 | $\gamma$= 0.99<br>$\alpha = 0.0003$ | 0.0347 | |
| TD3_1 | $\gamma$= 0.91 | 0.03310 | |
| TD3_2 | $\gamma$= 0.99 | 0.03521 | |
| DDPG_1 | $\gamma$= 0.91 | 0.05337 | Best result |
| DDPG_2 | $\gamma$= 0.91<br>learning_starts =1000 | 0.05212 | Modifying learning starts did not make much of a difference |
| DDPG_3 | $\gamma$= 0.91<br>$n$ = 12 | 0.01435 | Charged and discharged an average of about 1.4 cycles. Did this too early in the day and could not charge and discharge much in peak hours. See A2 for plots. |
| DDPG_4 | $\gamma$= 0.91<br>$n$ = 18 | 0.045661 | Charged and discharged about 40% more energy than other approaches, about 1.2 cycles on average. Same issues as DDPG_3. |
| DDPG_5 | $\gamma$= 0.99 | 0.00934 | 1.4 charge and discharge cycles on average, but a poor charging strategy. |

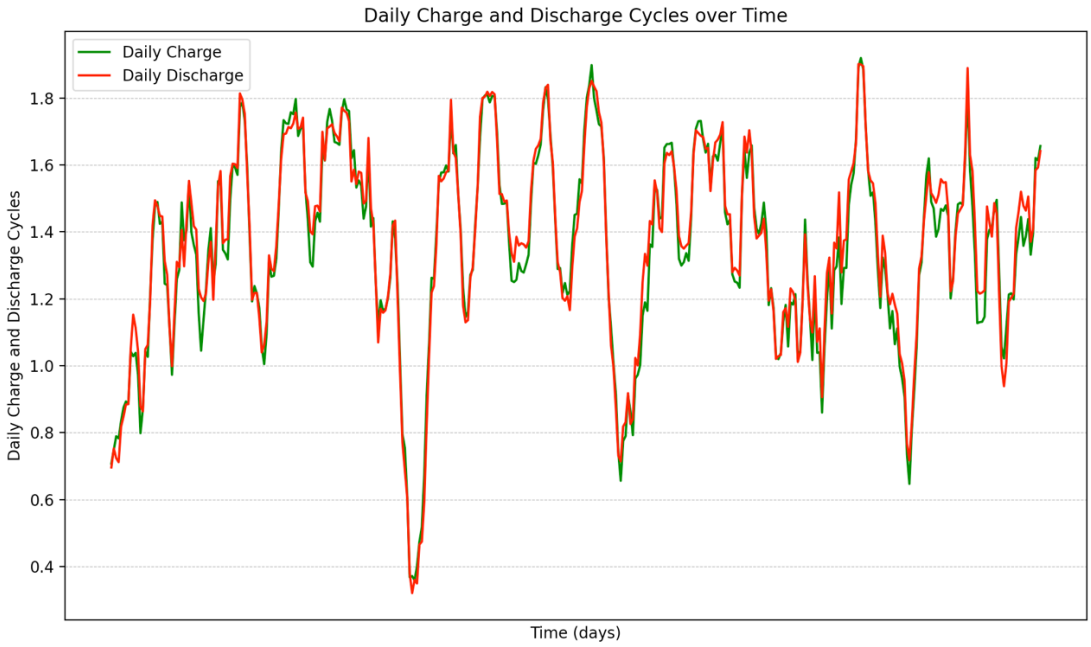
## A2. Notable Agent Test Plots

Best model run on 1 day of data: We can see that the agent, on a single day, discharges and charges almost perfectly at peak times.



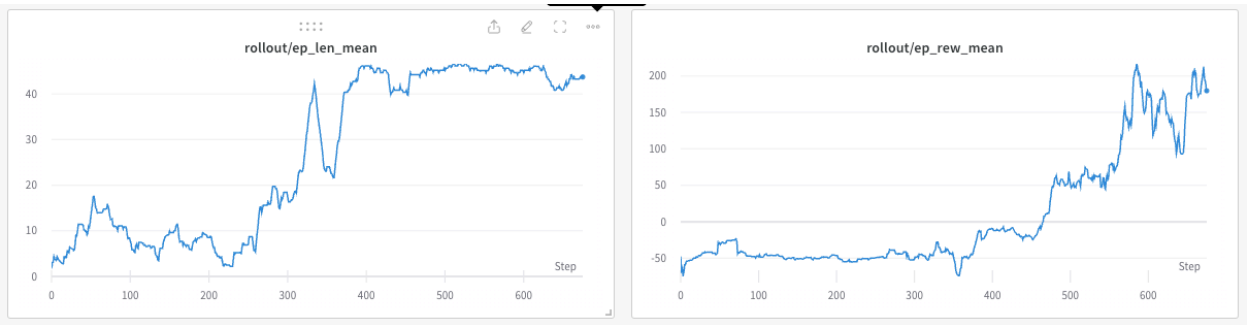

DDPG_3:





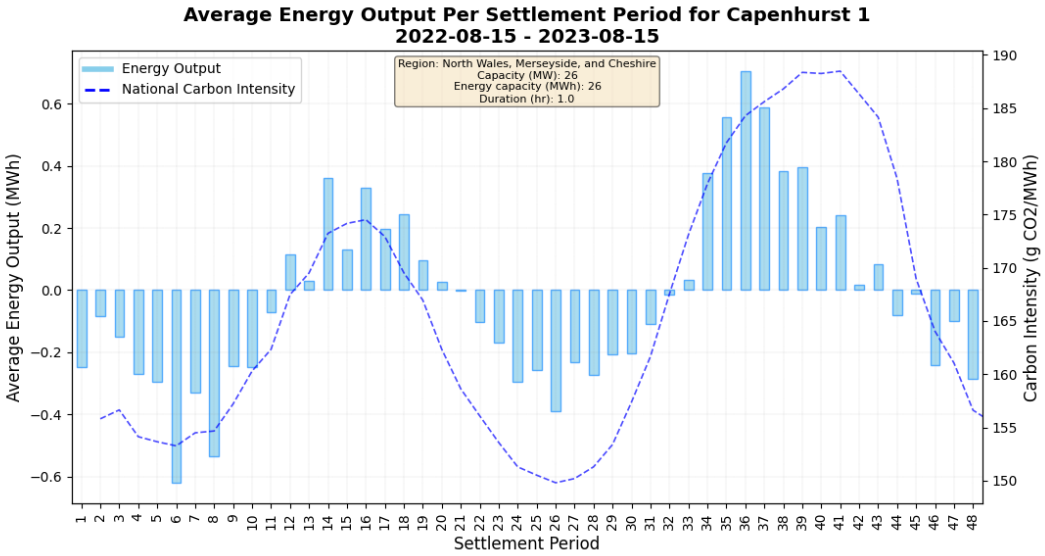

DDPG_5:

Gets a much higher reward because it focuses on getting the cycle reward at the end of the episode, but the charging schedule is not optimal.

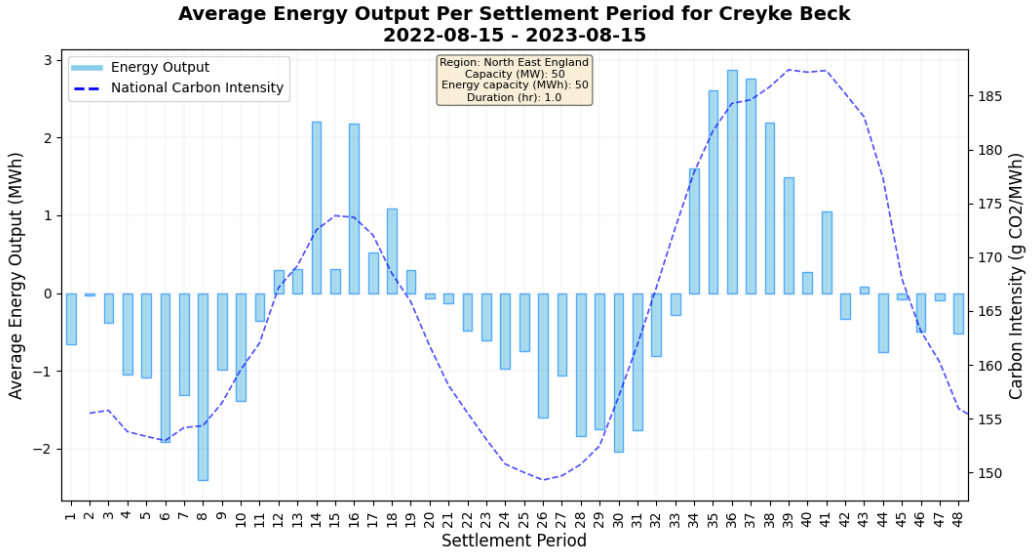


**A3. Energy output profiles of BESS Assets in the last year**

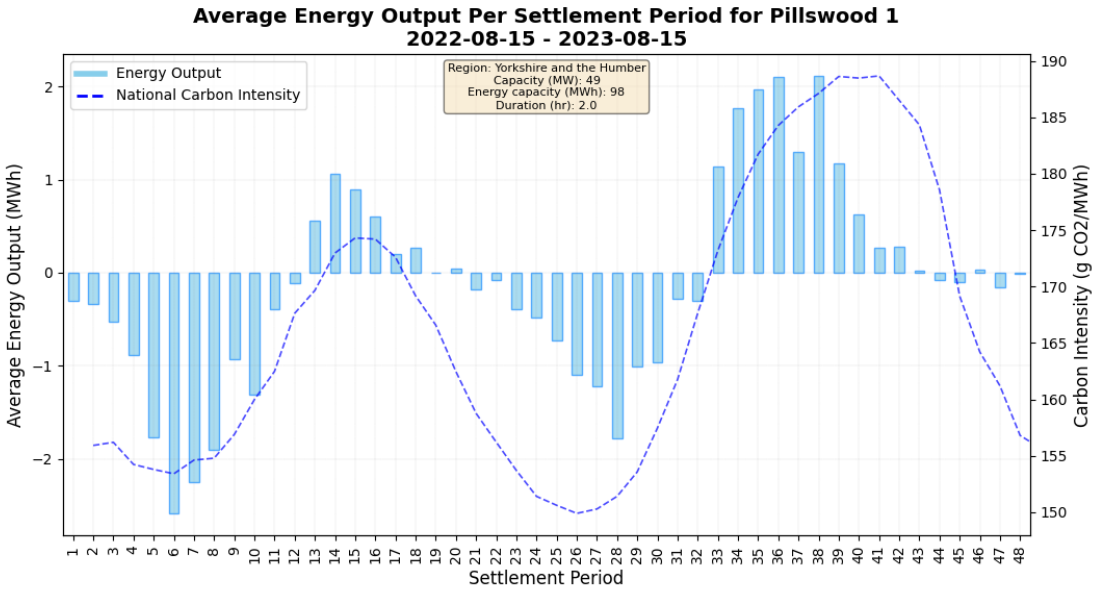
Capenhurst 1:



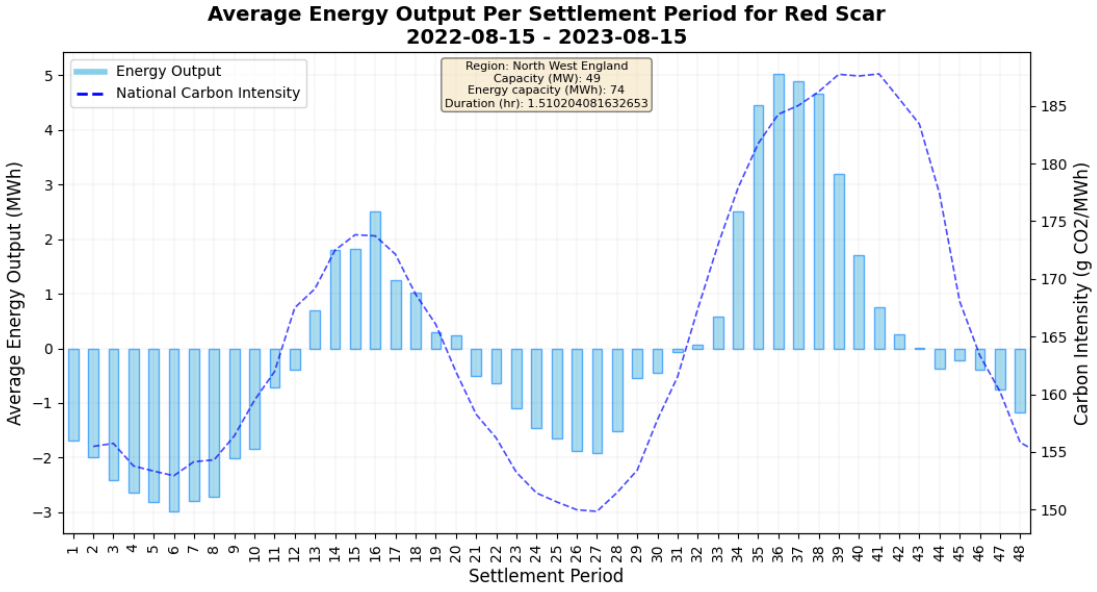

Creyke Beck:



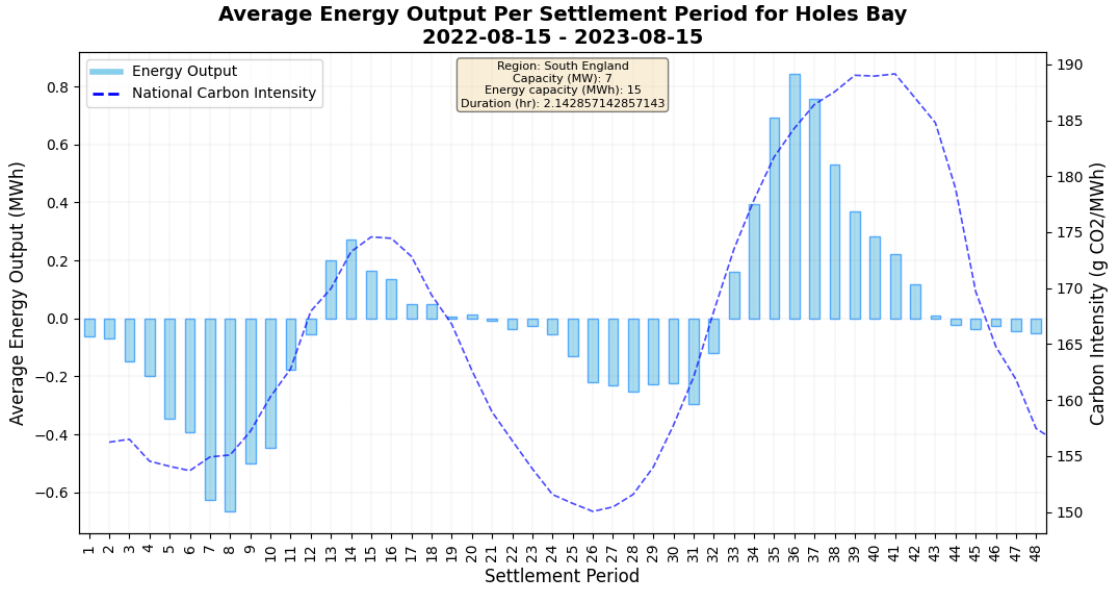
Pillswood 1:




Red Scar:



Holes Bay:




Tynemouth: