energies

MDPI

Article

Dual-Layer Q-Learning Strategy for Energy Management of Battery Storage in Grid-Connected Microgrids

Khawaja Haider Ali [1,2,*], Mohammad Abusara [1,*], Asif Ali Tahir [1] and Saptarshi Das [1]

[1] Faculty of Environment, Science and Economy, University of Exeter, Penryn Campus, Cornwall TR10 9FE, UK
[2] Department of Electrical Engineering, Sukkur IBA University, Sukkur 65200, Pakistan
* Correspondence: haiderali@iba-suk.edu.pk (K.H.A.); m.abusara@exeter.ac.uk (M.A.)

Abstract: Real-time energy management of battery storage in grid-connected microgrids can be very challenging due to the intermittent nature of renewable energy sources (RES), load variations, and variable grid tariffs. Two reinforcement learning (RL)–based energy management systems have been previously used, namely, offline and online methods. In offline RL, the agent learns the optimum policy using forecasted generation and load data. Once the convergence is achieved, battery commands are dispatched in real time. The performance of this strategy highly depends on the accuracy of the forecasted data. An agent in online RL learns the best policy by interacting with the system in real time using real data. Online RL deals better with the forecasted error but can take a longer time to converge. This paper proposes a novel dual layer *Q*-learning strategy to address this challenge. The first (upper) layer is conducted offline to produce directive commands for the battery system for a 24 h horizon. It uses forecasted data for generation and load. The second (lower) *Q*-learning-based layer refines these battery commands every 15 min by considering the changes happening in the RES and load demand in real time. This decreases the overall operating cost of the microgrid as compared with online RL by reducing the convergence time. The superiority of the proposed strategy (dual-layer RL) has been verified by simulation results after comparing it with individual offline and online RL algorithms.

Keywords: reinforcement learning (RL); microgrid; energy management; offline and online RL; dual-layer Q-learning

check for updates

Citation: Ali, K.H.; Abusara, M.; Tahir, A.A.; Das, S. Dual-Layer Q-Learning Strategy for Energy Management of Battery Storage in Grid-Connected Microgrids. *Energies* **2023**, *16*, 1334. https://doi.org/10.3390/en16031334

Academic Editor: Antonio Cano-Ortega

Received: 13 December 2022
Revised: 17 January 2023
Accepted: 19 January 2023
Published: 27 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, studies on microgrids (MGs) have focused on improving energy management systems (EMS) in diverse environments. The term microgrid refers to the decentralised group of renewable-energy-based electricity sources, energy storage system (ESS), and variable load [1]. A microgrid can operate in either the standalone or grid-connected mode exchanging power with the main grid. An effective EMS can achieve different goals, such as optimisation of operating cost, better utilisation of RES, demand-side management, reduction in the use of polluting-fossil-fuel-based power sources, or balancing energy demand [2]. Most of the efforts are intended to reduce the operating cost of the microgrid through the EMS controlling the ESS. However, this is challenging due to various unknown factors, which may change over time, such as RES production, load demand, and utility prices, which are strongly influenced by the weather conditions [3]. There is a lack of resolution for the nonlinearity and complexity of energy forecasts, including modelling growth and computational load, particularly when combining multilevel parameters' uncertainty [4]. In order to promote energy efficiency, precise energy forecasts based on simple models using machine learning schemes are promising. Moreover, high-level controllers with short-term building energy forecasts under high-level scenario uncertainties should be studied more deeply. Different methods have been used to manage ESS in microgrids, including linear programming (LP), nonlinear programming (NLP),

dynamic programming (DP), mixed integer linear programming (MILP), genetic algorithms (GA), and mixed integer nonlinear programming (MINLP). These approaches require a good forecasting system to build a model to achieve decent optimisation results. These approaches have the their own pros and cons. Linear programming (LP) is the best method of utilising productive resources because it gives feasible and practical solutions and improves decision making [5]. Nonlinear programming simplifies complex problems, but it is computationally expensive [6]. Dynamic programming reduces a complex problem into smaller parts, which can be solved more easily. However, this suffers from the curse of dimensionality when the system has many state variables [7]. Mixed integer linear programming offers a flexible set of subfunctions and intelligent convergence behaviour. However, to facilitate self-adoption, choosing the appropriate parameters is an essential step, especially under conditions of uncertainty due to weather or fluctuations in real-time load demand [8].

One concern with the above-mentioned classical model-based approaches is the increase in computational costs resulting from the addition of more information due to modifications in design, scale, and capacity [9]. Moreover, model-based methods solve multitime serial-decision problems at each time slot, causing a hindrance in real-time decisions [10]. In real time, the variation of information is not always accompanied by model formation using predicted data. To deal with the above limitations, some other advanced methods, such as fuzzy logic (FL), neural network (NN), and multiagent system (MAS), are used in the literature. The construction of an FL model is not complicated, although it requires precise information from the sources [11]. In [11], FL was applied to control the charge and discharge of the battery, which effectively reduced power consumption using reliable information of energy demand. An NN model was successfully implemented in [12] for the EMS of a microgrid. The NN approach has the ability to generalise and perform many calculations at once as a generalised approximator [12] and produce optimal results even with less input information, but this leads to a higher computational burden [13]. In the same way, function approximation methods (FAM) need to choose a proper function for approximation to achieve an optimal result. The MAS technique, on the other hand, is a decision-based approach from multiple agents working towards common or conflicting objectives. It provides resistant, robust, and quick solutions [14]. A MAS approach was applied in [15] to manage multiple renewable energy sources, such as PV, wind turbines, and fuel cells, to ensure a stable operation of the microgrid. However, not every scheduling problem may be solved using a MAS scheme because it needs to decompose the conditions for each individual agent. FL, NN, and MAS received high attention to solve scheduling problems related to microgrids to achieve cost-effective solutions [12]. Nevertheless, the high-quality optimisation result relies heavily on the forecasting accuracy, which is a challenge in real-time decision making due to uncertainties, such as abrupt change of weather or load demand.

The above EMS challenges are addressed by several machine learning techniques developed in the last few decades. Reinforcement learning (RL) is a type of machine learning algorithm that has gained high interest due to its potential to solve critical real-time optimisation in model-free environments. Different RL algorithms were used to optimise ESS in microgrids. In this regard, contemporary works such as [9,16,17] have applied Q-learning for battery scheduling. Q-learning is an RL algorithm that seeks to identify the best course of action based on the current situation. The training of a Q-learning agent can take place offline using forecasted profiles, such as PV and load demand. After training, it is quite possible that the applied actions in real time do not give optimal performance due to uncertainties happening in the real environment. The solution to deal with environmental uncertainties was presented in [18,19], which uses online RL methods to find the optimal control strategy for battery operation while interacting with the real system in real time. Our previous work in [20] has compared the performance of offline RL with that of online RL for managing ESS in microgrids. Synthetic forecasted data were constructed by adding white Gaussian noise with a range of standard deviations on real

data to approximate forecasted data. When the difference between the real and predicted data is greater than 1.6%, online RL produces better results than offline RL. However, online RL takes a relatively long time to converge during which the performance is suboptimal.

This paper proposes a new dual-layer Q-learning strategy to address this challenge. The first layer is conducted offline to produce directive commands for the battery system for a 24 h horizon. This layer uses forecasted data for generation and load. The second Q-learning-based layer uses the offline directive as part of its state space and refines the battery commands every 15 min by considering the changes happening in the RES and load demand in real time. This decreases the overall operating cost of the microgrid as compared with online RL by reducing the convergence time. The main contributions of this paper are summarised as:

The implementation of a hierarchical Q-learning-based EMS with 2 layers. The first (upper) layer makes use of forecasted data and produces 24 battery commands for the next 24 h. The second (lower) layer makes use of access to real data in real time and further tunes the offline battery commands every 15 min before applying them in real time.

The comparison of the proposed strategy against two known strategies: offline and online RL in addition to the ideal case with zero forecast error.

2. Microgrid Structure

The grid-connected microgrid considered in this paper is shown in Figure 1. It consists of PV panels, ESS, and load. The EMS is deployed to control the battery operation. There are three different grid tariff rates: high, medium, and low. It should be noted that there is only one feed in tariff. The scheduling of battery consists of different levels of battery actions, such as charge or discharge at different power levels or remain idle. The objective of the EMS is to minimise the cost of the grid imported power by planning the optimal operations of the battery. For example, battery is charged during the low peak tariff rates or when the generation of PV is greater than the load demand. Thus, the battery can be charged directly from the utility grid or PV source. Extra available PV after fulfilling the load demand can be injected into the main grid to increase revenue. It is also possible that PV, battery, and main grid collectively fulfil the load demand if PV and battery do not have enough power to meet the load demand.

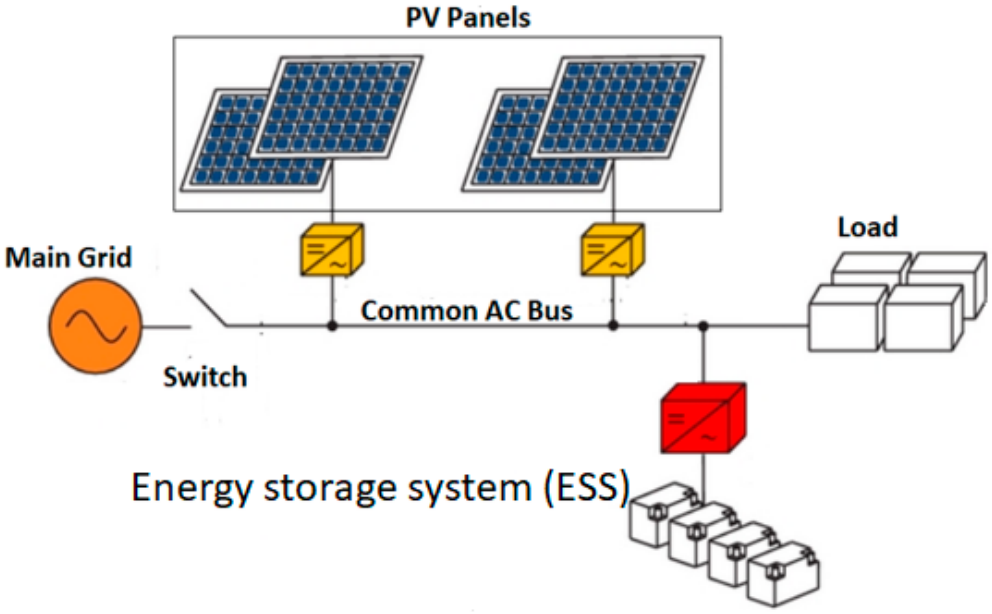


Figure 1. The grid-tied microgrid with an energy storage system.

3. The Proposed Two-Layer Strategy

Figure 2a shows the proposed two-layer strategy. The first (upper) offline layer uses predicted data profiles to train the RL agent. At the start of each day, the Q-table is initialised with all state action values set to zero. The forecasted PV and load data are collected as inputs to RL. The Q-learning algorithm is then run using the unchanged input data until convergence is attained. In the next 24 h, battery charging/discharging commands are generated using the policy established in this phase. Every day, this strategy is repeated.

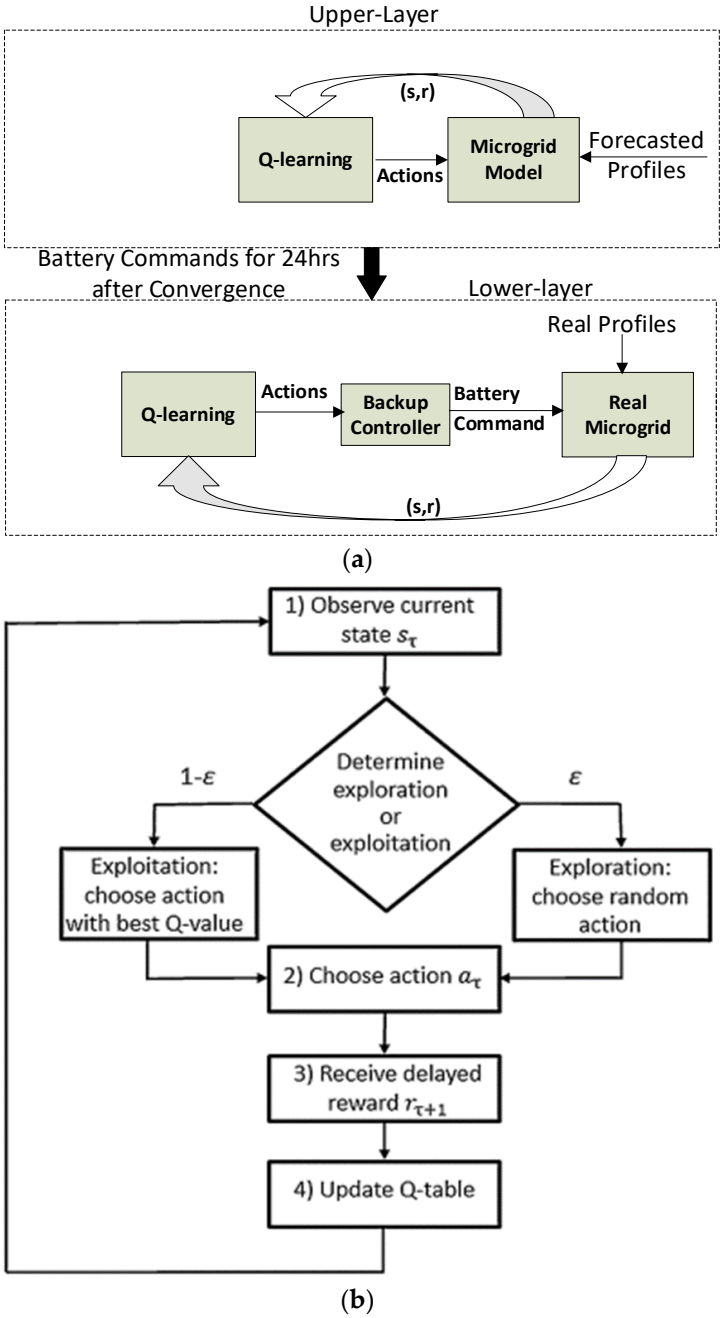


Figure 2. (**a**) Proposed 2-layer RL strategy using Q-learning with respective upper and lower layers. (**b**) Q-learning algorithm flowchart.

The recommended offline actions from the upper layer are passed to the second (lower layer) Q-learning, which are used as part of its state space. The lower-layer Q-learning updates the actions of the battery (charging, discharging, and idle) using knowledge of real-time data. It acts as a fine tuner for the battery actions based on the difference between

forecasted data and real data. The lower-layer *Q*-learning runs and dispatches modified battery actions every 15 min regardless of the status of convergence. The modified battery actions form the lower layer are used by a real-time (backup) controller, which can override the actions to avoid over-/undercharging of the battery at the end of the 15 min interval. The flowchart below shows the Q-learning algorithm used by both upper and lower layers, as discussed in Figure 2a above.

3.1. Upper-Layer Q-learning

The details of the upper offline layer are described below:

3.1.1. State Space

The state space (S^1) is discretised at Δt = 1 h. In Equation (1), *SOC* and *t* denote the battery state of charge state and time step, respectively. It is important to note that the information about generation and load is implicitly included in the time step *t* because the generation and load data profiles are fixed with respect to time during offline optimisation.

$$s_t{}^1 = [SOC, t] \in S. \tag{1}$$

SOC is bounded by maximum and minimum limits, such as:

$$SOC_{\min} \leq SOC(t) \leq SOC_{\max}. \tag{2}$$

The state space is discretised as:

$$S^1_{discrete} = \{S_{i,j}\}, \tag{3}$$

where i = 8 levels (for *SOC*), j = 24 (for *t*). Thus, the total number of states is $8 \times 24 = 192$.

3.1.2. Action Space

At each time step *t* (1 h), one action is selected from the action space (*Af*):

$$Af = \{a|0, \pm 10\%, \pm 20\%, \pm 30\% \cdots\cdots\cdots \pm 100\%\}. \tag{4}$$

where the sign (−) means charging and (+) means discharging, while zero means idle. The percentage actions are with respect to the maximum battery power according to the rating of the battery and its inverter:

$$P_t^{batt} = P_{\max}^{batt} \times Af. \tag{5}$$

3.1.3. Reward

The aim of *Q*-learning is to minimise power imported from the grid; thus, the reward function is given by:

$$r^1(s_t, a_t) = -P_t^{grid} \times \Delta t \times Tariff - C, \tag{6}$$

where P_t^{grid} is the imported grid power and is given by:

$$P_t{}^{grid} = e_t{}^{Net} - P_t{}^{batt}, \tag{7}$$

where $e_t^{Net} = e_t^{demand} - e_t^{PV}$ (net demand). The *C* term in Equation (6) is a penalty factor that is set to a high value (500) if *SOC* exceeds the limits; otherwise, it is set to zero. The import tariff is given as:

$$Tariff_{imp} = \left\{ \begin{array}{r} 0.05£/\text{kWh} \\ 0.08£/\text{kWh} \\ 0.171£/\text{kWh} \end{array} \right. \left. \begin{array}{r} \text{low peak, 05:00 to 11:00; 21 to 00:00} \\ \text{medium peak, 12:oo to 15:00; 19:00 to 20:00} \\ \text{high peak, 01:00 to 04:00; 16:00 to 18:00} \end{array} \right\}. \tag{8}$$

Excess microgrid power can be fed into the grid to benefit from the feed-in tariff, which is $Tariff_{\exp} = 0.033£/\text{kWh}$. Keeping real-time scenarios in mind, the tariff rate is determined based on reliability indicators [21].

3.2. Lower-Layer Q-Learning

The details of the lower online layer are described below:

3.2.1. State Space

The state space (S^2) for a real-time layer, discretised at Δt = 15 min, is given by:

$$s_t{}^2 = [Af,\ D_{f-r}] \in S, \tag{9}$$

where Af denotes the actions provided by the upper offline layer. D_{f-r} is the normalised difference between forecasted and real net power demand, such as:

$$D_{f-r} = (e_t^{Net(\text{Real})} - e_t^{Net(Forecast)})/e_t^{Net(\text{Real})}, \tag{10}$$

The state space is discretised as shown in Equation (11) as k, l indices, such as:

$$S^2_{discrete} = \{S_{k,l}\}, \tag{11}$$

where k = 21 levels for Af and l = 14 levels for D_{f-r}. Therefore, the total number of states in the lower layer is $21 \times 14 = 294$.

3.2.2. Action Space

At each time step of 15 min, one action is selected from the action space *Ar.* Hence,

$$Ar = \{a|(0, \pm 10\%, \pm 20\%, \pm 30\%)\}. \tag{12}$$

The objective of this real-time lower layer is to revisit the offline upper-layer actions and fine-tune them in response to changes in real-time net demand as compared with forecasted net demand. Thus, 10%, 20%, or 30% charging and discharging are allowed on the actions suggested by the upper layer. The zero action means that no action is necessary, and the battery commands remain unchanged.

3.2.3. Reward

The reward function is given by:

$$r^2(s_t, a_t) = -P_t^{grid} \times \Delta t \times Tariff - C, \tag{13}$$

The penalty factor C is set to a high value (500) if SOC exceeds the limits; otherwise, it is set to zero.

3.3. Q-Learning Algorithm

Q-learning is an RL-based algorithm in which transition probabilities can be learned implicitly through experience without any prior knowledge [22]. Model-free Q-learning is not dependent on a model of the environment, and it can handle problems involving stochastic transitions and rewards without adaptation [23]. In Q-learning, an action is taken to maximise its respective future reward at each time step. Thus, R_t^{π} in Equation (14) is the sum of the instant reward at time step t plus the future discounted rewards:

$$R_t{}^{\pi} = r(s_t, a_t) + \sum_{i=1}^{\infty} \gamma^i . r(s_{t+i} . a_{t+i}), \tag{14}$$

The first component of Equation (14) shows the effect of the current action on future rewards, and the second component is the total discounted rewards at time step t under a

given policy π. The action-value function is approximated by repeatedly updating $Q(s_t, a_t)$ through experience as in Equation (15) [24]:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r(s_t, a_t) + \gamma \min Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (15)$$

The RL agent uses the ε-greedy policy to manage its exploration/exploitation strategy. The Q-learning algorithm begins by selecting random actions and evaluating corresponding rewards. The agent tries every decision once and then chooses the one that would result in the highest future reward until the agent learns to maximise the value of the state-action pair and updates the Q-table. Random and greedy actions correspond to exploration (ε) versus exploitation, respectively. This work uses $\varepsilon \leftarrow \varepsilon / \sqrt{M(s) - M_{max}}$ for exploration and exploitation in which $M(s)$ is the number of times a certain action is taken in a specific state. M_{max} is the maximum constant value selected after which greedy actions are selected by the Q-learning algorithm.

4. Simulation Results

A 15 MW of installed PV power has been used for high residential loads. The battery capacity is considered to be 12,000 kWh. The constraints SOC_{max} and SOC_{min} are 100% and 40%, respectively. The maximum allowable charging and discharging limit of the battery is 7200 kW. **The rate at which the battery charges or discharges depends on the action taken by the agent using the action space described in Section 3.** Open source has been used to retrieve data profiles for a region in Denmark [25]. This data source provides forecasted and real net demand for a full year. The data profiles were selected such that they have varying percentages of error between forecasted and actual net demands. **The diverse variations between forecasted and real net demand are incorporated in this work for the validation of the proposed dual-layer RL algorithm.**

Figure 3 shows the variation between forecasted and real net demand on a monthly basis. It is evident from the figure that the total forecasted net demand per month is either higher or lower than the total real net demand per month, as indicated by the positive or negative values. In this regard, the deviation between forecasted and actual (real) net demand by month is tracked and is 10%, 7%, −9%, −11%, 4.5%, 1.9%, −5%, −2%, −2.5%, −2%, 6.5%, and 2.3% from January to December.

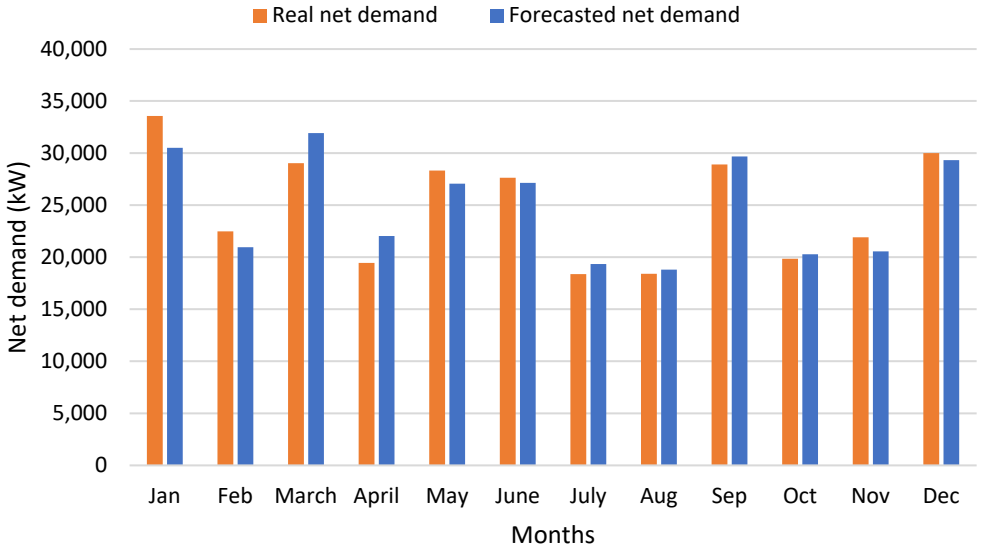


Figure 3. Total forecasted and real net demand per month having diverse variations of errors.

The convergence and cost savings of the proposed dual-layer Q-learning architecture are compared for a full year with those of the online and offline RL algorithms reported in [20]. The hyperparameters used in this work are γ, α, and ε. The hyperparameter γ denotes the discount factor where $0 \leq \gamma \leq 1$. It defines the importance of future

rewards from the next step $t + 1$ to infinity. For example, $\gamma = 0$ suggests that the EMS will consider only the current reward, while $\gamma = 1$ implies that the system considers both the current reward and future long-term rewards equally [23]. In this work, $\gamma = 0.9$ has been used. Parameter α is the learning rate that controls how much the newly obtained reward supersedes the old value of $Q(s_t, a_t)$. For instance, $\alpha = 0$ implies that the newly obtained information is ignored, whereas $\alpha = 1$ implies that the system considers only the newest information. This work uses the α value of 0.5 and 0.7 for upper and lower layers, respectively, whereas ε is equal to 0.7 for offline and online implementations and M_{max} is 15.

The convergence is highly dependent on the hyperparameters, and thus, fine tuning is required to achieve an optimal performance. Figure 4 shows the convergence of offline RL using data for 1 day. It shows that the convergence is attained after 3000 iterations approximately. Q-learning RL displays variations even after convergence due to continuous exploration.

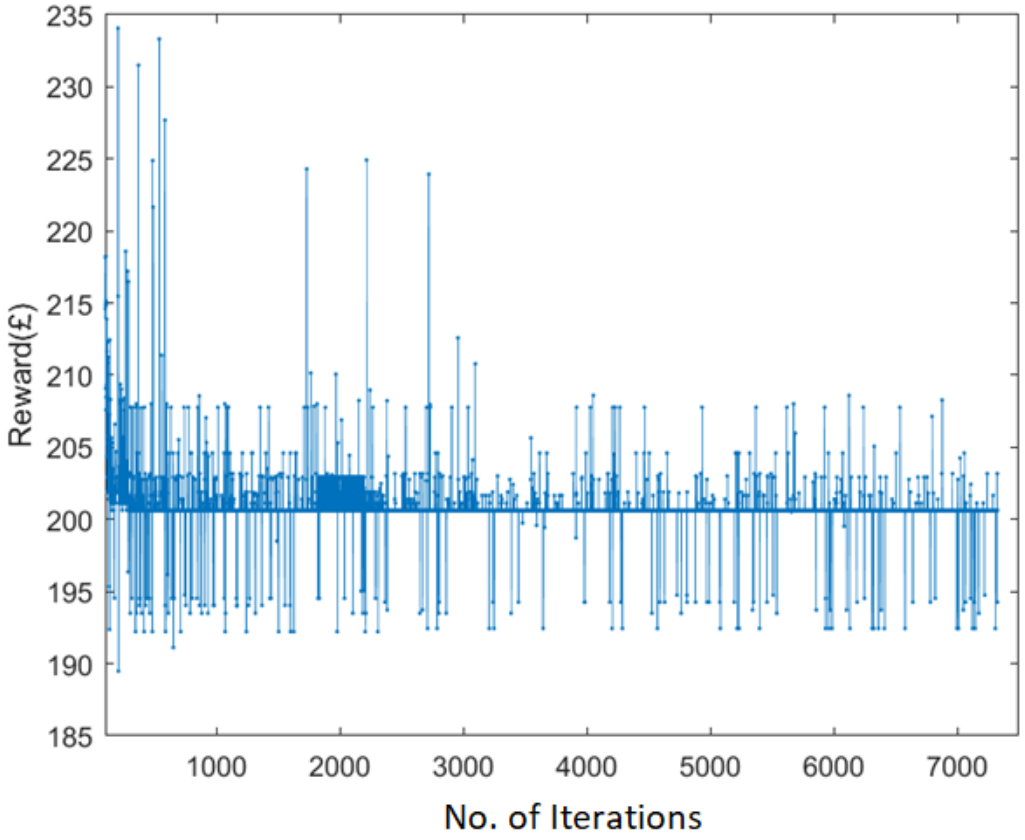


Figure 4. Convergence characteristics of offline RL.

In order to test the stability of the offline RL algorithm shown in Figure 4, the same offline RL algorithm (Figure 2b) is run multiple times using the same data profiles. Variations are observed between each run until convergence is reached.

In Figure 5, the average cumulative reward achieved by running the offline RL algorithm five times is observed. Every run starts showing a different cumulative reward, but towards the end, between 3000 and 5000 iterations, it starts to give the same cumulative reward, as shown in Figure 4. The average and standard deviations of all 3000 iterations are calculated after each run. The bars in Figure 5 show a similar cumulative reward for 1 day after 3000 iterations approximately. The average result of all five simulations in terms of a 1-day cumulative reward is GBP (£) 201. However, it shows variations during convergence expressed as standard deviations. The standard deviation for offline RL is shown to be between 3.7 and 4.4.

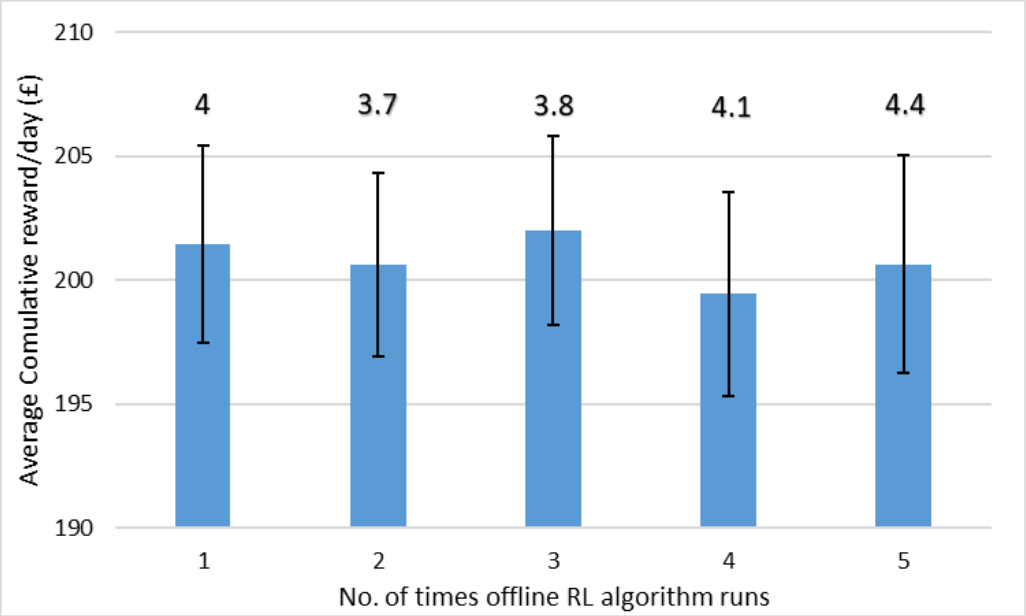


Figure 5. Average rewards for different runs of the offline RL algorithm.

4.1. RL with Zero Forecast Error (Benchmark)

In this ideal case, it is assumed that the forecasted and real net demands are equal; that is, forecast error is zero. Although this case is not real, it will serve as a comparison benchmark for the offline, online, and dual-layer strategies. Here, the time interval is set to 15 min. For each day, the net demand data for that day are used by the *Q*-learning algorithm until convergence is achieved. The algorithm is repeated for 365 days to calculate the total cost of the year. Figure 6 shows the pattern of battery actions and *SOC*.

4.2. Offline RL with Forecast Error

At the beginning of each day, the forecasted PV and load data are gathered as inputs to the RL algorithm. *Q*-learning is then run using the same input data until convergence is achieved. The policy developed at the end of this phase is used to generate the charging, discharging, and idle commands for the next 24 h. This strategy is repeated for each day. Due to the difference between forecasted data used by Q-learning and real data, it is likely that some of the battery commands might violate the SOC limits. Therefore, the Q-learning battery commands are passed to a backup controller that ensures that all physical constraints and limitations are met before actually applying them to the physical system [19].

Figure 7 shows the simulation of offline RL. It shows that the extra forecasted PV is available from 13:00 and 15:00 as the net forecasted net demand is negative. However, according to real net demand, the extra PV is available between 10:00 and 12:00. As a result, battery optimal actions, based on forecasted profiles, are suboptimal as they charge the battery during the forecasted time for the extra PV not during the actual time when the PV is available.

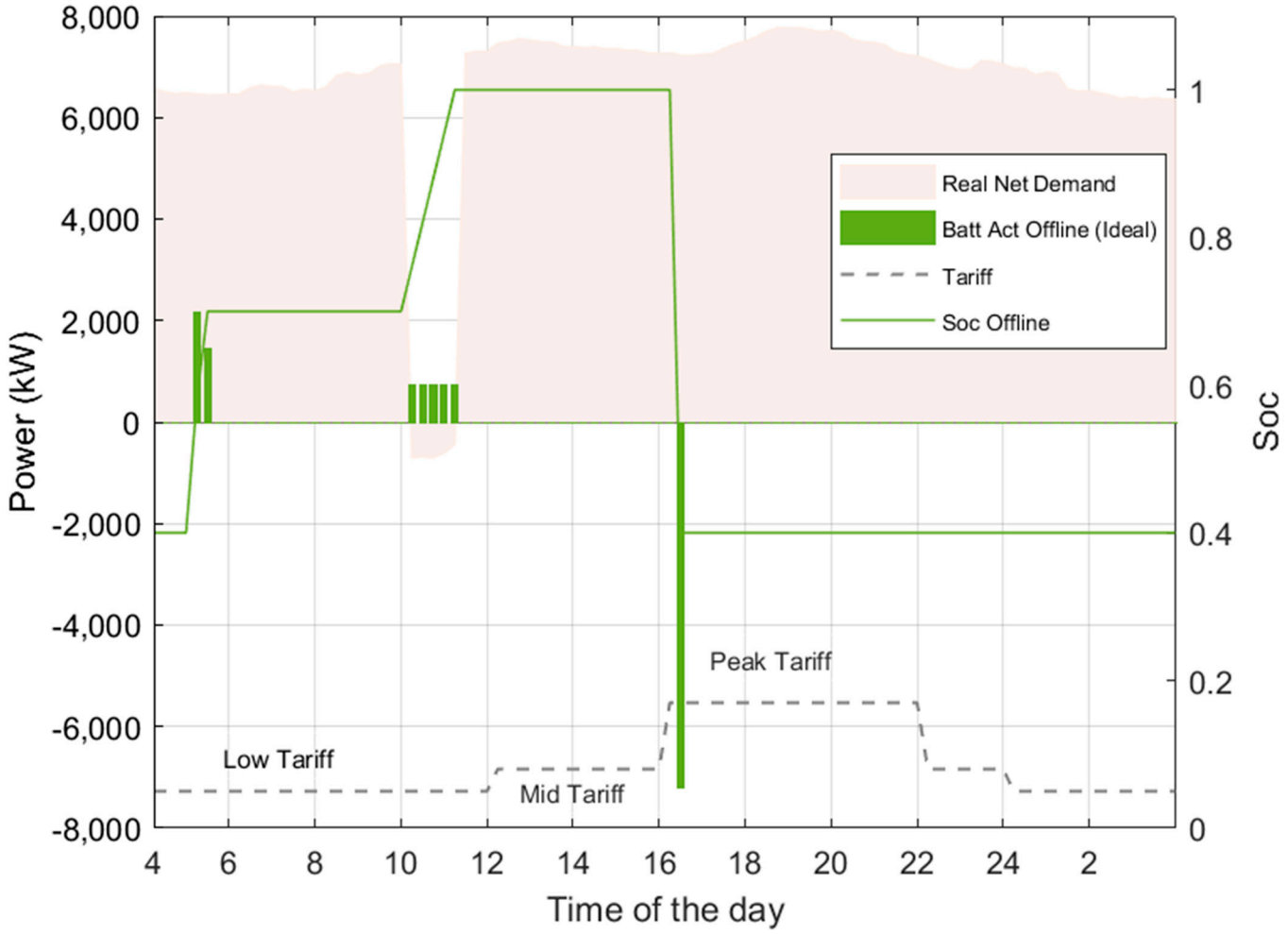


Figure 6. Offline RL (ideal case) with no forecast error using real-time data.

4.3. Online RL with Forecast Error

In online RL, *Q*-learning is applied directly to real data in real time. Therefore, the agent learns the optimal policy by interacting with the real system. There is no pretraining in this online approach unlike the offline techniques. However, there is an initialisation problem of the state action pair of the *Q*-table, which is addressed, at the beginning of the first day, by initialising the *Q*-table with a short-sighted future reward by setting the hyperparameter γ to 0. This simple initialisation step reduces the convergence time substantially [17]. After that, the *Q*-table will be updated in real time by interacting with the environment. The online *Q*-learning algorithm updates the actions of the battery and dispatches them every 15 min in real time. Learning can be very slow, and before convergence, the performance would be suboptimal. With time, the agent develops an optimal policy. The function of the backup controller in the online *Q*-learning implementation is the same as that for offline *Q*-learning. Figure 8 illustrates the battery commands suggested or implemented during real-time learning with its respective *SOC* levels. It is clear that the battery is commanded to charge when an extra PV is available.

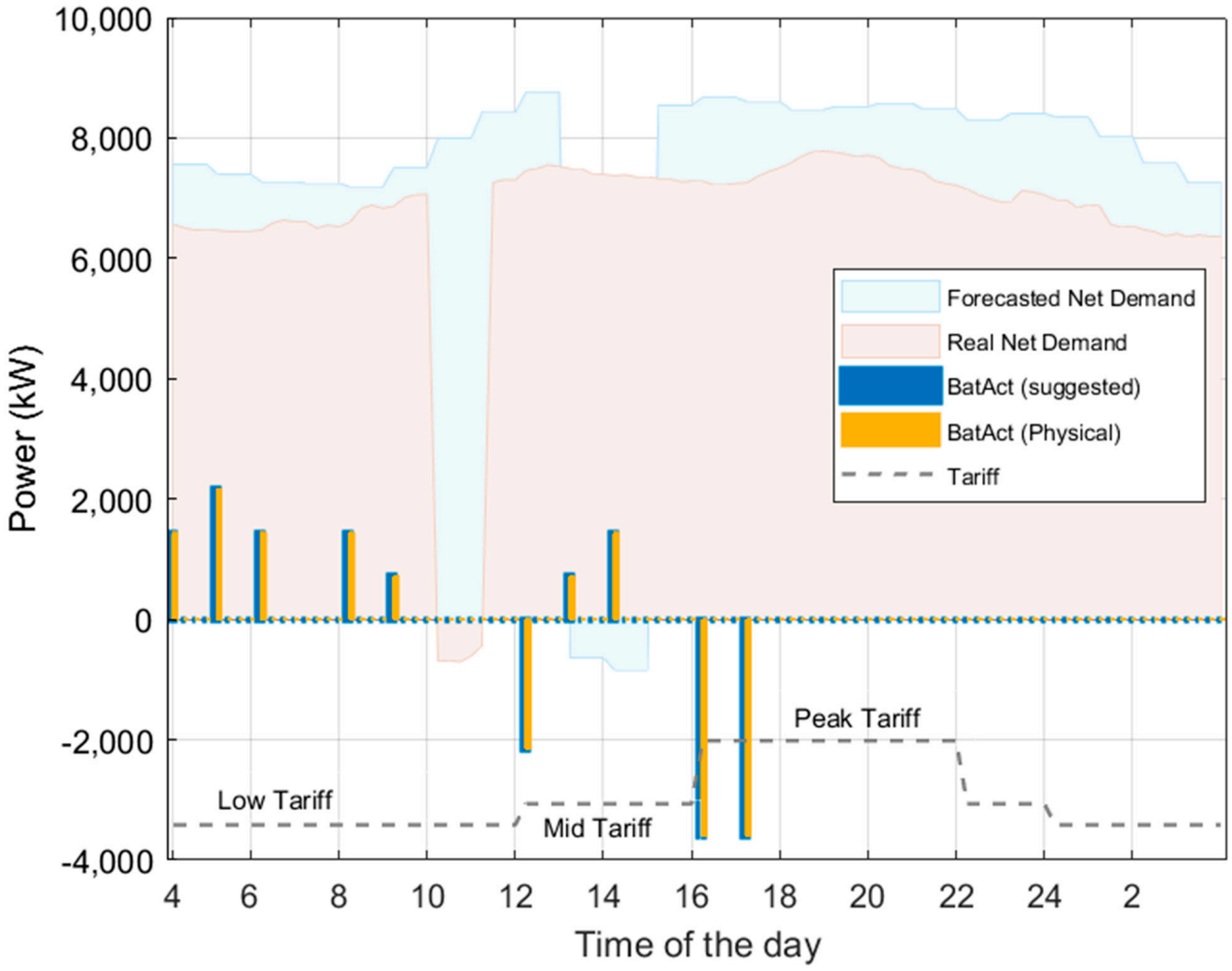


Figure 7. Offline RL using 1-day forecasted data.

4.4. Proposed Dual-Layer Q-Learning Algorithm

As mentioned above, the first layer is conducted offline to produce directive commands for the battery system for a 24 h horizon. It uses forecasted data for generation and load. The second *Q*-learning layer refines these battery commands every 15 min interval by considering the changes happening in RES and load demand in real time. Figure 9 shows the simulation results of the proposed dual-layer Q-learning. The upper offline layer commands would charge the battery between 13.00 and 15.00 during which forecast data suggest that an extra PV will be available. Additionally, it would discharge just before 12:00 to create some space within the battery for PV charge. It can be noticed that the lower layer made adjustments to the battery command, so the charging takes place between 10.00 and 11.00 during which a real extra PV is available. This shows how the suggested dual-layer strategy can improve performance according to real-time change in the net demand.

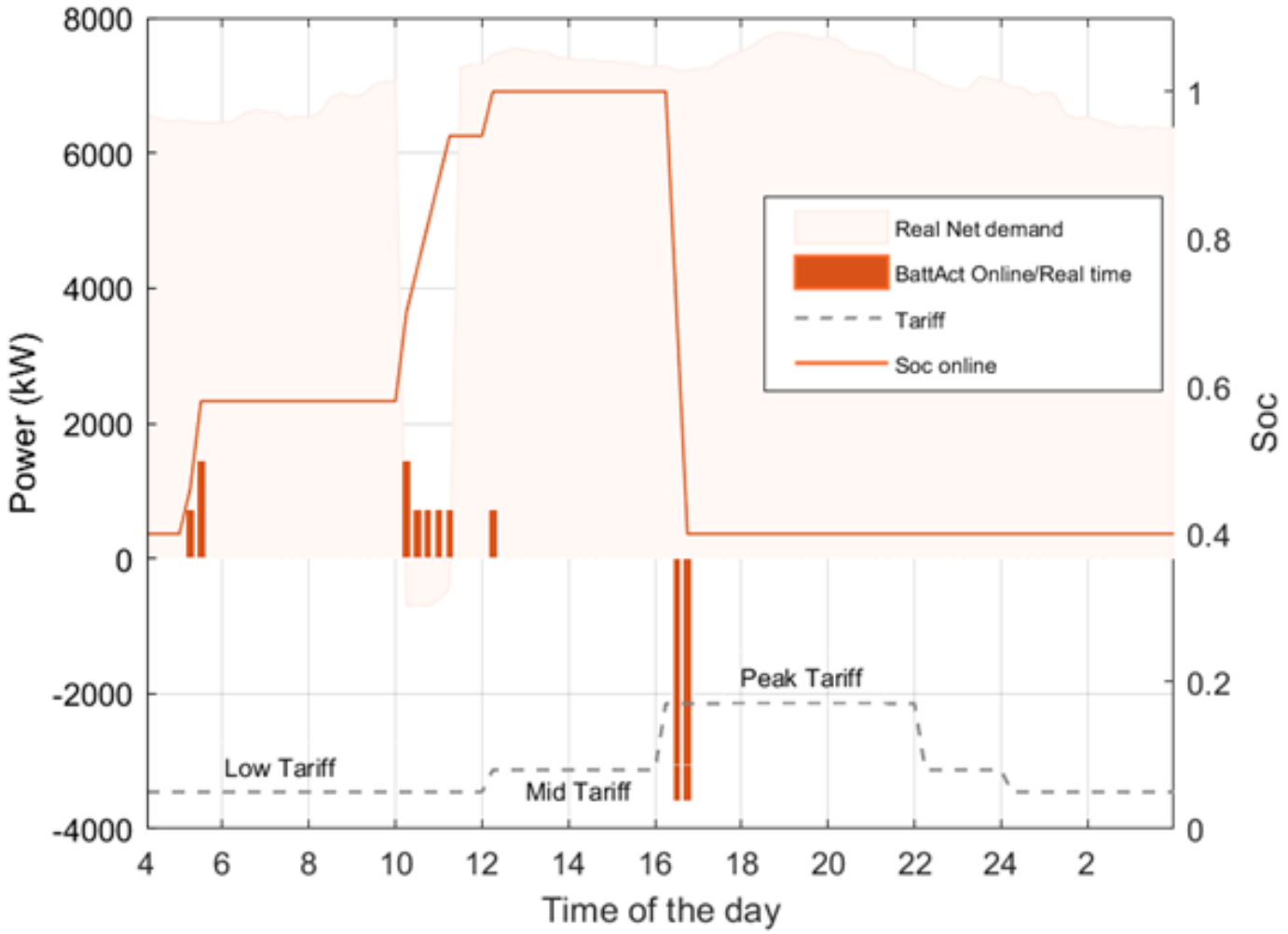


Figure 8. The online (real-time) RL algorithm using 1-day real data profiles.

4.5. Comparison of Different RL Architectures

In Figure 10, the proposed dual-layer strategy is compared with offline and online RL approaches. Furthermore, an ideal case with zero forecast error is included as a benchmark. The temporal responses are shown after convergence reached for all approaches. It can be seen that in the ideal case, online RL and dual-layer RL, the charge of the battery happens during excess PV, and discharge happens during the high tariff. Offline RL, however, follows the forecasted profile. Even though the pattern of actions seems similar between online and dual layer, the effect on cost is different, as shown in the following section.

4.6. Cost Comparison after Convergence

A 1-day performance comparison of the above strategies after convergence is shown in Figure 11. It shows that the proposed dual-layer strategy outperforms the online strategy, and its running cost is the closest to the ideal benchmark case. To test the stability, all four algorithms were run five times, and the standard deviation varies between 0.5 and 1.3. Figure 11 shows that the stability of the dual-layer RL approach is very close to the ideal case scenario when a standard deviation is considered, such as 0.75 and 0.5, respectively. Therefore, the performance of the dual-layer strategy is the closest to that of the ideal case scenario in terms of cost, convergence, and stability.

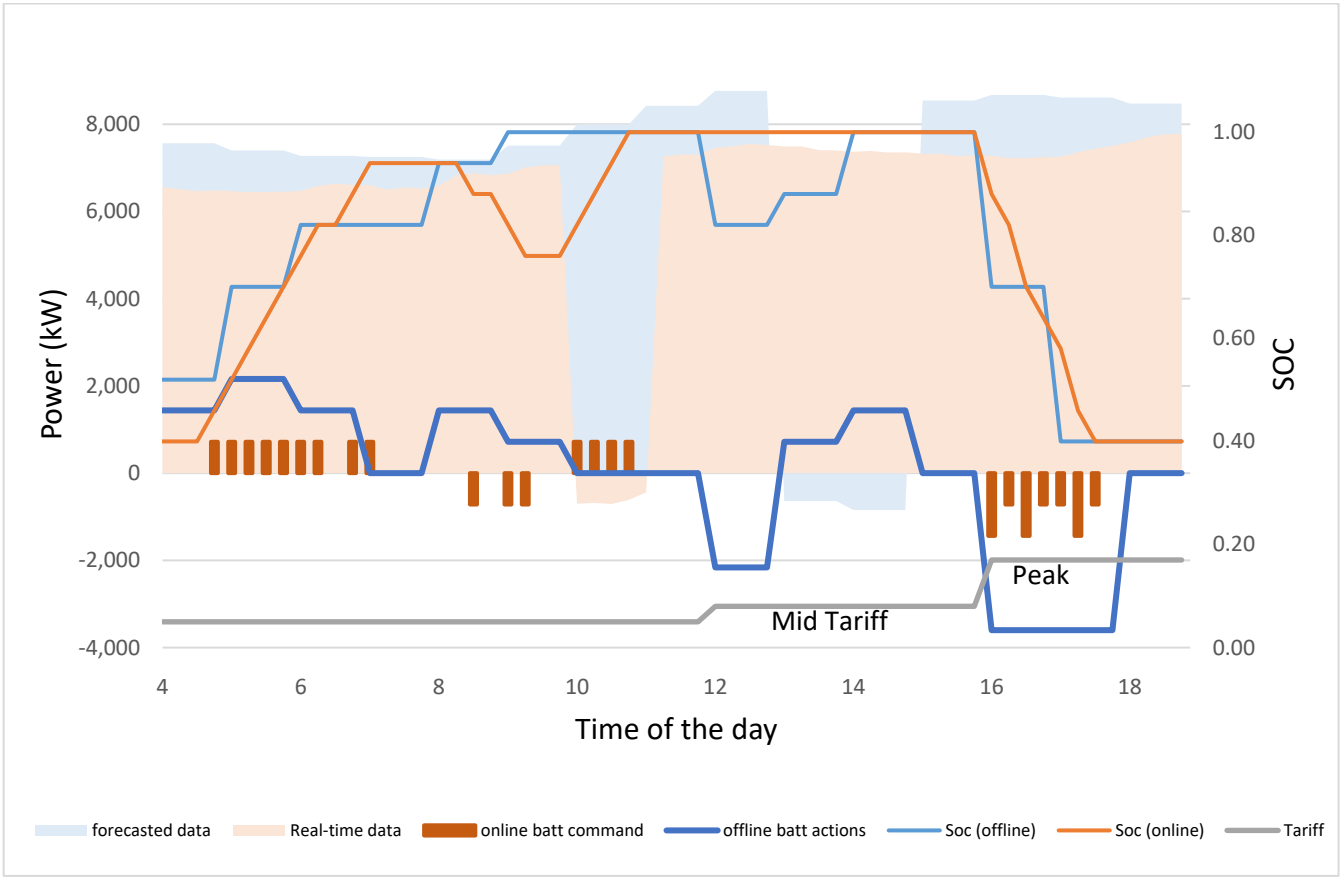


Figure 9. Proposed dual-layer (upper offline and lower real-time layer's) response.

All four RL algorithms are tested on the 365 days of the year with varied predicted and real data profiles (net demand) to allow for a full comparison. The monthly costs are illustrated in Figure 12. The cost is calculated using Equations (6)–(8) and (13). It can be seen that the dual layer behaves approximately similarly as the ideal case scenario after 2 months in terms of monthly operating costs. The dual layer performed consistently throughout the year after convergence. In comparison with online RL, the early convergence of the dual layer means more cost reduction on a yearly basis.

4.7. Performance Consistency

Each algorithm was run five times using the same forecasted and real net demands. The average yearly costs and average standard deviations are recorded for all algorithms by running each method five times to check the consistency of every algorithm, as shown in Figure 13. The proposed dual-layer RL has an average standard deviation equal to 2.4 in a complete year, which is slightly higher than 1.8 for the ideal case. Offline and online RLs have standard deviations (average) of 7.8 and 4.9, respectively. The superiority of the dual layer in terms of cost and convergence is clear.

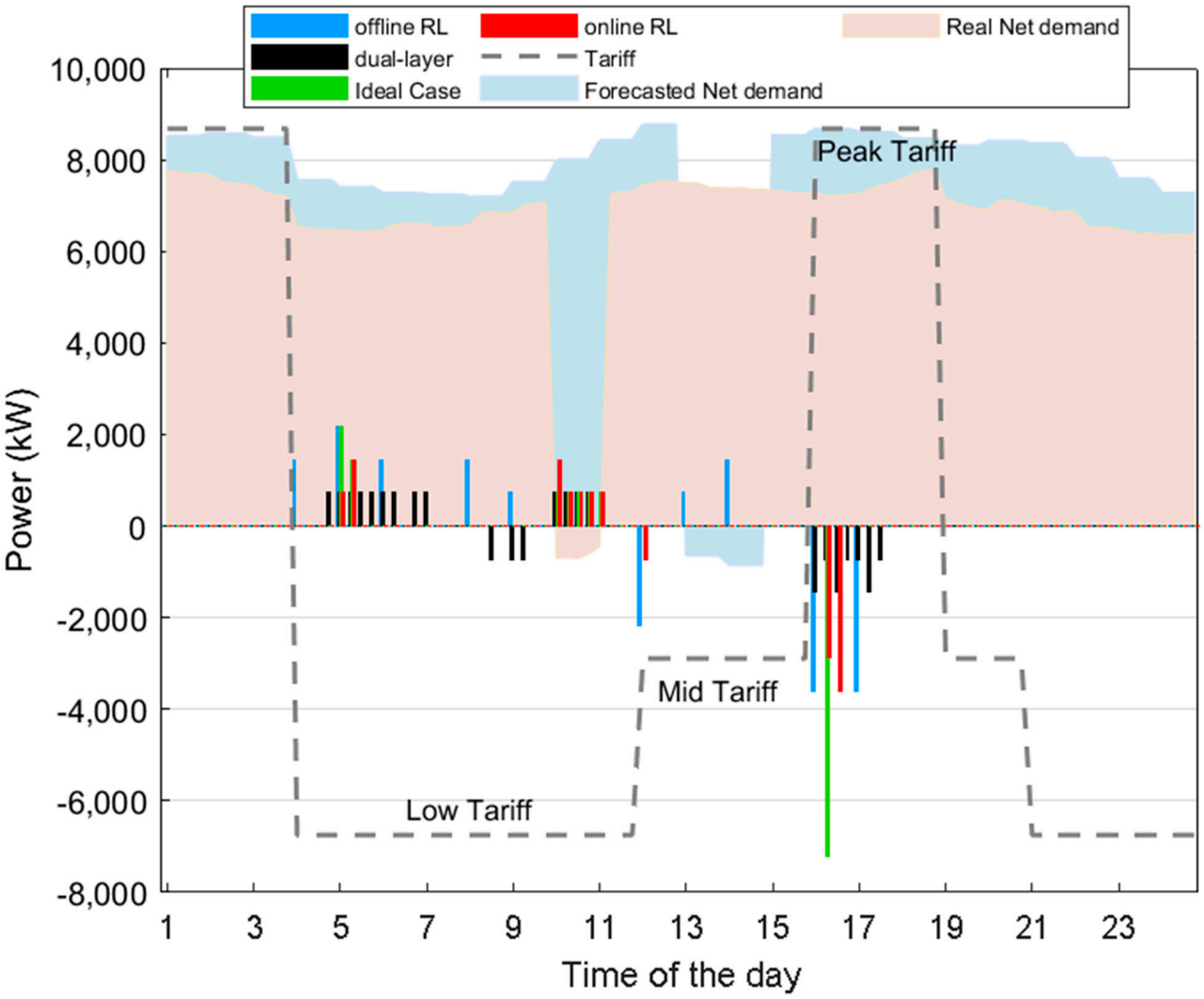


Figure 10. Comparison of the suggested dual layer with other (three) RL approaches in the literature.

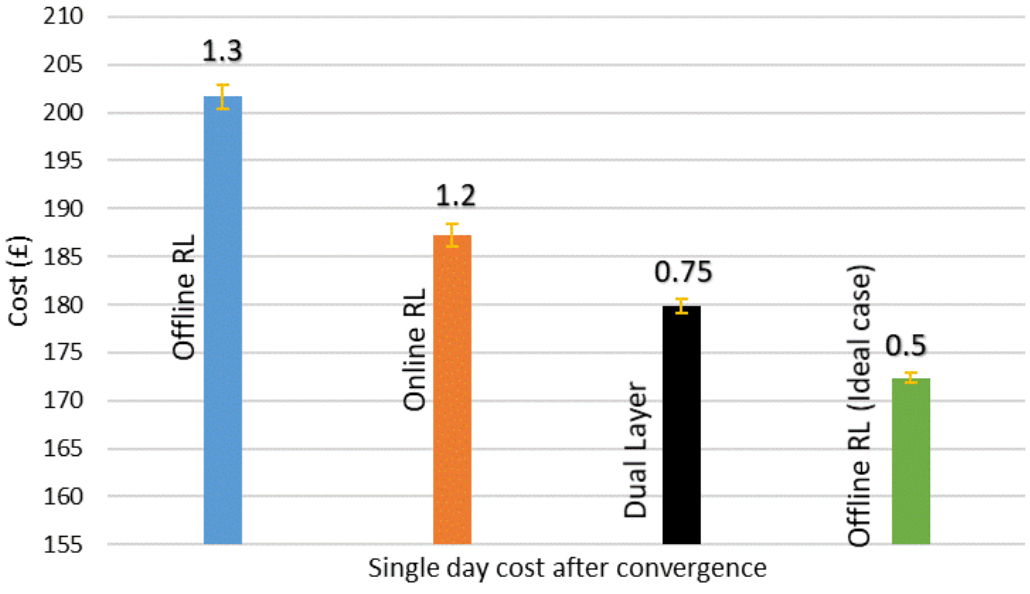


Figure 11. The average 1-day cost of the five simulations after convergence of three RL approaches with a dual layer, and error bars represent standard deviation across all approaches.

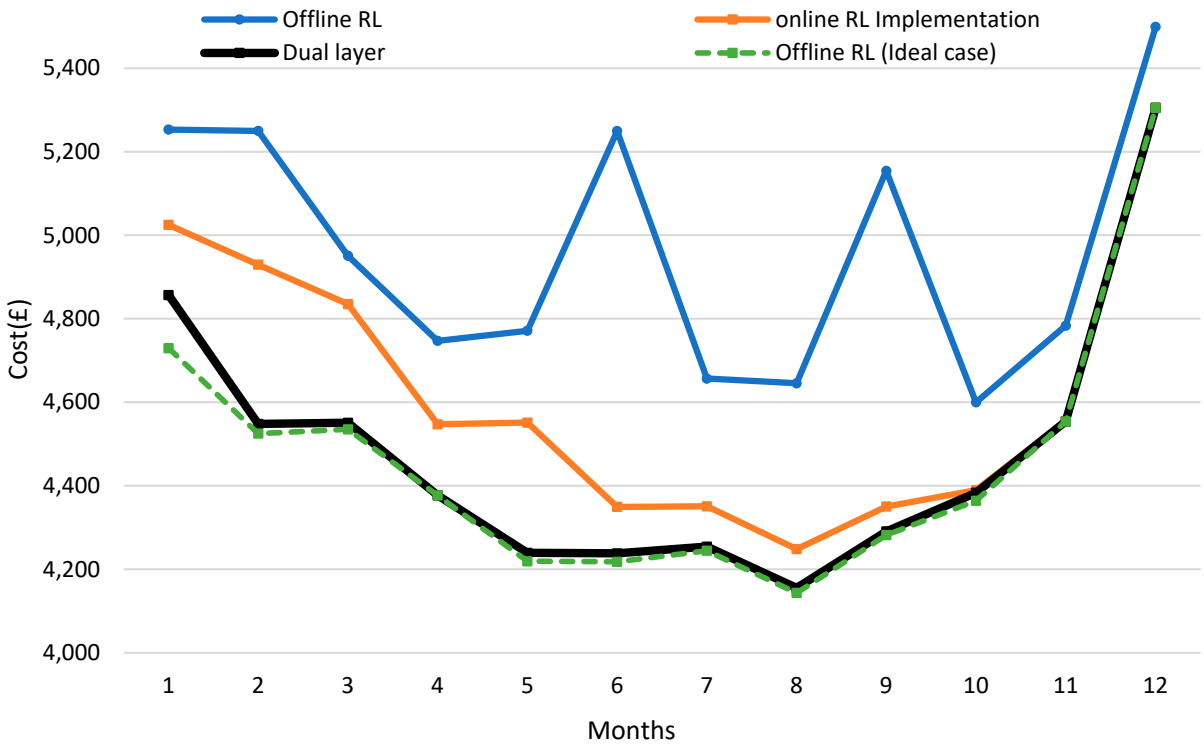


Figure 12. Yearly cost comparison, dual layer vs. other RL approaches.

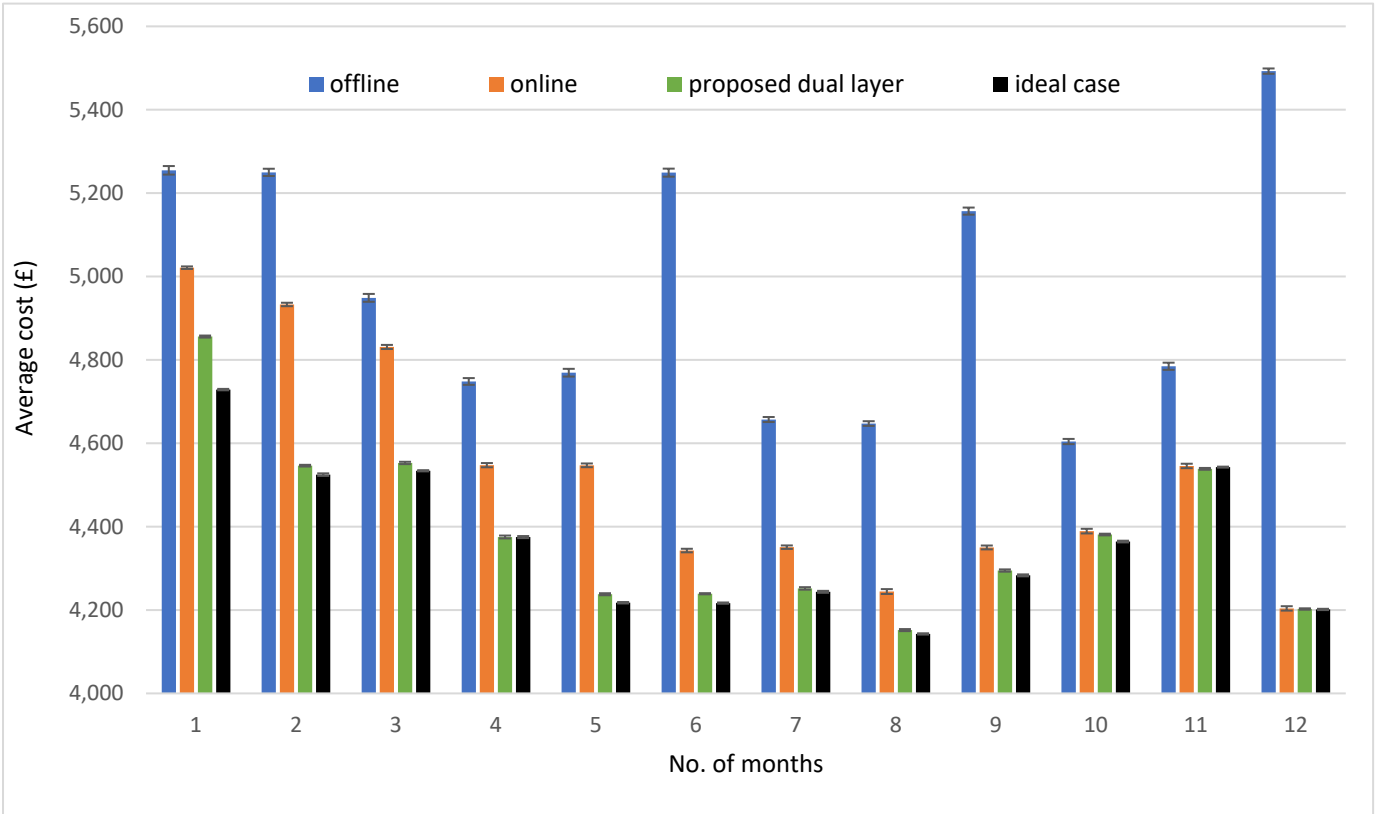


Figure 13. Five time consistency check simulation runs to calculate the average cost and standard deviation per month.

5. Conclusions

A new dual-layer *Q*-learning strategy to optimally manage battery storage systems in grid-connected microgrids has been presented. The upper layer, which runs every hour, is conducted offline using forecasted data to produce directive commands for the battery system for a 24 h horizon. The second layer refines these battery commands every 15 min by considering the changes happening in RES and load demand in real time. An original contribution of this work is not only the development of a new algorithm of a dual-layer RL, but also a comparison with other two well-known RL algorithms, namely, offline and online *Q*-learning, to prove its superiority. Simulation results show that the proposed strategy has a faster convergence rate than the reported online strategy [16,19]. In real-time optimisation, time is crucial as delay decisions cannot be effective. The proposed approach also addresses the issue of time constraint associated with real-time optimisation. Unlike offline RL techniques, this suggested approach does not require predicting real-time data prior to implementing *Q*-learning. In this study, different variations between forecasted and real net demand were incorporated into the analysis of the dual-layer RL algorithm for validation. The performance of the dual-layer *Q*-learning has also been benchmarked against the ideal case with zero forecast error and shows close performance.

Author Contributions: Conceptualisation, M.A. and K.H.A.; methodology, K.H.A.; validation, M.A. and S.D.; formal analysis, A.A.T.; investigation, K.H.A. and M.A.; resources, A.A.T.; writing—original draft preparation, K.H.A.; writing—review and editing, M.A., S.D., A.A.T. and K.H.A.; supervision, M.A. and A.A.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Engineering and Physical Sciences Research Council (EPSRC) grant number EP/T025875/1, and the APC was partially funded by the editor allowance.

Data Availability Statement: The data are available from the lead or the corresponding author upon reasonable requests.

Acknowledgments: K.H.A. would like to thank the Sukkur Institute of Business Administration University, Pakistan, for the fully funded PhD studentship.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

Nomenclature	Description	Nomenclature	Description
EMS	energy management systems	Af	offline action space
DP	dynamic programming	Ar	real-time action space
ESS	energy storage system	SOC/Soc	state of charge
FL	fuzzy logic	P_{batt}	battery power
GA	genetic algorithm	P^{grid}	main grid power
LP	linear programming	D_{f-r}	normalised difference between forecasted and real net power demand
MAS	multiagent system	S	state space
MGs	microgrids	$S_{discrte}$	discrete state
MILP	mixed interlinear programming	Δt	difference between two time intervals
MINLP	mixed integer nonlinear programming	α	learning rate
MIP	mixed integer programming	e^{net}	net demand
NLP	nonlinear programming	γ	discount factor
NN	neural networks	ε	exploration rate
FAM	function approximation methods	e^{real}	real net demand
RES	renewable energy sources	$e^{forecassted}$	forecasted net demand

References

1. Gavilema, Á.O.T.; Álvarez, J.D.; Moreno, J.L.T.; García, M.P. Towards optimal management in microgrids: An overview. *Energies* **2021**, *14*, 5202. [CrossRef]
2. Jabir, H.J.; Teh, J.; Ishak, D.; Abunima, H. Impacts of demand-side management on electrical power systems: A review. *Energies* **2018**, *11*, 1050. [CrossRef]

3. Do, L.P.C.; Lyócsa, Š.; Molnár, P. Residual electricity demand: An empirical investigation. *Appl. Energy* **2021**, *283*, 116298. [CrossRef]
4. Sanjeevikumar, P.; Samavat, T.; Nasab, M.A.; Zand, M.; Khoobani, M. Machine learning-based hybrid demand-side controller for renewable energy management. In *Sustainable Developments by Artificial Intelligence and Machine Learning for Renewable Energies*; Elsevier: Amsterdam, The Netherlands, 2022; pp. 291–307. [CrossRef]
5. Truma, Y. *Linear Programming: Theory; Algorithms; Applications*; Springer: New York, NY, USA, 2014.
6. Yang, X.S.; Koziel, S.; Leifsson, L. Computational optimization, modelling and simulation: Past, present and future. *Procedia Comput. Sci.* **2014**, *29*, 754–758. [CrossRef]
7. Thudumu, S.; Branch, P.; Jin, J.; Singh, J. A comprehensive survey of anomaly detection techniques for high dimensional big data. *J. Big Data* **2020**, *7*, 42. [CrossRef]
8. Kantor, I.; Robineau, J.L.; Bütün, H.; Maréchal, F. A Mixed-Integer Linear Programming Formulation for Optimizing Multi-Scale Material and Energy Integration. *Front. Energy Res.* **2020**, *8*, 49. [CrossRef]
9. Ji, Y.; Wang, J.; Xu, J.; Fang, X.; Zhang, H. Real-time energy management of a microgrid using deep reinforcement learning. *Energies* **2019**, *12*, 2291. [CrossRef]
10. Guo, C.; Wang, X.; Zheng, Y.; Zhang, F. Real-time optimal energy management of microgrid with uncertainties based on deep reinforcement learning. *Energy* **2022**, *238*, 121873. [CrossRef]
11. Teo, T.T.; Logenthiran, T.; Woo, W.L.; Abidi, K. Fuzzy logic control of energy storage system in microgrid operation. In Proceedings of the IEEE PES Innovative Smart Grid Technologies Conference Europe, Melbourne, VIC, Australia, 28 November–1 December 2016; pp. 65–70. [CrossRef]
12. Ouramdane, O.; Elbouchikhi, E.; Amirat, Y.; Gooya, E.S. Optimal sizing and energy management of microgrids with Vehicle-to-Grid technology: A critical review and future trends. *Energies* **2021**, *14*, 4166. [CrossRef]
13. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.E.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef] [PubMed]
14. Blondin, M.J.; Hale, M. An Algorithm for Multi-Objective Multi-Agent Optimization. In Proceedings of the American Control Conference, Denver, CO, USA, 1–3 July 2020; Volume 2020, pp. 1489–1494. [CrossRef]
15. Eddy, F.Y.S.; Gooi, H.B. Multi-agent system for optimization of microgrids. In Proceedings of the 8th International Conference on Power Electronics—ECCE Asia: "Green World with Power Electronics", ICPE 2011-ECCE Asia, Jeju, Korea, 30 May–3 June 2011; pp. 2374–2381. [CrossRef]
16. Mbuwir, B.V.; Ruelens, F.; Spiessens, F.; Deconinck, G. Battery energy management in a microgrid using batch reinforcement learning. *Energies* **2017**, *10*, 1846. [CrossRef]
17. Kim, S.; Lim, H. Reinforcement learning based energy management algorithm for smart energy buildings. *Energies* **2018**, *11*, 2010. [CrossRef]
18. Yang, J.J.; Yang, M.; Wang, M.X.; Du, P.J.; Yu, Y.X. A deep reinforcement learning method for managing wind farm uncertainties through energy storage system control and external reserve purchasing. *Int. J. Electr. Power Energy Syst.* **2020**, *119*, 105928. [CrossRef]
19. Chen, P.; Liu, M.; Chen, C.; Shang, X. A battery management strategy in microgrid for personalized customer requirements. *Energy* **2019**, *189*, 116245. [CrossRef]
20. Ali, K.H.; Sigalo, M.; Das, S.; Anderlini, E.; Tahir, A.A.; Abusara, M. Reinforcement learning for energy-storage systems in grid-connected microgrids: An investigation of online vs. offline implementation. *Energies* **2021**, *14*, 5688. [CrossRef]
21. Nasab, M.A.; Zand, M.; Padmanaban, S.; Bhaskar, M.S.; Guerrero, J.M. An efficient, robust optimization model for the unit commitment considering renewable uncertainty and pumped-storage hydropower. *Comput. Electr. Eng.* **2022**, *100*, 107846. [CrossRef]
22. Anderlini, E.; Forehand, D.I.M.; Bannon, E.; Xiao, Q.; Abusara, M. Reactive control of a two-body point absorber using reinforcement learning. *Ocean Eng.* **2018**, *148*, 650–658. [CrossRef]
23. Delgado, R.; Stefancic, J.; Harris, A. *Critical Race Theory: An Introduction*, 2nd ed.; NYU Press: New York, NY, USA, 2012; Volume 3, Available online: http://www.jstor.org/stable/j.ctt9qg9h2 (accessed on 8 February 2021).
24. Watkins, C.J.C.H.; Dayan, P. Q-Learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
25. Minde, A. Data Platform—Open Power System Data. CoSSMic 2020. Available online: https://data.open-power-system-data.org/household_data/ (accessed on 8 February 2021).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.