Prioritized Experience Replay Based Deep Distributional Reinforcement Learning for Battery Operation in Microgrids

Deepak Kumar Panda[1], Oliver Turner[1], Saptarshi Das[1,2], and Mohammad Abusara[3]

***Abstract*: Reinforcement Learning (RL) provides a pathway for efficiently utilizing the battery storage in a microgrid. However, traditional value-based RL algorithms used in battery management focus on formulating the policies based on the reward expectation rather than its probability distribution. Hence the scheduling strategy is solely based on the expectation of the rewards rather than the distribution. This paper focuses on scheduling strategy based on probability distribution of the rewards which optimally reflects the uncertainties in the incoming dataset. Furthermore, the prioritized experience replay samples of the training experience are used to enhance the quality of the learning by reducing bias. The results are obtained with different variants of distributional RL algorithms like C51, Quantile Regression Deep *Q*-Network (QR-DQN), Fully Quantizable Function (FQF), Implicit Quantile Networks (IQN) and rainbow. Moreover, the results are compared with the traditional deep *Q*-learning algorithm with prioritized experienced replay. The convergence results on the training dataset are further analyzed by varying the action spaces, using randomized experience replay and without including the tariff-based action while enforcing the penalties for violating battery SoC limits. The best trained *Q*-network is tested with different load and PV profiles to obtain the battery operation and costs. The performance of the distributional RL algorithms is analyzed under different schemes of Time of Use (ToU) tariff. QR-DQN with prioritized experience replay has been found to be the best performing algorithm in terms of convergence on the training dataset, with least fluctuation in validation dataset and battery operations during different tariff regimes during the day.**

***Keywords:* Microgrid, Battery, Deep distributional *Q*-Learning, tariff, prioritized experience replay**

NOMENCLATURE

Environment Description

Symbol	*Meaning*
t	Time instant representing the steps.
T	Final time instant representing the end of the episode.
SoC	State of charge of the battery.
S_t	Tariff state depending upon the time step t
S_c	Controllable battery state representing the state of charge at time step t
$P_{\text{Net}}(t)$	Net power to be met which acts as an external state at time step t
Φ_t	Vector of the states which is utilized in the state transition process for samples j at time step t.
$P_b(t)$	Battery power charged or discharged at time step t.
$P_L(t)$	Load demand at time step t.
$P_{PV}(t)$	Solar PV generation at time step t.
n	Discretization levels in the action space.
$\mathcal{P}$	Vector of battery power discretized at n levels.
$P_b^{\text{ov}}(t)$	The dispatched battery power when the agent experience a penalty as battery operates at extreme SoC ends at time step t.
a_t	Actions taken by the agent based on the Q-value and following a policy π at time step t.
$P_g(t)$	Net power imported from the grid at time step t.
E_b	Energy capacity of the battery.

1) Centre for Environmental Mathematics, Faculty of Environment, Science and Economy, University of Exeter, Penryn Campus, Cornwall TR10 9FE, United Kingdom (Email: dp457@exeter.ac.uk, ollie-t@live.co.uk, saptarshi.das@ieee.org).
2) Institute for Data Science and Artificial Intelligence, University of Exeter, Laver Building, North Park Road, Exeter, Devon EX4 4QE, United Kingdom.
3) Department of Renewable Energy, Faculty of Environment, Science and Economy, University of Exeter, Penryn Campus, Cornwall TR10 9FE, UK. (Email: M.Abusara@exeter.ac.uk).

C_{S_t}	The tariff cost corresponding to the state S_t at time step *t*.
$\rho(\Phi_t, a_t)$	Reward or penalty experienced by the agent.

Deep Reinforcement Learning and Prioritized Experienced Replay Parameters

Symbol	*Meaning*
$V(\Phi)$	Value function at state Φ
γ	Discount factor for computing Temporal Difference error.
π	Policy for exploration or exploitation.
ε	Per-cent agent exploration.
e_t	Collection of tuple consisting of the state Φ_t, next state Φ_{t+1}, action a_t and reward ρ
D_j	The replay buffer consisting of the state transition samples e_t for training the deep *Q*-network of the *j*th sample
N	The strength of the replay buffer.
δ_{TD}	Temporal difference error
θ_i	The parameters of the deep *Q*-network which maps the values from the state action pair (Φ, a)
θ_i^-	The parameters of the deep neural network which adjusts the target value to compute δ_{TD}
$Q(\Phi_t, a_t)$	The value of the state-action pair (Φ, a)
$\hat{l}_j$	The loss function to be optimized for training the deep *Q*-network of the *j*th sample.
y_j	The target used to compute the temporal difference error of *j*th sample.
C	The number of steps required to update the target y_j to compute the temporal-difference error.
$P(i)$	The probability of weighing a transition sample *i* based on δ_{TD}.
ε	The random noise added to prevent the edge case transition.
w_j	Importance coefficient of transition samples to reduce the bias introduced due to sampling error of the *j*th sample.
p_i	Probability of sampling the transitions in the replay buffer
α	The coefficient that puts importance to selection of samples based on p_i.
β	The coefficient that puts importance to reduce the bias.

Distributional Reinforcement Learning Parameters

Symbol	*Meaning*
Z	The distribution of the rewards over the entire episode.
Z	The space of distribution with bounded moments.
T	Distributional Bellman operator.
Y	Distribution belonging to Z operated by the distributional Bellman operator T under policy π.
P^{π}	Transition operator on the *Z*.
U, V	Random variables
F, G	Cumulative distribution function for the random variables U, V
$\bar{d}_p$	*p*-Wasserstein metric
Δz	The value of each atom the quantile distribution.
$\check{N}$	Number of bins in quantile distribution.
Θ	Probabilities of each atom.
$\mathcal{X}$	State space in distributional RL.
$\mathcal{A}$	Action space in distributional RL.
$R(\Phi, a)$	Reward obtained from the state-action in distributional RL.
$\bigvee_{\mathrm{MIN}}$	Maximum atom value.
$\bigvee_{\mathrm{MAX}}$	Minimum atom value.
$\hat{p}_j$	Probability of atoms at projected quantile distribution of the *j*th sample.
Φ	Projection of the reward distribution *Z*.

Φ'	Next state distribution.
$\tilde{\Theta}$	Parameter of the projected distribution which is obtained by minimizing Kullback Leibler Divergence.
$\tilde{\alpha}$	Leaning rate of the parameters of *Q*-network.
$\bar{d}_p$	*p*th Wasserstein metric between two distributions.
$\bar{d}_\infty$	Maximal Wasserstein metric between two distributions.
$\breve{\mathrm{Y}}$	Distribution with the bounded first moment.
$\breve{\mathrm{U}}$	Uniform distribution over $\breve{N}$ Dirac delta function.
δ	Dirac delta function.
Y	Target quantile distribution to minimize the Huber loss.
τ	Quantile midpoints.
θ	Parameters of the
ρ_τ	Huber loss associated with the quantile midpoints
Π_{W_1}	The operator that projects the distribution to minimize 1-Wasserstein metric.
F_U	The quantile function associated with the random variable U
β	The distorted risk measure.
H	Heavyside step function.
$q_{\breve{\theta}}$	Value function for the state-action pair for double *Q*-learning having the parameters $\breve{\theta}$.
$q_{\breve{\theta}^-}$	Target state and action pair to compute the TD error for double *Q*-learning having parameters $\breve{\theta}^-$.
v_η	Value stream in dueling network
$\tilde{a}_\psi$	The advantage stream in the dueling network.
$\mathbf{b}_{\text{noisy}}, \mathbf{W}_{\text{noisy}}$	Bias and weights of the noisy nets.

1. INTRODUCTION

Microgrids are self-supporting generation sources that incorporate Renewable Energy Sources (RESs) and various energy storage devices, and maybe fossil fuel generation sources like diesel or gas generators. The fundamental focus lies in employing battery energy storage in microgrid is to fulfil the load requirements and harness surplus solar energy for recharging purposes. This categorically aligns with sustainable and cleaner strategies aimed at addressing the growing demands for electrical energy. Whether isolated or connected to the grid, microgrids are controllable entities responsible for supplying quality and reliable power. However, the dispatchable energy storage devices are used to mitigate the uncertainty and variability posed by intermittent RES. Controllable energy storage devices can shift the energy from peak hours to off-peak hours, supply shortage power to load or absorb excess power from RES. The ramping capability of the energy storage devices also helps compensating for the uncertainty and high variability of renewable energy sources and prevents the ramping up and down of baseload units (Kintner-Meyer et al., 2010).

Battery Energy Storage Systems (BESS) have a particular advantage as they do not have a no-load cost and minimum power input/output levels for charging and discharging. Furthermore, they can enhance the efficiency of the existing power network assets (Pudjianto et al., 2013). BESS are commonly used to curtail the peak loads and reduce the need for demand forecast and peak load generators (Mahmud et al., 2018), provide frequency regulation in the power system (Kottick et al., 1993), control active and reactive power flow in a stand-alone wind-diesel system (Sebastián and Alzola, 2010), improve power quality and transient stability of wind farms (Zeng et al., 2006), act as active filters in grid-connected mode or UPS in a standalone mode (Ming-Tsung Tsai et al., 1995). BESS have also been indirectly used in the power system supplementing the operations of STATCOM devices (Cheng et al., 2006; Leung and Sutanto, 2003; Yang et al., 2001), for voltage control and dip mitigation in distribution systems, improving the stability and power quality of wind farms and load levelling for energy management, power factor correction, spinning reserve and harmonic filter. They are also used for optimal discharge strategies under the line and generator contingency scenarios (Panda and Das, 2021) at different locations in the grid while supplementing the stochastic wind power.

In a residential environment, the batteries in grid-connected PV are utilized to minimize grid exports, improve consumer finances from the retail electricity tariffs, and formulate variable pricing to increase self-consumption (Braun et al., 2009). In battery technology, lead acids dominate the market due to lower production costs (Chang et al., 2009). They are of two types: floating valve-regulated lead-acid (FRLA) and open flooded ones. Li-ion, NaS, NiCd are the leading battery technologies in high power density batteries (Divya and Østergaard, 2009). Li-ion is mainly used for microgrid applications as it has high energy density and higher efficiency (Tervo et al., 2018). The efficiency of the Vanadium redox battery is discussed in (Guggenberger et al., 2012), based on known climatic conditions and load requirements. Lead-acid batteries contain low energy density even at high mass. Vanadium redox batteries are better in terms of high efficiency and scalability, fast response, long life, and low maintenance,

making them ideal for power smoothing applications for fluctuating RES. However, it is essential to manage the charge-discharge level of the batteries while considering smoother long-term efficiency and the reliability of the devices.

1.1 Existing Literature on Deep Reinforcement Learning in Microgrids

RL is a popular control technique used in power and energy engineering applications to execute time-sequential decisions under dynamic environments (Glavic et al., 2017). The design of the optimal policies in RL mainly depends on the system states and rewards by executing the actions, thus reducing the dependency on the system model. The major advantage of RL based scheduling strategy is that it formulates optimal policy just considering the input data without explicitly requiring a model of the system. For a discrete state-space environment, the policies are represented from the values corresponding to a state-action pair in a *Q*-table. The real-time control of the battery energy storage using reinforcement learning has been studied in (Abedi et al., 2022; Kolodziejczyk et al., 2021; Zhou et al., 2022; Zhuang et al., 2022). The review of the reinforcement learning methods have been studied in (Subramanya et al., 2022).

The RL algorithms function as a supervisory algorithm that furnishes the requisite instructions for generating Pulse Width Modulation (PWM) signals used in grid-connected inverters. These inverters possess bidirectional power exchange capabilities, enabling them to regulate the direction and magnitude of power flow between the battery and the utility grid. To develop an optimal policy, the RL algorithm undergoes initial training with a dataset. In practical scenarios, the RL algorithm extracts actions from the trained optimal policy based on real-world sensor readings. These actions are then used to govern the operations of the inverter, thereby managing the charging and discharging processes of the battery.

The deep RL algorithm formulates the policies without forecasting the uncertain renewable energy and energy demand. Other variants of deep *Q*-learning that have been implemented for microgrids include proximal policy optimization (Guo et al., 2022), A3C methods (Hua et al., 2019), novel variation in A3C and PPO (Nakabi and Toivanen, 2021), branching-duelling DQN (Shuai et al., 2021), prioritized replay (Song et al., 2021), finite horizon deep deterministic policy gradient, and finite horizon recurrent deterministic policy gradient (Lei et al., 2021). The RL states are obtained from the time series electricity price using a representation network in (Wan et al., 2019). The rainbow algorithm, which utilizes the concept of distributional reinforcement learning, is deployed for microgrid management in (Harrold et al., 2022). Deep deterministic policy gradient algorithm for analyzing the market equilibrium and simulating different bidding levels of the agents is proposed in (Liang et al., 2020). Monte Carlo tree search has been formulated with distributed BESS in (Shuai and He, 2021). A detailed description of the algorithms and their benchmarked version and various system configurations are provided in Table 1.

Table 1: Comparison of the existing deep RL algorithms for BESS in grid connected microgrids. The bold ones are the best algorithms obtained from the given papers using diverse set of criteria.

Reference	System Configuration	Benchmarked Deep RL Algorithms	Reward/Penalty Functions
(Nakabi and Toivanen, 2021)	Microgrid with price responsive loads and batteries	Value-based Deep RL (DQN, double DQN), Policy Iteration based Deep RL (REINFORCE, Actor-Critic, A3C, PPO) and **Novel Variants of A3C and PPO**	Net profit (i.e. from power import and export)
(Zhang et al., 2021)	DISCO and Demand Response Aggregator	DQN, **Dueling DQN**, DDQN, Actor-Critic, Policy Gradient.	Trading profit
(Shuai et al., 2021)	Distributed energy storage with microgrid with controllable generators	**Branching Dueling Q-Network**, MuZero, Lyapunov Optimization, Approximate Dynamic Programming, DDPG, Mixed Integer Second Order Cone Programming.	RES curtailment cost, exchange cost between microgrid and utility and degradation cost.
(Lei et al., 2021)	Isolated microgrid with load bank, diesel generator, battery	**FH-DDPG, FH-RDPG**, DDPG, RDPG.	DG cost optimization along with the cost of wasted power with system constraints
(Song et al., 2021)	Community energy storage consisting of batteries, supercapacitor banks with grid-connected microgrids, BESS, DG and distributed generators	**Prioritized Replay Dueling DDQN**, DQN, double DQN, duelling DQN.	Net profit (i.e. from power import and export)
(Guo et al., 2022)	PV, Wind Generator, Load, Electricity Price and ESS	**PPO**, DDPG, DQN, Stochastic Optimization	Operating cost of the microgrid and ESS, net cost after trading power. Extra reward for constraint satisfaction. Penalty for violation of SoC.
(Hua et al., 2019)	Fuel cell, Wind power, microturbine, load, ESS, PV units.	**Actor-Critic** methods with OPF	Power flow between the subgrids, the cost for optimal generator utilization, cost function for the

			SoC functioning around a specific value.
(Wan et al., 2019)	Grid-connected EV charging and discharging	**Representation network with DQNQ**, MPC, MPC + LSTM, MPC + NN, Day ahead, FQI, Uncontrolled.	Net energy cost (after import and export), battery degradation cost and penalty on uncharged energy.
(Harrold et al., 2022)	ESS, PV, Wind Turbine, Microgrid	**Rainbow**, actor-critic, linear programming with discrete optimization. DQN, DDQN, D3QN, PER-DQN, Multi-step DQN, Noisy net DQN, Categorical DQN, Rainbow DQN,	Net energy cost.
(Totaro et al., 2021)	PV, Generator Set, Loads, Energy storage device	**PPO and Quantile Regression RL,** Rule-Based Controller and Optimization-based Look Ahead Controller	RES Curtailment, Fuel Cost, Cost of shedding non-flexible loads.
This Paper	ESS, Grid, Electricity Price, PV	Prioritized Replay Based Reinforcement Learning Variant - DQN**,** C51**, QR-DQN, FQF, Rainbow, IQN**	Net energy cost, penalty on SoC violation.

1.2 Application of Distributional Reinforcement Learning

The variants of distributional RL for power and energy engineering applications have been discussed in (Harrold et al., 2022; Totaro et al., 2021). Lifelong control of the off grid-microgrids is implemented using quantile regression (Totaro et al., 2021) while considering progressive changes in demand and abrupt failure of the energy storage devices. The algorithm is benchmarked against a rule-based look-ahead controller. The steerable generators and energy storage devices act as controllable devices. Similarly, the policies derived from the rainbow algorithm has been utilized for battery-based energy management system in (Harrold et al., 2022). Distributional soft actor-critic methods are utilized for frequency control in power systems operating under emergency conditions (Xie and Sun, 2022). The distribution of the value function is estimated rather than the mean for executing bounded actions in frequency control.

Deep distributional RL has also been introduced in (Min et al., 2019) to reach the given target distance while minimizing unnecessary lane changes. Here the driving on the highway has been modelled in a probabilistic fashion. The system performance is analyzed for distributional variants like C51 and QR-DQN algorithms. A novel framework utilizing a quantile regression-based reinforcement learning approach has been discussed in (Zhang et al., 2022) for an imperfect channel state information. The receivers learn the distribution of the downlink rate while optimizing the infrared reflection coefficient to maximize the downlink capacity. The Huber loss, used as a generic loss function for distributional reinforcement learning algorithms, are utilized in (Fan et al., 2022; Tsantekidis et al., 2021).

1.3 Contribution of This Paper

In this paper, a wide variety of distributional RL algorithms have been utilized for battery management in a residential PV-microgrid setup. The primary focus for energy management is to reduce the net power imported from the grid. Special attention is paid to formulate the penalty function so that the battery does not operate at extreme limits. A high negative penalty is introduced for the agents as per the works in (Bui et al., 2020; Guo et al., 2022). However, redundant rewards are incorporated in the learning process if the agents do not violate any constraints (Guo et al., 2022). A complicated reward function can cause slower convergence in the learning process, which can be avoided using a corrective action (Wan et al., 2019) independent of any external conditions like grid tariff. ToU tariff has been presented in this paper, where reinforcement-learning algorithms have been applied (Ali et al., 2021; Foruzan et al., 2018; Wang et al., 2020) for battery and demand-side management for RES dominated microgrids. The corrective actions in this paper are formulated by leveraging the ToU tariff. Furthermore, in the previous works, the battery operations have not been analyzed at different tariff regimes, which this paper explores.

The focus has been on the prioritized experience replay samples for both deep and distributional deep *Q*-learning algorithms on the algorithmic front. Prioritized experience replay for deep *Q*-learning and its variant algorithms have been deployed for the microgrids (Song et al., 2021). However, prioritized experience replay has not been explored along with distributional RL algorithms used for power engineering applications (Harrold et al., 2022; Totaro et al., 2021; Xie and Sun, 2022). Furthermore, the online learning mechanism (Ali et al., 2021) is not an optimal energy management strategy, as it does not provide enough episodes to learn the new trends in the incoming data, which might lead to the development of suboptimal policies. The training, validation, and testing of the algorithms in this paper have been adopted from (Bui et al., 2020). However, the training and testing dataset was considered for a one-day and two-day interval, respectively. A longer training dataset is needed for optimal training performance, as the agent tends to experience more variety of data patterns. As we compare a variety of distributional reinforcement learning algorithms for microgrid management in (Harrold et al., 2022; Totaro et al., 2021), we see that the system-level analysis like the characteristics of batteries example, voltage, columbic efficiency, and other parameters such as self-sufficiency ratio as

defined in (Tervo et al., 2018) is missing. Hence, system-level analysis and the appropriate battery technology are also explained in this paper. Even though the system has various RES sources in microgrids, as summarized in Table 1, if we consider the net demand as the state in RL, the complexity of the former becomes redundant. The works in (Harrold et al., 2022; Totaro et al., 2021; Xie and Sun, 2022) do not consider the performance of various distributional reinforcement learning algorithms. Even the testing dataset is also not utilized; hence we do not get an idea about the generalization properties of the trained distributional reinforcement learning agents. However, we utilize the trained distributional RL agents in this paper to check their generalized performance on the testing dataset. Hence, we can summarize the key contributions of this paper as:

- Benchmarking different deep distributional and *Q*-learning algorithms with prioritized experience replay to analyze training and validation performance and overall system performance.
- Analysis of the deep distributional and *Q*-learning algorithm learning training performance with varied discretized action space, random experience replay and penalty without ToU induced corrective action.
- Performance analysis of the battery operations at different tariff zones for different deep distributional and *Q*-network algorithms for the testing dataset. The analysis is performed with respect to the charge, discharge and idle states of the battery with respect to different algorithms.
- The implications of the scheduling strategy on the battery performance characteristics have been discussed with respect to different distributional reinforcement learning algorithms.

2. SYSTEM DESCRIPTION

2.1 Overall Schematic Description

The schematic shown in Figure 1 represents the generic implementation of the RL algorithm on a PV-microgrid environment using a deep *Q*-network. The deep-*Q* network is trained from the environment states like the battery state of charge $\left(\mathrm{SoC}_t\right)$, net power import $\left(P_{Net}\right)$ and the tariff $\left(C_{S_t}\right)$. Based on the values of the deep *Q* network at a particular state Φ_t, actions a_{uncor} are initiated based on the exploration and exploitation strategy of the agent. However, the actions a_{uncor} obtained from the deep learning environment might cause the battery to operate at extreme SoC ends. Hence, it has to be corrected using the tariff information τ_{Tar}, in case the $\mathrm{SoC}_t = \mathrm{SoC}_{\min}$ or $\mathrm{SOC}_t = \mathrm{SOC}_{\max}$, while inducing a penalty. The corrected action a_t is initiated, where the environment transitions to state Φ_{t+1} with reward or penalty. Experiences are stored $\left\{\Phi_t, \rho_t, \Phi_{t+1}, a_t\right\}_j$ for a particular sample j in a tuple $e_t = \left\{\Phi_t, \rho_t, \Phi_{t+1}, a_t\right\}_j$ and the collection of the tuples are stored in replay buffer $\mathrm{D} = \left\{e_1, \ldots, e_N\right\}$, where *N* represents the capacity of the replay buffer. The prioritized experience replay algorithm uses p_j and w_j to select and weigh the experience samples from the buffer respectively while minimizing the loss value $\hat{l}_j$ to train the network. The deep *Q*-network is trained to minimize the loss function while increasing the episodic rewards. The process described in Figure 1 is repeated a certain number of times to obtain a different *Q*-network every time it is trained. Hence, Monte Carlo simulations are performed to obtain different trained *Q*-network, tested against new load demand and solar power data to probe deeper into the battery operations. The description of the environment and the states is given in the following subsections.

2.2 Input Dataset

The grid-connected microgrid consists of solar power as a RES, load demand and BESS. The load demand is primarily met by the power generated from PV and battery, and the remaining to be imported from the grid when these sources are not available. The charging and discharging power commands for the BESS is determined by the EMS. ToU tariff (Newsham and Bowker, 2010) has been used, which captures the general trend of load demand variation throughout the day. The purpose of this tariff is to encourage the consumers to shift their energy demand. The tariff data used in this paper is Green UK's tariff policy ("Green Energy UK Tariff").

The load demand and solar power data are obtained by scaling the UK's consumption and generation profile thus making it realistic in a microgrid case ("GB National Grid Status"). The sampling rate of the data is 5 minutes; hence it has to be averaged to formulate control commands at a longer time interval. The average PV power generation across the UK is considerably smaller than the load demand. Hence it is increased 15 times the mean so that the profile closely resembles that of the microgrid utilizing local PV generation to simulate a scenario with high PV installation capacity. The load demand is capped to a maximum value of 10 kW. The load demand, tariff and PV generation profiles used in this paper are shown in Figure 2.

As observed in Figure 2, three tariff slabs are used, peak, mid-peak and off-peak, as given in ("Green Energy UK Tariff"). The training dataset has been considered for 7 consecutive days, and the validation dataset has been considered for the next 3 days. The starting date for the data set is May 1, 2019. Peak tariff is observed when the load demand is higher with a moderate PV generation. Figure 2 represents the mean and the uncertainty in the load demand in the training and testing dataset, which is considered over a 24-hour interval. We observe that the uncertainty in the load demand for both training and testing datasets is more considerable than solar power. The difference equation representing battery operation used for the energy management system is described in the next section.

2.3 Battery Model

The battery storage system $P_b(t)$ represents the battery power at time *t*, where its positive and negative sign signifies charge and discharge operations, respectively. The state of charge represents the percentage of the energy available within the battery. The $\mathrm{SoC}(t)$ or state of charge of the battery is represented by the difference equation as:

$$\mathrm{SoC}(t) = \mathrm{SoC}(t-1) + P_b(t) \cdot \Delta t / E_b, \quad (1)$$

where E_b represents the battery energy capacity and Δt represents the duration of the battery control command. However, the SoC of the battery must stay within the given limits to prevent over-charging and over-discharging. Hence it has constrained with a specific interval as:

$$\mathrm{SoC}_{\min} \le \mathrm{SoC}(t) \le \mathrm{SoC}_{\max} \quad (2)$$

where, $\mathrm{SoC}_{\min}$ and $\mathrm{SoC}_{\max}$ represents the minimum and maximum SoC limit. The rated battery power (P_{bat}) used in this paper is 2 kW with an energy capacity (E_b) 8 kWh. The values of $\mathrm{SoC}_{\min}$ and $\mathrm{SoC}_{\max}$ used in this study are 15% and 100%, respectively. The difference equation (1) is used to describe the next state of the environment after EMS has regulated battery power $P_b(t)$ in the time interval Δt.

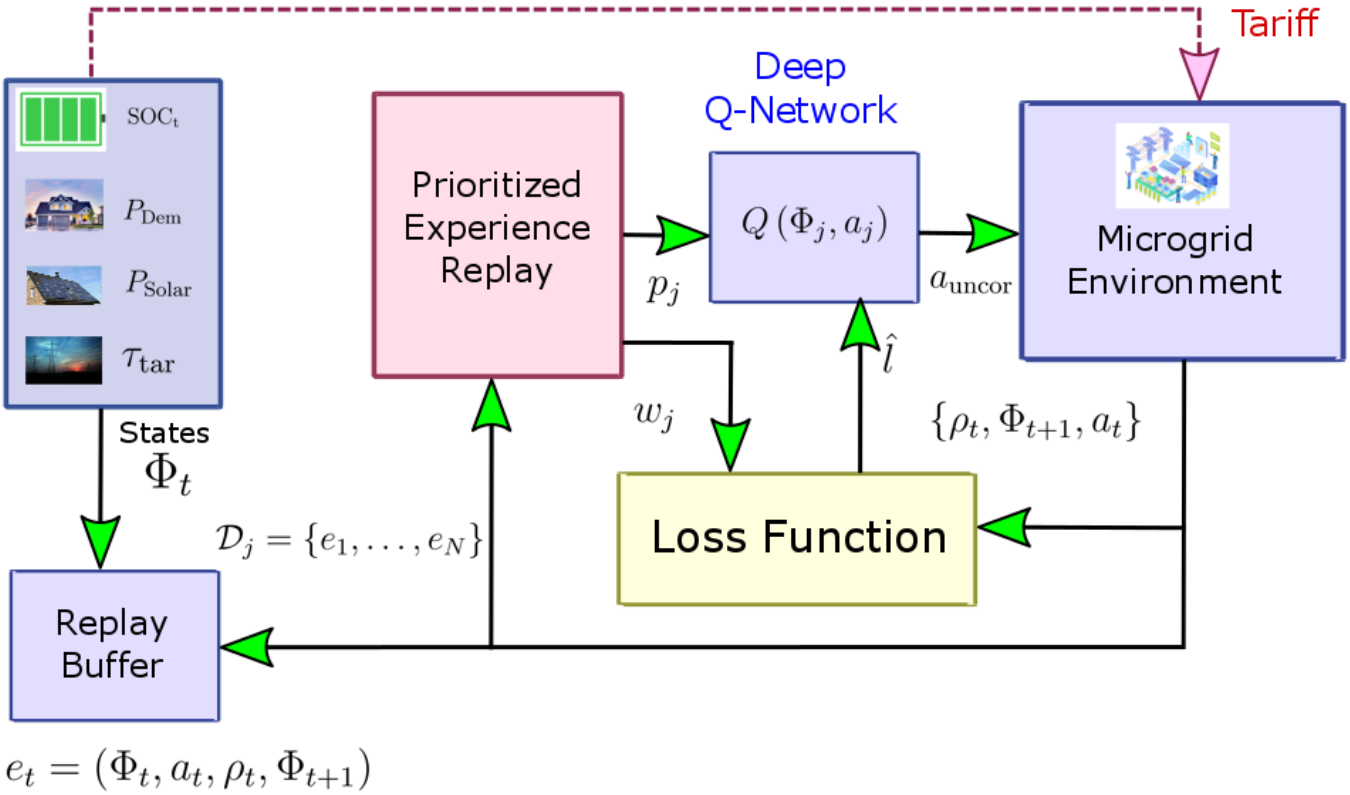

Figure 1: The schematic of the prioritized experience replay based reinforcement learning for battery management in a PV-microgrid environment.

3 Environment Description

In RL, the agent interacts with the environment and takes optimal actions based on the states from the environment to maximize the cumulative rewards obtained over an entire episode. The episode in this paper is considered equal to the length of the dataset used for training which is a span of a day or 24 hours. The states, actions and rewards used in the study are described in the following subsections.

3.1 States

As described in (Mbuwir et al., 2017; Ruelens et al., 2018), the state space of the environment generally consists of three features: timing, controllable, and non-controllable. We define the system state for time *t*, $\forall t \in \{1, 2, \ldots, T\}$, here *T* represents the final time instant. The training and validation dataset is set to 168 steps (7 days) and 72 steps (3 days) per episode, respectively. The timing state component is time-dependent, and it incorporates the environment's state information within the given time. The ToU tariff used in the study consists of three discrete states (see Figure 2):

$$S_t = \{\text{'Off-Peak','Mid-Peak','Peak'}\} \quad (3)$$

The controllable component S_c consists of the state information, measured and controlled via the agent's actions. For this study, SoC is considered as the controllable component. The SoC is controlled using the battery power action $P_b(t)$ at time *t* as defined in (1). However, the controllable states must be constrained within limits (2). The non-controllable component significantly influences the controllable component and the cumulative rewards of the episodes. However, it is not influenced by the control actions from the policies, as it is dependent on external conditions. The net power $P_{Net}(t)$ is represented as:

$$P_{Net}(t) = P_L(t) - P_{PV}(t) \quad (4)$$

where $P_L(t)$ is the load demand and $P_{PV}(t)$ is the PV power. With the availability of different states, it is essential to analyze the agent's actions after interacting with the environment, having the state $\Phi_t = \{S_t, S_c, P_{\text{Net}}(t)\}$.

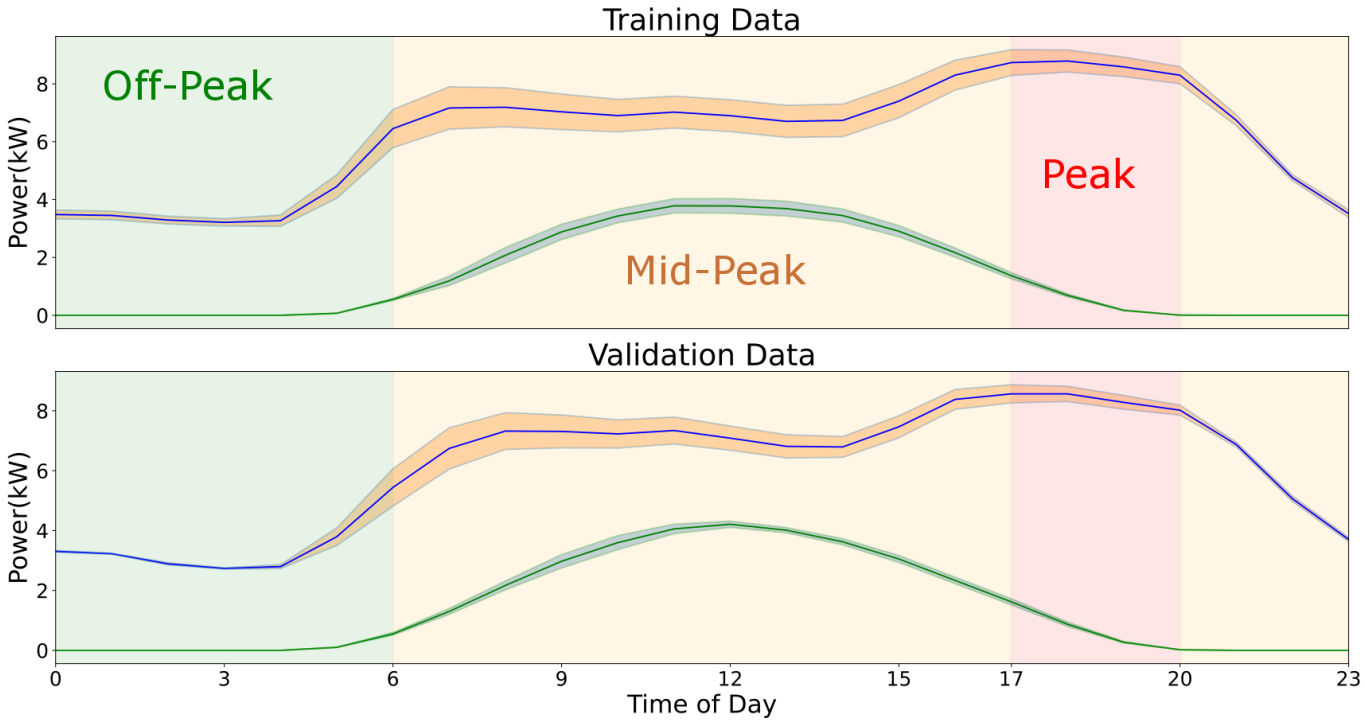

Figure 2: Daily load demand P_{dem} and PV generation P_{PV} and ToU tariff used in the analysis. The green curve shows the PV generation which gives a peak around midday. The yellow curve shows the load which is highest around the evening.

3.2 Actions

Grid-connected batteries represent energy storage systems with the capacity to both absorb and release electrical energy to and from the power grid. They play a pivotal role in contemporary energy management by aiding in the regulation of electricity supply and demand, enhancing the stability of the grid, and promoting the utilization of renewable energy sources. Grid-connected batteries have the capability to be charged directly from the electricity grid. This typically occurs during periods of surplus electricity availability and reduced costs, such as off-peak hours or when there is an excess supply of renewable energy sources like solar or wind power. The process involves the conversion of electrical energy from the grid into chemical energy, which is then stored within the battery. The agent takes action $P_b(t)$, which is a supervisory command to the inverter for regulating the power flow of the battery. It depends on the given state s_t at time *t*. The action space of the agent is discretized into *n* modes. Hence the battery action set $\mathcal{P}$ is defined as:

$$\mathcal{P} = \left\{P_b^{(1)}, \ldots, P_b^{(n)}, 0, -P_b^{(n)}, \ldots, -P_b^{(1)}\right\} \quad (5)$$

where, $P_b^{(i)} = \left[(n-i+1)/n\right] \cdot P_{\text{bat}}$ and *n* is an integer. It is essential that the SoC limit, as defined in (2), is not exceeded. When the action taken from the policy causes the violation of the limits, overriding actions are taken which may be implemented in the schematic diagram shown in Figure 1. The over-ride action depends upon the timing component of the state S_t, i.e. the tariff. The over-ride action, when SoC goes below SoC_{min} is defined as:

$$P_b^{ov}(t)=\begin{cases}0, & \text{if } S_t=\{\text{'Peak'}\}\\ P_{\text{bat}}, & \text{else.}\end{cases} \tag{6}$$

The idle action is preferred during the peak period, which prevents the battery from charging when the tariff is high. Similarly, the over-ride action is implemented when SoC goes beyond $\text{SoC}_{\max}$ defined as:

$$P_b^{ov}(t)=\begin{cases}0, & \text{if } S_t=\{\text{'Off Peak'}\}\\ -P_{\text{bat}}, & \text{else.}\end{cases} \tag{7}$$

Since the battery capacity used here is $P_{\text{bat}}=2$ kW, hence for $n<3$, the battery power $P_b(t)$ will be an integer value. The nominal case is considered to be 2, suggesting integer battery actions. A penalty of higher magnitude than the obtained reward is incurred if the override actions (6)-(7) are initiated. This concept is very similar to the one implemented in (Bui et al., 2020), hover the corrective actions were not based on the grid tariff. The override actions used here is also dependent on the tariff which is also responsible to keep the battery operation within specified SoC limits.

3.3 Rewards

The overall objective of the RL agent is to maximize the reward by exploring the actions space hence converging to an optimal policy. Conventionally the reward is defined as the grid cost, which must be effectively minimized. The reward function is given by:

$$\rho_t(\Phi_t,a_t)=-P_g\cdot C_{S_t}, \tag{8}$$

where, C_{S_t} is the tariff cost at time *t*, and $P_g=P_{Net}(t)-P_b(t)$ is the net power taken from the grid, and a_t represents the action taken by the agent as defined in the previous section. The cost of tariff is similar to the one in (Ali et al., 2021). A negative value maximizes the rewards, thus minimizing the overall grid cost. However, the override actions, (6)-(7) , are sub-optimal since they are not derived from the policy. As given in (Divya and Østergaard, 2009), the battery life depends upon its mode of operation. It should not operate with a high depth of discharge, which means a low state of charge. Moreover, the battery should not be operating near the high state of charge mode, as it acts like a power-consuming source rather than a generation one, hence making the objective futile. Therefore, a high penalty value is used, as derived from (Bui et al., 2020), which is defined as:

$$r_t(\Phi_t,a_t)=-10. \tag{9}$$

The magnitude of penalty for taking sub-optimal action (9) is considered higher compared to the penalty for consuming the power from the grid as defined in (8). The definition of the rewards is analogous to the switched objectives to control the path planning of a mobile robot. It is essential to see how optimal policy is obtained from different RL algorithms described in the next section.

4. DEEP DISTRIBUTIONAL REINFORCEMENT LEARNING ALGORITHMS

In this section, the utilization of deep distributional RL algorithms with prioritized experience replay will be described for the battery management in PV-microgrid. Firstly, we will discuss the deep *Q*-learning algorithms and the requirement of prioritized experience replay. Then we describe the various distributional RL algorithms as a modification of the deep *Q*-learning algorithm.

4.1 Deep Q-Learning Algorithm

The *Q*-network (Mnih et al., 2016) acts as a parameterized function approximation, where the input is the states and the output refers to the *Q*-value concerning a specific action. It addresses the problem of evaluating the *Q*-value of a high dimensional state-space encountered in *Q*-learning (Watkins and Dayan, 1992) and can be adapted for new data (Du and Li, 2020). The temporal difference update for evaluating a policy π for a value function $V(\cdot)$ is given by:

$$\begin{aligned}&V(\Phi)\longleftarrow V(\Phi)+\alpha\left[r+\gamma V(\Phi')-V(\Phi)\right]\\ &a\sim\pi(\cdot|\Phi),\rho\sim R(\Phi,a),\Phi'\sim P(\cdot|\Phi,a)\end{aligned} \tag{10}$$

The value function given in (10) is replaced by *Q*-value based on action-value pairs. The ε-greedy methods select the optimal action based on the *Q*-value obtained. The optimal value function $Q^*(\Phi,a)$ is given as:

$$Q^*(\Phi,a)=\max_{\pi}\mathrm{E}\left[\rho_t+\gamma\rho_{t+1}+\gamma^2\rho_{t+2}+\ldots+\middle|\Phi_t=\Phi,a_t=a,\pi\right] \tag{11}$$

Once the action is executed, the controllable state moves to a new state Φ_{t+1} with a reward or penalty ρ_t. The transition in the environment $e_t=\{\Phi_t,\rho_t,\Phi_{t+1},a_t\}_j$ given in Figure 1. The transition is stored in a replay buffer $\mathrm{D}_j=\{e_1,\ldots,e_t\}$ with a capacity of *N*. The algorithm diverges or becomes unstable when a non-linear approximator is used for state-action value (Tsitsiklis and Van Roy, 1997). The significant reason for the divergence is the correlation between the immediate observations used to train the network. Minor updates in *Q* change the policy and data distribution and the correlation between the action values. Biologically inspired experience replay (McClelland et al., 1995) is utilized to randomize the data and remove the correlation between the

observations. Secondly, the action-value update for the temporal difference target y_j occurs periodically to reduce the correlations. The temporal difference error is given by:

$$\delta_{\mathrm{TD}} = \rho + \gamma \max_{a'} Q\left(\Phi, a'; \theta_i^-\right) - Q\left(\Phi, a; \theta_i\right). \tag{12}$$

Hence the loss function $\hat{l}_j$ is defined to minimize the temporal difference error in (12) maximising the episodic rewards, as shown in Figure 1. The loss function is given as

$$\hat{l}_j\left(\theta_j\right) = \mathrm{E}_{(\Phi_t, \rho_t, \Phi_{t+1}, a_t) \square U(\mathrm{D})}\left[\left(\rho + \gamma \max_{a'} Q\left(\Phi, a'; \theta_i^-\right) - Q\left(\Phi, a; \theta_i\right)\right)^2\right] \tag{13}$$

where γ is the discount factor to determine agents horizon, θ_i are the parameters of the *Q*-network at iteration *I* and θ_i^- are the network parameters used to compute targets at iteration *i*. After every *C* step, the target network parameters θ_i^- are updated with the *Q*-network parameters θ_i. This method to generate the targets improves stability in learning as a clone network is used to generate the temporal difference targets. The deep neural network representing the *Q*-network is updated using the targets obtained from older parameters, making the divergence unlikely. The detailed algorithm for implementing the Deep *Q*-network is described in (Mnih et al., 2015).

4.2 Prioritized Experience Replay

Experience transitions from the replay memory generally utilize the same transitions regardless of their significance. Prioritizing the experience (Liu et al., 2022) helps in the learning process. The TD error (12) represents how far the optimal values from the current state-action pair are from its bootstrap estimate as applicable for online algorithms like TD-learning and SARSA. However, the given method performs poorly when the rewards are noisy. The experience samples for the DQN generally replay low TD-error transitions. Furthermore, the learning process is also sensitive to impulsive spikes to the TD errors. Greedy prioritization focuses only on small subsets of experience, and the errors shrink slowly using function approximation. However, the greedy prioritization of the samples causes a lack of diversity in learning, which results in overfitting. Hence stochastic sampling is used, which balances greedy prioritization and random sampling. We can define the probability of sampling transition *i* as,

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}, \tag{14}$$

where, α represents the prioritization coefficient and $\alpha = 0$ represents the uniform case. We define $p_i = \left|\delta_i\right| + \varepsilon$, where ε presents the edge case transition, in case the TD error becomes zero. Rank-based prioritization involves defining $p_i = \frac{1}{\mathrm{rank}(i)}$ where $\mathrm{rank}(i)$ is the rank of transition *i* when the replay memory is sorted according to $\left|\delta_i\right|$. However, the prioritized replay can introduce a bias, which changes the distribution in an uncontrolled fashion and the convergence of the moments of the probability distribution. This bias can be corrected using importance sampling weights given by:

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^\beta. \tag{15}$$

The sample weights (15) compensate for the non-uniform probabilities $P(i)$ if $\beta = 1$. These weights are incorporated *Q*-learning update by using $w_i \delta_i$. They are normalized by $1/\left[\max_i w_i\right]$ so that the updates are scaled upwards, leading to a stable learning process. The unbiased nature of the update is highly desirable in RL as the process is highly non-stationary due to the variability in policies, state distributions and bootstrap targets. So, the importance sampling update coefficient β is corrected over time, from the initial value of 1. Importance sampling provides a significant advantage when combined with prioritized replay for approximating non-linear functions. Smaller global step size is considered as large steps can be disruptive in the learning process since first-order gradient approximation is only locally reliable. Hence if we combine both approaches, prioritization ensures that the high-error transitions are observed multiple times. At the same time, the importance sampling correction reduces the gradient magnitudes, and the algorithm follows the curvature of the highly non-linear optimization landscapes as the Taylor expansion is re-approximated constantly.

4.3 Deep Distributional Reinforcement Learning

The distributional approach to RL differs from the value-based approach as it is based on the probability distribution of the returned rewards rather than the expectation of the rewards (Bellemare et al., 2017). Here, the reward function is viewed as a random vector $\square \in \mathrm{Z}$, where Z represents the space of value distribution with bounded moments. Hence, if we specify $Q^\pi(\Phi, a)$ as the value function associated with a given policy π, then the corresponding distributional analogue is $Z^\pi(\Phi, a)$. Here Z^π

represents the intrinsic randomness due to the uncontrollable state as described before. The transition operator $P^{\pi}: \mathrm{Z} \to \mathrm{Z}$ is defined as,

$$P^{\pi} Z(\Phi, a) = Z(\Phi', A'), \tag{16}$$

where $\Phi' \sim P(\cdot|\Phi, a)$ and $A' \sim \pi(\cdot|\Phi')$. Let us define the distributional Bellman operator $\mathrm{T}^{\pi}: \mathrm{Z} \to \mathrm{Z}$, as

$$\mathrm{T}^{\pi} Z(\Phi, a) = R(\Phi, a) + \gamma P^{\pi} Z(\Phi, a). \tag{17}$$

Since we are using a distribution instead of a particular measure, hence we need to state the convergence of the distribution $Z(\cdot)$. Hence, we use a metric called the Wasserstein metric (Rösler, 1992), a measure between the cumulative distribution. If we have two Cumulative Distributions Functions (CDF): F and G over the real space, the metric is defined as:

$$\bar{d}_p(F, G) = \inf_{U,V} \|U - V\|_p, \tag{18}$$

where the infimum is obtained over all the pairs of random variables (U, V) having cumulative distribution F and G. The infimum is obtained using inverse CDF transformation of the random variable U, uniformly distributed in [0,1]. Thus, we can write the metric d_p as:

$$\bar{d}_p(F, G) = \left(\int_0^1 \left| F^{-1}(u) - G^{-1}(u) \right|^p du \right)^{1/p} \tag{19}$$

We can consider a process $Y_{k+1} = \mathrm{T}^{\pi} Y_k$, such that $Y_0 \in \mathrm{Z}$. As per Lemma 3 in (Bellemare et al., 2017), T^{π} converges in a strong sense and is a contraction in $\bar{d}_p$, so that all the moments of $\{Y_k\}$ are bounded as it converges to Y^{π}. The advantage of value distribution (17) is threefold. Firstly, it involves the contraction property of the evaluated Bellman operator, which corresponds to the maximal form of the Wasserstein metric $\bar{d}_p$. Secondly, the distribution Bellman operator helps us learn the approximate probability distribution of the values, thus maintaining the multi-modality and leading to stable learning. Thirdly, it also helps to learn from the non-stationary policy. T^{π} is fundamentally similar to the Bellman operator except for the distribution in $\mathrm{T}^{\pi} Z(x, a)$ defines randomness in $R(\Phi, a)$, $P(\cdot)$ and $Z(\Phi', a')$.

4.3.1 Categorical DQN

Proposition 1 in (Bellemare et al., 2017) states that the operator T is not a contraction, even if the optimality operator has a fixed point. Hence, as per Proposition 3, it will not guarantee the convergence of the sequence $\{Z_k\}$ to Z^*. In order to facilitate the learning process, $\{Z_k\}$ has to be projected to an approximate distribution. We parameterize the value distribution $\{Z_k\}$ using a discrete distribution, by $\check{N}$ and $V_{\mathrm{MIN}}, V_{\mathrm{MAX}} \in \mathbb{R}$. The approximate distribution is supported by a set of atoms $\{z_i = V_{\mathrm{MIN}} + i\Delta z : 0 < i < \check{N}\}$, where,

$$\Delta \mathrm{z} = \frac{V_{\mathrm{MAX}} - V_{\mathrm{MIN}}}{\check{N} - 1}. \tag{20}$$

We can consider these atoms as canonical returns of the value distribution. We can define the atom probabilities as $\Theta: \mathcal{X} \times \mathcal{A} \to \mathbb{R}^n$, where,

$$Z_{\Theta}(\Phi, a) = z_i, \text{w.p. } p_i(\Phi, a) := \frac{e^{\Theta_i(\Phi, a)}}{\sum_j e^{\Theta_j(\Phi, a)}}, \tag{21}$$

where $\mathcal{X}$ represents the state space and $\mathcal{A}$ represents action space. One disadvantage of discrete distribution is that Bellman update TZ_{Θ} and Z_{Θ} have disjoint supports. Hence, it is logical to minimize the Wasserstein loss. However, learning from sample transitions is not possible from Wasserstein loss. Hence, we project the Bellman update TZ_{Θ} in support of Z_{Θ}, which reduces Bellman update to multiclass classification. Let us consider π the greedy policy with respect to $\mathrm{E}Z_{\Theta}$ and the sample transition given by (Φ, a, ρ, Φ'). Thus, we can compute the Bellman update as

$$\hat{\mathrm{T}}Z_{\Theta} = \rho + \gamma z_j, \tag{22}$$

for each atom z_j with the probability $\hat{p}_j(\Phi', \pi(\Phi'))$ to the immediate neighbors of $\hat{\mathrm{T}}Z_{\Theta}$. The i^{th} component of the projected update $\Phi \hat{\mathrm{T}}Z_{\Theta}(\Phi, a)$ is given by:

$$\left(\Phi\hat{\mathrm{TZ}}_{\Theta}\left(\Phi,a\right)\right)_i=\sum_{j=0}^{\check{N}-1}\left[1-\frac{\left|\left[\hat{\mathrm{TZ}}_j\right]_{V_{\mathrm{MIN}}}^{V_{\mathrm{MAX}}}\right|}{\Delta z}\right]_0^1\hat{p}_j\left(\Phi',\pi\left(\Phi'\right)\right) \tag{23}$$

Here $\tilde{\Theta}$ represent the parameters of the next state distribution. The Kullback-Leibler (KL) divergence, $D_{KL}\left(\Phi\hat{\mathrm{TZ}}_{\tilde{\Theta}}\left(\Phi,a\right)\middle\|Z_{\Theta}\left(\Phi,a\right)\right)$, representing the sample loss, is minimized using gradient descent. The overall algorithm is called categorical DQN, where the detailed implementation algorithm is explained in (Bellemare et al., 2017). It was observed that few atoms, as defined in (20), lead to poor behaviour. Hence $\check{N}$ =51 is the standard number of atoms used, and the overall algorithm is called C51.

4.3.2 Quantile Regression DQN (QR-DQN)

The C51 algorithm, as described in the previous subsection, models the $Z^{\pi}\left(\Phi,a\right)$ using a discrete distribution at fixed locations $z_1<\ldots<z_n$ as described in (20), which are uniformly spaced within a given interval. Here the given discrete distribution parameters are the probabilities $p_i\left(\Phi,a\right)$ associated with z_i given in (21). However, the major disadvantage of the C51 algorithm is that the approximate projection of the distribution of the returns is not associated with the Wasserstein metric (19) as it does not suffer from disjoint support. In C51, *N* atoms are used for the approximate distribution, and the probabilities are adjusted accordingly. However, in QR-DQN, the location adjustment is stochastic to minimize the Wasserstein distance to the target distribution. C51 does not guarantee the minimization of the *p*-Wasserstein metric.

Let us consider $\mathrm{Z}=\left\{Z:\mathcal{X}\times\mathcal{A}\to\mathcal{P}(\square)\middle|\mathrm{E}\left[\left|Z\left(\Phi,a\right)\right|^p\right]<\infty,\forall\left(\Phi,a\right),p\geq1\right\}$. Hence for two action-value distributions $Z_1,Z_2\in\mathrm{Z}$, consider maximal form of Wasserstein metric as defined in (19) such that,

$$\bar{d}_p\left(Z_1,Z_2\right)=\sup_{\Phi,a}W_p\left(Z_1\left(\Phi,a\right),Z_2\left(\Phi,a\right)\right) \tag{24}$$

Hence as per Lemma 1 in (Dabney et al., 2018b), T^{π} is a γ-contraction for any $Z_1,Z_2\in\mathrm{Z}$, such that

$$\bar{d}_p\left(\mathrm{T}^{\pi}Z_1,\mathrm{T}^{\pi}Z_2\right)\leq\gamma\bar{d}_p\left(Z_1,Z_2\right). \tag{25}$$

Even though $\bar{d}_p$ is an useful metric to understand distributional reinforcement learning behaviour, we cannot minimize this metric using gradient descent techniques. However, quantile regression can be utilized to minimize the Wasserstein metric approximately. Unlike the C51 algorithm, the probabilities have fixed weight $p_i=\frac{1}{\check{N}}$ at various locations for $i=1,\ldots,\check{N}$, and the approximate target distribution quantile estimates are called quantile distribution. Let us consider probabilities associated with each distribution given by $\tau_1,\ldots,\tau_n$ so that $\tau_i=\frac{1}{\check{N}}$ and $\tau_0=0$ where $\Theta:\mathcal{X}\times\mathcal{A}\to\square^{\check{N}}$ is a parametric model. Let us consider that the quantile distribution $Z_{\Theta}\in\mathrm{Z}_Q$ maps each state-action pair $\left(\Phi,a\right)$ to a probability distribution on $\left\{\Theta_i\left(\Phi,a\right)\right\}$ which is specified as:

$$Z_{\Theta}\left(\Phi,a\right)\coloneqq\frac{1}{\check{N}}\sum_{i=1}^{\check{N}}\delta_{\Theta_i\left(\Phi,a\right)}, \tag{26}$$

where δ_z specifies Dirac delta function at $z\in\square$. The advantage of this parameterization is that the algorithm is not restricted to the bounds at uniform resolution. Hence it leads to accurate predictions. Furthermore, there are no issues regarding projections that can get unwieldy, especially for the C51 algorithm. With these parameterized improvements, the Wasserstein metric can be managed without biased gradients. Function approximation in RL causes instabilities in the learning process (Tsitsiklis and Van Roy, 1997). Here distributional Bellman update is projected into parameterized quantile distribution. Let us consider arbitrary value distribution $Z\in\mathrm{Z}$ onto Z_Q such that:

$$\Pi_{W_1}Z\coloneqq\arg\min_{Z_{\theta}\in\mathrm{Z}_Q}W_1\left(Z,Z_{\Theta}\right). \tag{27}$$

If we consider $\check{\mathrm{Y}}$ be the distribution with the bounded first moment and $\check{\mathrm{U}}$ be the uniform distribution over $\check{N}$ Diracs using $\left\{\Theta_1,\ldots,\Theta_n\right\}$. Then we can consider that,

$$W_1\left(\check{\mathrm{Y}},\check{\mathrm{U}}\right)=\sum_{i=1}^{N}\int_{\tau_{i-1}}^{\tau_i}\left|F^{-1}\left(\omega\right)-\Theta\right|d\omega, \tag{28}$$

where F^{-1} is the inverse cumulative distribution function (CDF). The lemma 2 in (Dabney et al., 2018b) states that for $\tau, \tau' \in [0,1]$ satisfying $\tau < \tau'$, which minimizes $\int_{\tau}^{\tau'} \left| F^{-1}(\omega) - \Theta \right| d\omega$, the parameters Θ are given by:

$$\left\{ \Theta \in \square \,\middle|\, \Theta = F^{-1}\left(\frac{\tau + \tau'}{2} \right) \right\} \tag{29}$$

If the quantile midpoints are denoted by $\hat{\tau}_i = \frac{\tau_i + \tau_{i-1}}{2}$, then as per Lemma 2, the values for $\{\Theta_1, \ldots, \Theta_n\}$ that minimizes $W_1\left(\breve{\mathrm{Y}}, \breve{\mathrm{U}}\right)$ are given by $\Theta_i = F_Y^{-1}(\hat{\tau}_i)$. The quantile regression loss for $\tau \in [0,1]$ is an asymmetric convex loss function that penalizes overestimation errors with weight τ and underestimation errors at $(1-\tau)$. If we have a distribution *Z*, for a given quantile τ, the quantile function given as $F_Z^{-1}(\tau)$ the quantile regression loss that has to be minimized is given by:

$$l_{QR}^{\tau}(\Theta) = \mathrm{E}_{\hat{Z} \square Z}\left[\rho_{\tau}\left(\hat{Z} - \Theta \right) \right] \tag{30}$$

where $\rho_{\tau}(u) = u\left(\tau - \delta_{\{u<0\}}\right), \forall u \in \square$. As per Lemma 2, $\{\Theta_1, \ldots, \Theta_n\}$ which gives an unbiased sample gradient of the loss of $\mathrm{E}_{\hat{Z} \square Z}\left[\rho_{\tau}\left(\hat{Z} - \Theta \right) \right]$ using stochastic gradient descent. However, one issue is that the quantile regression loss value is not smooth at 0, affecting the performance when a non-linear function approximation is used. Hence the quantile loss is modified with the Huber loss, which is given by:

$$l_{\kappa}(u) = \begin{cases} \frac{1}{2}u^2, \text{if } |u| \le \kappa \\ \kappa\left(|u| - \frac{1}{2}\kappa \right), \text{ otherwise} \end{cases} \tag{31}$$

The quantile Huber loss is the asymmetric variant of the Huber loss given by,

$$\rho_{\tau}^{\kappa}(u) = \left| \tau - \delta_{\{u<0\}} \right| \frac{l_{\kappa}(u)}{\kappa}. \tag{32}$$

Thus for $\kappa \to 0$, quantile Huber loss reverts to quantile regression loss. As per proposition 1, let us consider Π_{W_1} be the quantile projection, that gives projections to the state-value distributions. For any value distributions $Z_1, Z_2 \in \mathrm{Z}$, for a MDP with countable action and state spaces, it can be shown that:

$$\bar{d}_{\infty}\left(\Pi_{W_1} \mathrm{T}^{\pi} Z_1, \Pi_{W_2} \mathrm{T}^{\pi} Z_2 \right) \le \gamma \bar{d}_{\infty}\left(Z_1, Z_2 \right). \tag{33}$$

Thus we can say that the projected distribution using the operator $\Pi_W \mathrm{T}^{\pi}$ has a unique fixed point, $\hat{Z}^{\pi}$ and since $\bar{d}_p < \bar{d}_{\infty}$, we can state convergence occurs for all $p \in [1, \infty]$. Quantile regression helps us improve the quantile function estimation for a target distribution Y, using the samples $y \square \mathrm{Y}(x)$ that minimize the loss (30). As per Lemma 2, by estimating the quantile function parameter $\tau \in (0,1)$, we an approximate 1-Wasserstein distance from the original distribution. Hence, the quantile function can be combined with the distributional Bellman operator giving us the target distribution for the quantile regression. We can write the parameter update for the quantile regression temporal difference learning algorithm as a modification of the equation provided in (10), as

$$\begin{aligned} &\Theta_i(\Phi) \longleftarrow\!\!\!\!\!\!\!\!- \Theta_i(\Phi) + \tilde{\alpha}\left[\hat{\tau}_i - \delta_{\{\rho + \gamma z' < \Theta_i(\Phi)\}} \right], \\ &a \square \pi(\cdot|\Phi), \rho \square R(\Phi, a), \Phi' \square P(\cdot|\Phi, a), z' \square Z_{\Theta}(\Phi'), \end{aligned} \tag{34}$$

where Z_{Θ} represents a quantile distribution as given in (26), and $\Theta_i(\Phi)$ is the estimated value of $F_{Z^{\pi}(\Phi)}^{-1}(\hat{\tau}_i)$ at state Φ. $\tilde{\alpha}$ is the learning rate of the algorithm. The update shown in (34) is for every single value of $\hat{\tau}_i$. It is essential to calculate the update for the pair $(\Theta_i(\Phi), \Theta_i(\Phi'))$. The metric contraction properties of the quantile distribution as described in (25) and (33) allow us to use the similar algorithmic deployment as the Deep *Q*-network as we have described before. The differences are the size of the output layer with the dimensions $|\mathcal{A}| \times \breve{N}$, $\breve{N}$ representing the hyper-parameter with a given number of quantile targets. The loss function defined in (13) will be replaced by quantile Huber loss given in (32). Furthermore, RMSProp is used as a gradient descent optimizer for the loss function is instead of Adam. The complete description of the algorithm is given in (Dabney et al., 2018b).

One advantage of QR-DQN over the C51 algorithm is that it does not require projection onto approximate distribution, as the expansion and contraction of the values cover the actual range of return. Moreover, there are a lesser number of hyper-parameters as compared to C51. The number of quantiles *N* controls the resolution of the approximate value distribution. For lower values of $\check{N}$, QR-DQN resembles the DQN algorithm, and as *N* increases, it can estimate the lower and upper quantiles of the distribution. Moreover, it can distinguish the lower probability events.

4.3.3 Implicit Quantile Network (IQN)

IQN utilizes the information provided by the distribution over the returns while considering the "risk" factor in the policies (Dabney et al., 2018a). As per the expected utility theory, if a policy is consistent with the set of axioms regarding its choices, it behaves to maximize the utility function (Von Neumann and Morgenstern, 2007) for a given state-action pair (Φ, a). It is represented as:

$$\pi(\Phi) = \arg\max_a \mathop{\mathrm{E}}_{Z(\Phi,a)}\left[U(z)\right]. \tag{35}$$

The policy with a linear utility function is called risk neural. If we utilize concave/convex utility functions, it gives rise to risk-averse and risk-seeking policies. If we consider two random variables *X* and *Y* with the condition $X \succ Y$, then as per (Yaari, 1987), for any random variable *Z*, for any $\alpha \in [0,1]$, we can say

$$\alpha F_X^{-1} + (1-\alpha) F_Z^{-1} > \alpha F_Y^{-1} + (1-\alpha) F_Z^{-1}. \tag{36}$$

Here $F_X^{-1}, F_Y^{-1}, F_Z^{-1}$ represent quantile functions to random variables *X*, *Y* and *Z*, respectively. The relation (36) leads to an alternate dual theory of choice of expected utility. Under these axioms, decision policies maximize distorted expectation for continuous monotonic function *h*:

$$\pi(\Phi) = \arg\max_a \int_{-\infty}^{\infty} z \frac{\partial}{\partial z}\left(h \circ F_{Z(\Phi,a)}\right)(z)\, dz. \tag{37}$$

Here, *h* is called the distortion risk measure as it distorts the cumulative probability of random variables. Let us consider the distortion risk measure $\beta : [0,1] \to [0,1]$ where identity corresponds to a risk neutrality situation. Then we can consider the distorted expectation of $Z(\Phi, a)$ under β as:

$$Q_\beta(\Phi, a) = \mathop{\mathrm{E}}_{\tau \sim U([0,1])}\left[Z_{\beta(\tau)}(\Phi, a)\right]. \tag{38}$$

The distorted expectation is the expected value of $F^{-1}_{Z(\Phi,a)}$ weighted by β given as $Q_\beta = \int_0^1 F_Z^{-1}(\tau)\, d\beta(\tau)$. It implies that for any β there exists a sampling distribution for τ such that the mean Z_τ is the distorted expectation of *Z* under β. As per (Dhaene et al., 2012), a distorted expectation is written as a sum of quantiles. Let us consider π_β be a risk sensitive greedy policy:

$$\pi_\beta(\Phi) = \arg\max_{a \in \mathrm{A}} Q_\beta(\Phi, a), \tag{39}$$

Let us consider two samples $\tau, \tau' \sim \mathrm{U}([0,1])$ along with the policy π_β, the temporal difference error at step *t*, is given by:

$$\delta_t^{\tau,\tau'} = \rho_t + \gamma Z_{\tau'}\left(\Phi_{t+1}, \pi_\beta(\Phi_{t+1})\right) - Z_\tau(\Phi_t, a_t) \tag{40}$$

The IQN loss function is given as

$$\mathrm{L}(\Phi_t, a_t, \rho_t, \Phi_{t+1}) = \frac{1}{\check{N}'} \sum_{i=1}^{\check{N}} \sum_{j=1}^{\check{N}'} \rho_{\tau_i}^{\kappa}\left(\delta_t^{\tau_i, \tau_j'}\right), \tag{41}$$

where $\check{N}$ and $\check{N}'$ denote the number of IID (independent and identically distributed) samples $\tau, \tau' \sim \mathrm{U}([0,1])$ used to estimate the loss. The sample-based risk-sensitive policy is obtained by approximating Q_β by *K* samples of $\tilde{\tau} \sim \mathrm{U}([0,1])$. We can define the policy $\tilde{\pi}_\beta$ as:

$$\tilde{\pi}_\beta = \arg\max_{a \in \mathrm{A}} \frac{1}{K} \sum_{k=1}^{K} Z_{\beta(\tilde{\tau}_k)}(\Phi, a), \tag{42}$$

where τ, τ' and $\tilde{\tau}$ is sampled from continuous independent distributions. Risk sensitive policies $\tilde{\pi}_\beta$ are also explored, leading to the decorrelation of the TD errors due to the independent sampling of τ, τ'. The policies can also be implemented like DQN described in the previous subsection.

4.3.4 Fully Quantized parametric Function (FQF)

The fully parameterized quantile function proposed in (Yang et al., 2019) consists of two networks. The first is the fractional proposal network to generate quantile fractions for the state-action pair and quantile value network, which converts the probabilities

to quantile values. In FQF, $\check{N}$ adjustable quantile values are obtained for $\check{N}$ quantile fractions to approximate the quantile function. The return distribution is generally approximated by $\check{N}$ Diracs, which is given by:

$$Z_{\tilde{\theta},\tilde{\tau}}(\Phi,a)=\sum_{i=0}^{\check{N}-1}(\tilde{\tau}_{i+1}-\tilde{\tau}_i)\delta_{\tilde{\theta}_i(\Phi,a)}, \tag{43}$$

where δ_z represents a Dirac at $z\in\square,\tilde{\tau}_1,\ldots,\tilde{\tau}_{N-1}$ as described before. The $N-1$ adjustable fractions satisfy the relation $\tilde{\tau}_{i-1}<\tilde{\tau}_i$ such that $\tilde{\tau}_0=0$ and $\tilde{\tau}_N=1$. We have, $F_Z^{-1}(p):=\inf\{z\in\square : p\le F_Z(z)\}$ where p represents quantile function. Based on the quantile distribution, let us consider $\Pi^{\tilde{\theta},\tilde{\tau}}$ be the operator that projects the quantile function into staircase function with the parameters θ,τ where the projected quantile function is given by:

$$\left(F_Z^{-1}\right)^{\tilde{\theta},\tilde{\tau}}(\omega)=\Pi^{\tilde{\theta},\tilde{\tau}}F_Z^{-1}(\omega)=\tilde{\theta}_0+\sum_{i=0}^{N-1}\left(\tilde{\theta}_{i+1}-\tilde{\theta}_i\right)\mathrm{H}_{\tilde{\tau}_{i+1}}(\omega), \tag{44}$$

where H represents the Heaviside step function and $\mathrm{H}_{\tilde{\tau}}(\omega)$ is the notation for $\mathrm{H}(\omega-\tilde{\tau})$. Hence, a fractional set $\tilde{\tau}$ is generated using the fractional proposal network for every state-action pair (Φ,a). We obtain the quantile values $\tilde{\theta}$ corresponding to $\tilde{\tau}$ using the quantile value network. In order to measure the distortion between the approximate quantile function and true quantile function, the 1-Wasserstein metric is used. Proposition 1 in (Yang et al., 2019) suggests that for any continuous quantile function F_Z^{-1} which is non-decreasing, we define the 1-Wasserstein loss F_Z^{-1} and $\left(F_Z^{-1}\right)^{\tilde{\tau}}$ by:

$$W_1(Z,\tilde{\tau})=\sum_{i=0}^{N-1}\int_{\tilde{\tau}_i}^{\tilde{\tau}_{i+1}}\left|F_Z^{-1}(\omega)-F_Z^{-1}(\tilde{\tau}_i)\right|d\omega. \tag{45}$$

Then we can write $\frac{\partial W_1}{\partial\tilde{\tau}_i}$ as:

$$\frac{\partial W_1}{\partial\tilde{\tau}_i}=2F_Z^{-1}(\tau_i)-F_Z^{-1}(\hat{\tau}_i)-F_Z^{-1}(\hat{\tau}_{i-1})\ \forall i\in(0,N). \tag{46}$$

Furthermore $\forall\tilde{\tau}_{i-1},\tilde{\tau}_{i+1}\in[0,1],\tilde{\tau}_{i-1}<\tilde{\tau}_{i+1},\exists\tilde{\tau}_i\in(\tilde{\tau}_{i-1},\tilde{\tau}_{i+1}),s.t.\frac{\partial W_1}{\partial\tilde{\tau}_i}=0$. The given equations provide us with means to minimize W_1 without computing it. If we consider w_1 being the parameters of the fractional proposal network *P*, considering an arbitrary quantile function F_Z^{-1}, we can minimize W_1 by applying gradient descent to w_1 where the convergence is guaranteed. Since the true quantile function F_Z^{-1} is unknown to us, we use F_{Z,w_2}^{-1} where the parameters w_2 for current state and action are known as the quantile function. The expected return for the FQF is given as:

$$Q(\Phi,a)=\sum_{i=0}^{N-1}(\tilde{\tau}_{i+1}-\tilde{\tau}_i)F_{Z,w_2}^{-1}(\tilde{\tau}_i). \tag{47}$$

With adequately chosen probabilities, quantile regression and distributional Bellman update can train the quantile function. If *Z* is the random variable denoting the state-action value at (Φ_t,a_t) and Z' be the action-value random variable at (Φ_{t+1},a_{t+1}) then the temporal difference error for the probabilities $\tilde{\tau}_i$ and $\tilde{\tau}_j$ is defined by:

$$\delta_{ij}^t=\rho_t+\gamma F_{Z',w_1}^{-1}(\tilde{\tau}_i)-F_{Z,w_1}^{-1}(\tilde{\tau}_j). \tag{48}$$

The loss functions are similar to the ones for IQN and QR-DQN. A detailed explanation of the algorithm is given in (Yang et al., 2019).

4.3.5 Rainbow Algorithm

There have been many variants of DQN algorithms. However, it is unclear which of the extensions can be combined to get optimal results. In rainbow-DQN, six different extensions (Hessel et al., 2018) have been examined, and studies have been performed on their combinations. Six different extensions are described as follows:

- Double *Q*-learning: As described in the previous subsection, the conventional deep Q-learning algorithm is generally affected by an overestimation bias due to the maximization step for the loss function (13). The double *Q*-learning (Van Hasselt et al., 2016) addresses this bias by decoupling the maximization in the bootstrap target and selecting the action from the evaluation. Hence, the loss function is modified as:

$$\hat{l}_j(\breve{\theta}_j)=\left(\rho_{t+1}+\gamma_{t+1}q_{\bar{\theta}^-}\left(\Phi_{t+1},\arg\max_{a'}q_{\breve{\theta}^-}(\Phi_{t+1},a')\right)-q_{\breve{\theta}}(\Phi_t,a_t)\right)^2. \tag{49}$$

- Dueling Networks: The dueling network consists of two computational streams, one value and another advantage. It shares a convolutional encoder merged using a particular aggregator (Wang et al., 2016).

$$q_{\tilde{\theta}}(\Phi,a)=v_{\eta}\left(f_{\xi}(\Phi)\right)+\tilde{a}_{\psi}\left(f_{\xi}(\Phi),a\right)-\frac{\sum_{a'}\tilde{a}_{\psi}\left(f_{\xi}(\Phi),a'\right)}{\check{N}_{\text{actions}}}. \quad (50)$$

Here ξ,η,ψ are the parameters of the shared encoders f_{ξ}, value stream v_{η} and advantage stream $\tilde{a}_{\psi}$, and $\theta=\{\xi,\eta,\psi\}$ is considered as the concatenation.

- Multi-step learning: Instead of single-step targets, forward view multi-step target can be used (Sutton, 1988), using truncated $\tilde{n}$-step return from a given state Φ_t given as,

$$R_t^{(\tilde{n})}=\sum_{k=0}^{\tilde{n}-1}\gamma_t^{(k)}\rho_{t+k+1}. \quad (51)$$

The multi-step variant, which leads to faster learning, can be used for the DQN by minimizing the given loss:

$$\hat{l}_j(\theta_j)=\left(R_t^{(\tilde{n})}+\gamma_t^{(\tilde{n})}q_{\theta^-}(\Phi_{t+\tilde{n}},a')-q_{\theta}(\Phi_t,a_t)\right)^2 \quad (52)$$

- Noisy Nets: One issue with the -greedy exploration in RL is that many actions are needed to be implemented before the first reward is collected. Noisy Nets, as discussed in (Hessel et al., 2018), combines deterministic and noisy streams as follows:

$$\mathbf{y}=\left(\mathbf{b}+\mathbf{W}\Phi\right)+\left(\mathbf{b}_{\text{noisy}}\odot\tilde{\varepsilon}^b+\left(\mathbf{W}_{\text{noisy}}\odot\tilde{\varepsilon}^w\right)\Phi\right). \quad (53)$$

Here $\tilde{\varepsilon}^b$ and $\tilde{\varepsilon}^w$ are the random variables and $\odot$ represents element-wise product. Over time, the network will ignore the noisy stream at different state space domains. Integrated agent is formulated to incorporate all the above variants.

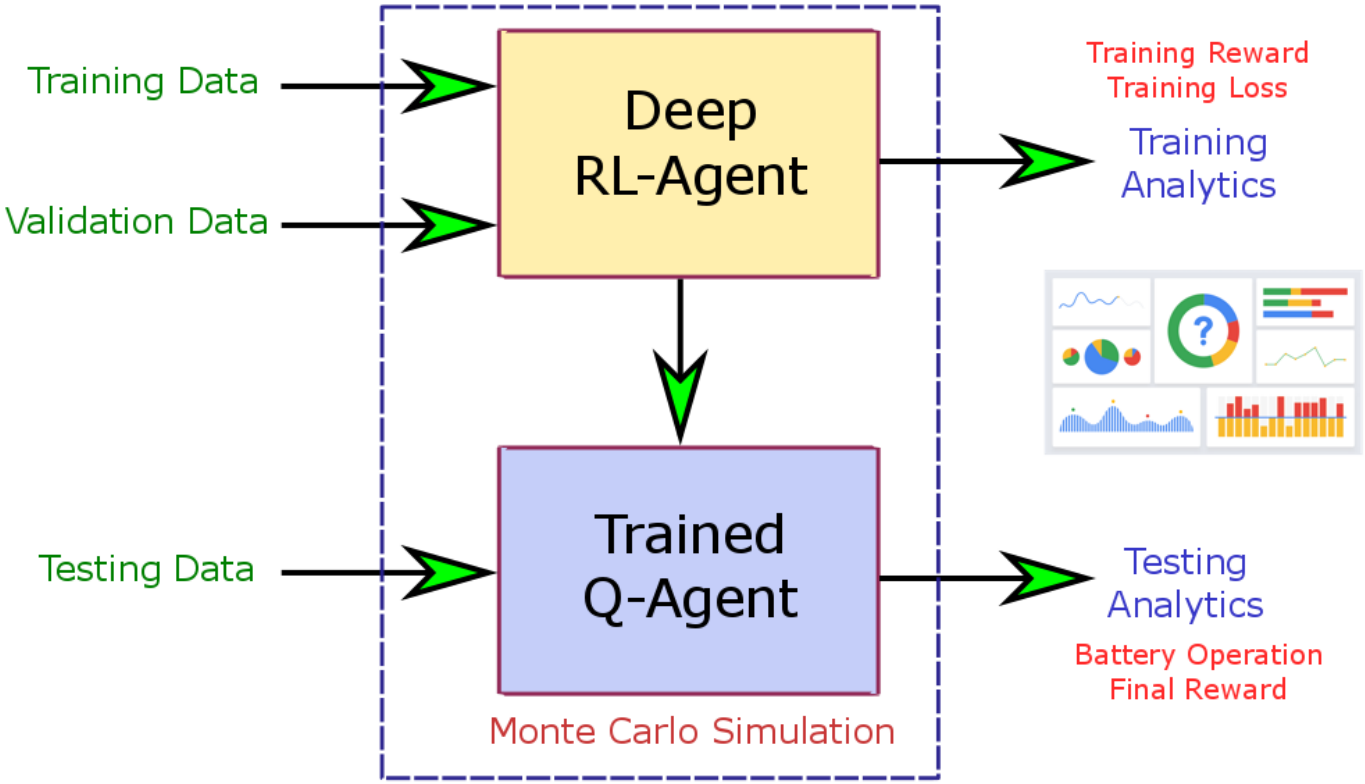

Figure 3: Training, Testing and Validation of the Q-Agent used in this paper. The process is repeated certain times due to the randomness in the algorithm, and final analytics is performed based on the results of the Monte Carlo simulation.

- Integrated Agent:

Step 1: The 1-step distributional loss, defined by Kullback Leibler divergence, along with the multi-step version as similar to multi-step learning as described in the previous subsection, is formulated by constructing the target distribution considering value distribution Φ_{t+n}, with a cumulative discount and *n*-step truncated distribution. Thus, we can represent the target distribution and the loss as:

$$d_t^{(\tilde{n})}=\left(R_t^{(n)}+\gamma_t^{(\tilde{n})}\mathbf{z},\ p_{\tilde{\theta}^-}\left(\Phi_{t+\tilde{n}},a_{t+\tilde{n}}^*\right)\right)$$
$$\hat{l}_j=D_{\text{KL}}\left(\Phi_{\mathbf{z}}d_t^{(\tilde{n})}\middle\|d_t\right) \quad (54)$$

Here Φ_z is the projection onto *z*, which is similar to the distributional reinforcement learning algorithm with a multi-step version. $D_{\mathrm{KL}}(\cdot)$ represents Kullback-Leibler divergence.

Step 2: Multi-step distributional loss is combined with a double *Q*-learning using greedy action for $\Phi_{t+\tilde{n}}$ as selected from the online network considering the bootstrap action $a^*_{t+\tilde{n}}$, and the action is evaluated using target network.

Step 3: Standard proportional prioritized replay is used to prioritize the transitions, computed in the distributional settings using the mean action values. Distributional rainbow variants prioritize the transitions using the KL loss, shown as:

$$\tilde{p}_t = \left(D_{\mathrm{KL}} \left(\Phi_{\mathbf{z}} d_t^{(\tilde{n})} \middle\| d_t \right) \right)^{\omega}. \tag{55}$$

KL loss works perfectly in a noisy stochastic environment because the loss decreases even with non-deterministic returns.

Step 4: Dueling network architecture is used, with the representation network f_ξ being fed into a value stream v_η having $\check{N}_{\text{atoms}}$ with advantage streams $\check{N}_{\text{atoms}} \times \check{N}_{\text{actions}}$. $a^i_\xi\left(f_\xi(s), a\right)$ represents the output to atom *i* and action *a*. Thus, we can say that the value and advantage streams are aggregated as in a dueling DQN and passed through a softmax layer to obtain parametric distribution estimating the return distribution.

$$\tilde{p}^i_\theta(s,a) = \frac{\exp\left(v^i_\eta(\phi) + \tilde{a}^i_\psi(\phi,a) - \bar{a}^i_\psi(s)\right)}{\sum_j \exp\left(v^j_\eta(\phi) + \tilde{a}^j_\psi(\phi,a) - \bar{a}^j_\psi(s)\right)}. \tag{56}$$

where, $\phi = f_\xi(\Phi)$ and $\bar{a}^i_\psi(\Phi) = \frac{1}{\check{N}_{\text{actions}}} \sum_{a'} \tilde{a}^i_\psi(\Phi, a')$

All the linear layers are replaced by their noisy equivalent, where factorized Gaussian noise reduces the number of independent noise variables.

5. Results and Discussions

5.1 Implementation Details

The microgrid environment with the states, actions and rewards, described in Sections 2 and 3, are modelled in the OpenAI Gym environment (Gym, 2016) . The various deep RL agents are modelled in Tianshou (Weng et al., 2022), a RL platform based on PyTorch. The advantage of using this package is that it provides a fast-computational framework and a pythonic PyTorch API. Due to simpler interfaces, it is easier to build and configure RL agents. The solar power and load demand dataset for training, validation, and testing are processed in Pandas and stored separately before accessing the microgrid environment modelled in OpenAI gym. The algorithm was implemented in a 64-bit PC with Intel Xeon CPU at 2.3 GHz.

The Deep RL agent is trained for 2000 episodes from the training and validation dataset shown in Figure 2. The training is performed based on the schematic shown in Figure 1. The trained agent is tested with the testing load demand and solar power dataset, and the battery operation is analyzed. The training, validation and testing operation is repeated 10 times, and we analyze the training reward validation loss and the final battery operation at a different time of the data for different Monte Carlo ensembles. The *Q*-agent is trained with the validation data after every 10$^{\text{th}}$ episode so that the agent can generalize well to the testing dataset, which is a common practice in deep learning. Hence validation loss is obtained only for 2000 episodes. The results obtained from the training, validation and testing process are collected for Monte Carlo simulations and analyzed for different deep and deep distributional RL algorithms. The training process is observed at different action spaces while considering random experience replay and no corrective actions for the penalty. Based on episodic rewards and loss characteristics, the best condition is obtained to evaluate the agent against the testing dataset. The hyper-parameters used for the q-agents are described in the next section.

The training dataset is illustrated in Figure 2, covering a span of 10 days, and comprising 240 hours of data. Given that the agent makes decisions every second regarding the battery, we can assert that it accumulates a sufficient amount of data to converge towards an optimal policy. The efficacy of this optimal policy is then evaluated over a 3-day duration, which is a customary practice as established in prior literature, as indicated in Table 1.

The deep neural networks used to train the transition and minimize the losses have 128 neurons with four deep layers. The learning rate used here is 0.0001. The Adam optimizer is used for varying the learning rate of the deep neural network. The value of γ considered for computing the TD error is 0.9. The TD target to evaluate the deep neural network loss value, as shown in (13) , is updated at the frequency of 320 steps. The buffer size for the experience replay samples has a capacity of 20000 transition samples. The training loss considered for updating the neural network is evaluated after 20 epochs where 10000 steps are taken per epoch. The batch size of 64 transition samples has been considered from the experience replay. The value of $\mathcal{E}$ considered is 0.2 for the episodes less than 500 and decreased to 0.05 to reduce the number of explorations. The training rewards are collected after every episode. The prioritized vector replay buffer has the size of 12000 samples with the value of α = 0.6 and $\beta = 0.4$. The number of forward steps considered for computing the reward is 3. The hyper-parameters and the deep neural network architecture to compute the action are the same as described in the previous section. The number of atoms considered for the discrete distribution is 51, justifying the name of the C51 algorithm. The value of V_{MIN} and V_{MAX} considered here are -10 and 10. In QR-DQN, the number of quantiles considered here is 200. In the case of IQN, the Multi-Layer Perceptron (MLP) is used along with the discrete

implicit quantile network. Firstly, the MLP is formulated with states as input and the hidden sizes described before. An implicit quantile network forms a discrete distribution where the output dimensions are the action space combined with the number of cosines, which is considered 64 in this case. IQN is used for deriving the policies while considering online sample size and target sample size of 8, respectively.

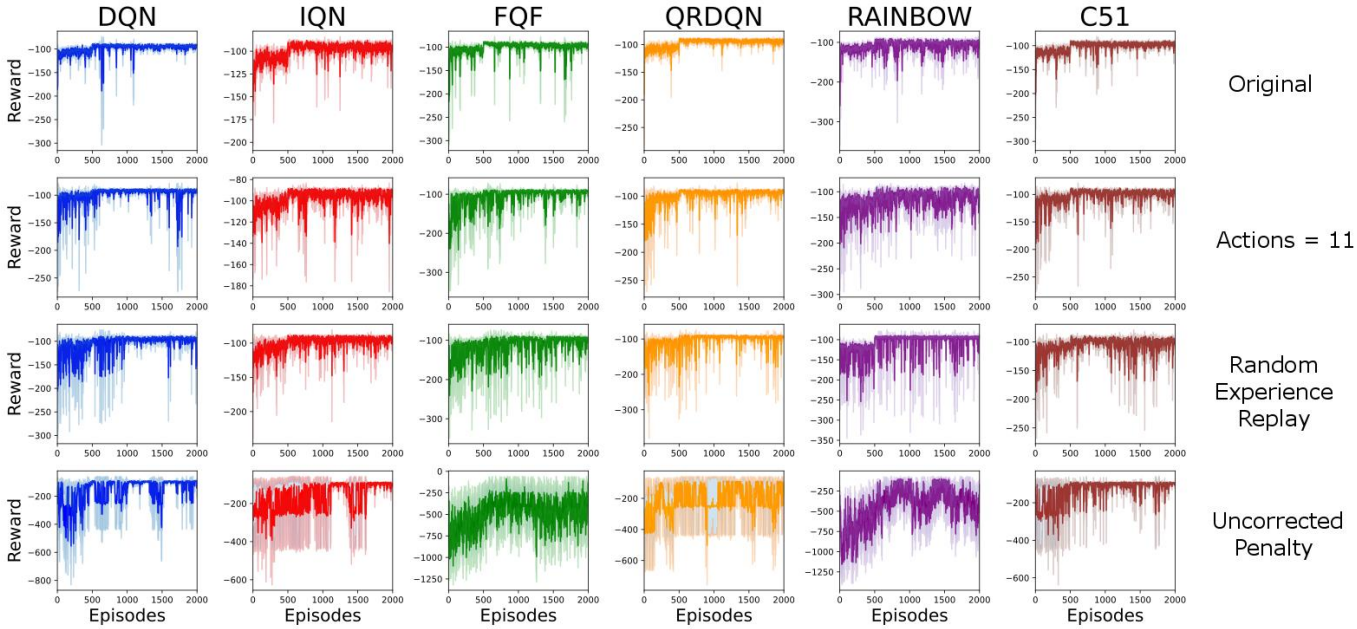

Figure 4: The episodic rewards after every episode of the deep RL and deep distributional RL algorithms were run for 2000 episodes. The simulation is conducted by varying the action space, considering random experience replay and uncorrected action after a penalty. The training occurs for 10 Monte Carlo simulations, and in the figure, we observe the mean and the uncertainty as the shaded area.

In FQF, a fractional proposal network is proposed with the feature network. The feature network is similar to the MLP network as described before, where the weights are obtained using Adam optimizers. Firstly, a full quantile function is formulated similar to the one in QR-DQN. The full quantile function consists of the MLP embedded within it as it depends upon the state space, and the output depends upon the action space. The parameters are obtained using Adam optimizer. Then the fractional proposal network is proposed, where the number of fractions considered are 32, with the same number of input space as that of the quantile function. The weights of the fractional proposal network are obtained using the RMSProp algorithm instead of Adam. The FQF policy is derived while considering the fully quantile function and fractional proposal network. The number of coefficients considered for policy is 10. As we described before rainbow is an integrated agent of different algorithms. Here we add a noisy layer representing the dueling networks with the standard deviation of 0.1. The number of atoms and the values of V_{MIN} and V_{MAX} are considered similar to the one in the C51 algorithm.

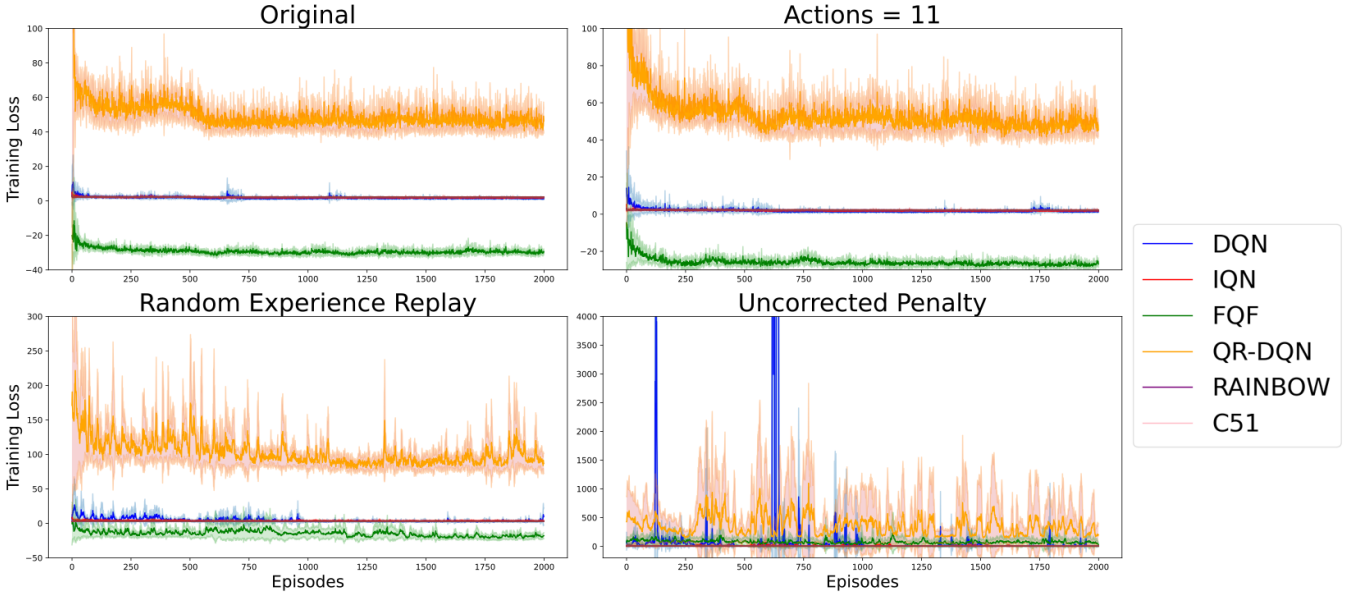

Figure 5a: The agent training loss over the episodes for different distributional RL algorithms and different cases. The results are obtained after the Monte Carlo simulations of the agent, and the results presented here are the mean and standard deviation of the training losses.

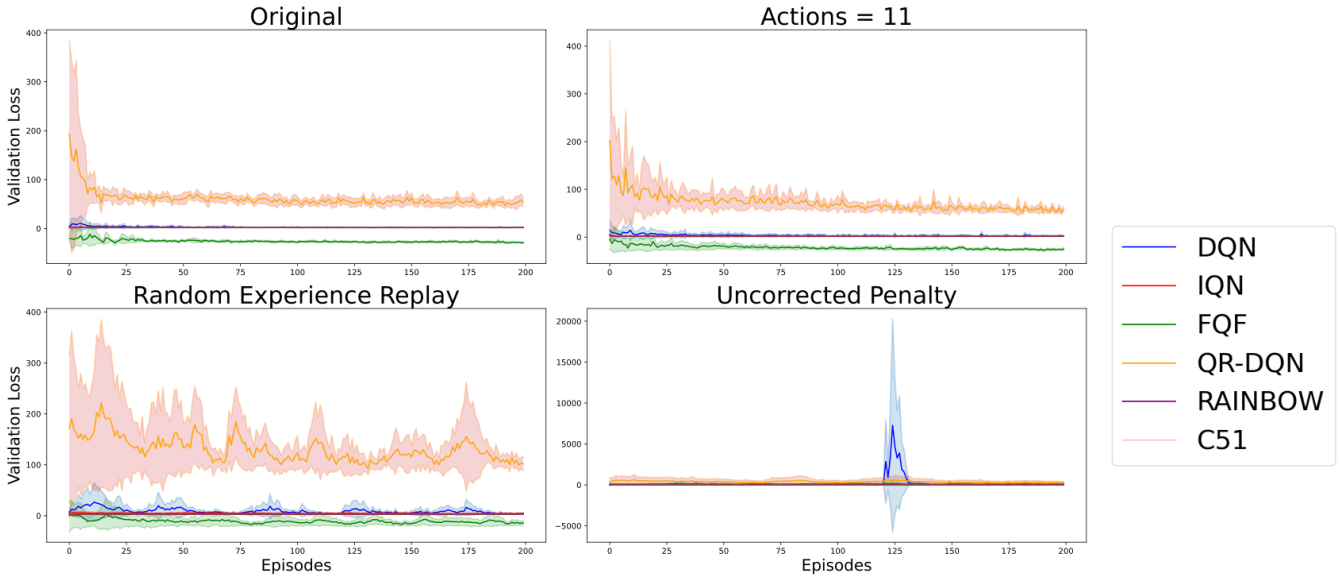

Figure 5b: The agent validation loss over the episodes for different distributional RL algorithms and different cases. The results are obtained after the Monte Carlo simulations of the agent, and the results presented here are the mean and standard deviation of the validation losses.

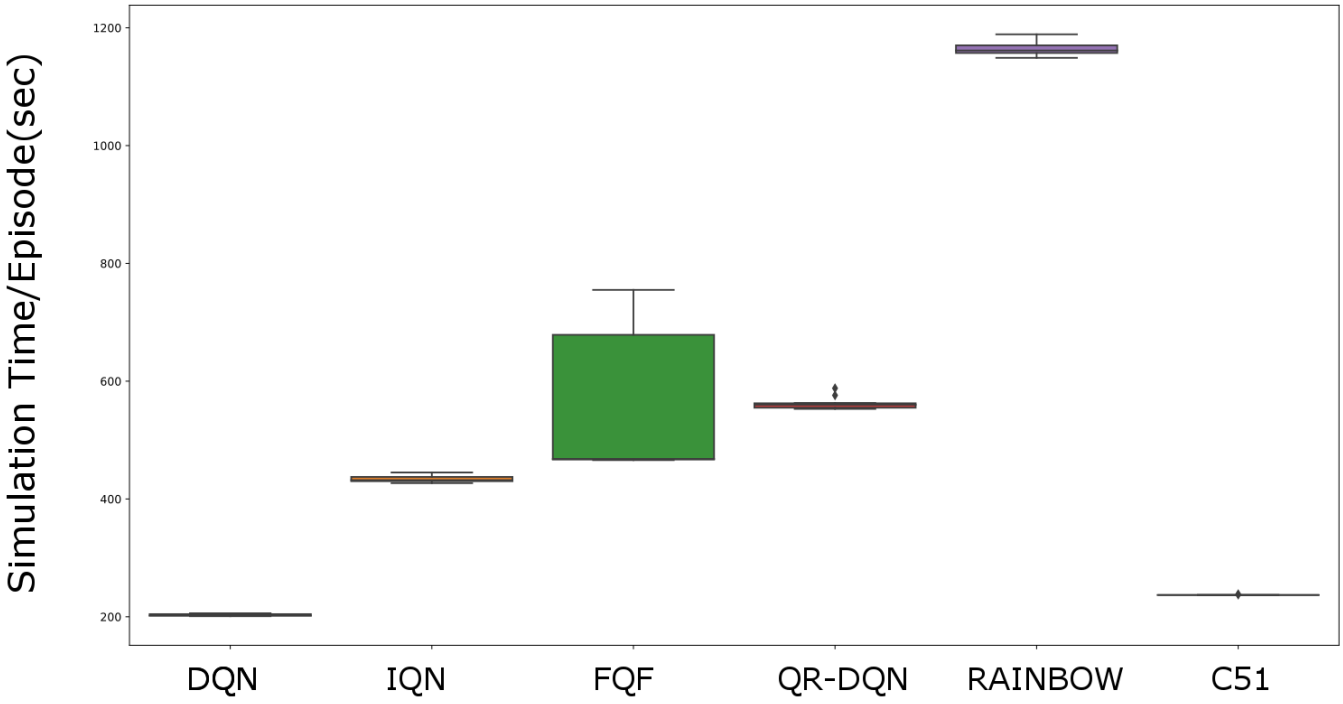

Figure 5c: The boxplot of the training time for different deep and distributional reinforcement learning algorithms when the system is run for 10 Monte Carlo runs for the algorithm running for 2000 episodes.

5.2 Training Reward, Training and Validation Loss

The RL agent is trained with the states as described in Section 3.1, and the 5 different action values are considered for the nominal case. The analysis between the algorithms is also conducted by making the following change in the environment and algorithms:

- Increasing the action space. Thus the value of n considered in this case is 5, resulting in 11 actions viz. 5 charge, 5 discharge and an idle state as given in (5). Thus the value of P_b becomes non-integer, as per the set $\mathcal{P}$ defined in (5).
- Using random experience replay where the batches, sampled from the replay buffer are utilized directly without prioritization in the deep Q-network.
- The corrections of the battery actions, when the SoC operates around the extreme points, are not provided to guide the agent. As observed, override actions are suggested in (6)-(7) based on the tariff. The analysis is performed without the override actions and only provides the penalty for the agent.

The reward convergence plot for the algorithms and cases, as shown in Figure 1, is not appropriately analyzed in the previous distributional RL implementation algorithms like Rainbow (Harrold et al., 2022) and QR-DQN (Totaro et al., 2021). The focus has been more on the system side, not the algorithmic side. Hence, benchmarking the convergence of episodic rewards with the existing distributional RL algorithms for the microgrid is not feasible in this study. However, the reward function incorporating the penalty formulated in this paper is similar to the studies performed in (Bui et al., 2020; Guo et al., 2022; Song et al., 2021). We observe in Figure 1 that the best performing algorithm with the best case takes around 500 episodes to converge. However, comparing the convergence results with the previous studies, we find that the reward takes up around 3000 episodes in (Song et al., 2021) and 35000 episodes in (Bui et al., 2020), although the simulation set-up is different in our case. The reduction in episodes for convergence is mainly due to the override actions based on the tariff in (6)-(7). However, the results are not comparable to the RL in (Guo et al., 2022) because the redundant reward functions like a reward for constraint satisfaction. The redundant reward function defeats the whole concept of the penalty function for SoC override.

The algorithm has the optimum performance in episodic rewards in fewer deviations when $n = 2$, leading to 5 action spaces. The convergence characteristics are the poorest when the experience replay is random. The characteristics show that for random experience replay and uncorrected penalty, more episodes will be required to reach the convergence. Hence we can say environments with prioritized experience replay and corrected actions with induced causes faster convergence, which is evident from the training and validation plot in Figure 5a and Figure 5b. The training and the validation loss value are very unstable in the uncorrected penalty case, as it suggests that it will need much more episodes for convergence. In the case of random experience replay, the training and validation loss is more stable than the uncorrected penalty in the environment. It suggests that the prioritized experience replay can help reduce the variance of the reward distribution as it focuses on sampling specific experiences for training the agent. However, if we compare the convergence characteristics for different action spaces, we observe that the variance of the losses along with the episodic rewards are lower when the action space is lower. The lower variance in the reward can be explained because lower action space causes agents to choose the same actions repeatedly with different Monte Carlo ensembles.

Furthermore, the convergence characteristics are similar as more penalties are observed when the larger action space. We also observe similar characteristics for the loss obtained by validating the agent, as shown in Figure 5b. We observe the higher variance in the loss function for random experience replay and unstable loss in case of an uncorrected penalty. When the action space is smaller, a lower amount of variance is observed for loss validation loss for Monte Carlo ensembles.

If we compare the various distributional reinforcement learning algorithms in Figure 4, the QRDQN algorithm performs optimally out of all the algorithms with minimal penalties observed. The optimal performance is observed due to the projection of the reward distribution as quantile distribution (27), which causes the algorithm to converge faster while experiencing minimal penalty. Furthermore, the contraction property of the operator towards the Wasserstein metric causes the algorithm to converge faster than other variants. However, the worst-performing algorithm based on the training convergence is the RAINBOW and IQN, which fared even worse than the conventional deep Q-learning algorithm. The performance may be attributed due to the integrated nature of the agents in RAINBOW algorithm, which loses its generality when it faces the real-world dataset. As far as IQN is concerned, the distorted expectation for the risk-seeking policies as given in (37) is redundant for this application.

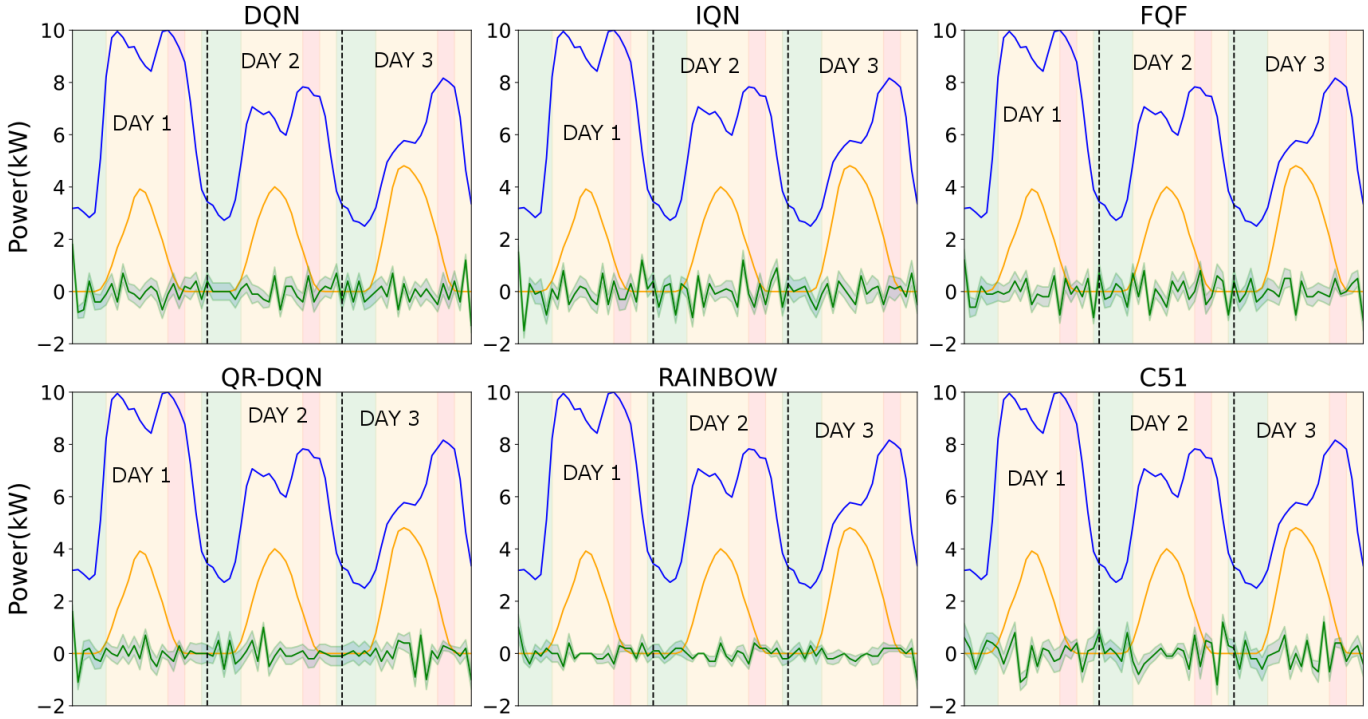

Figure 6: The battery power operation (shown in green) for the testing load demand P_{dem} (shown in blue) and solar power dataset P_{PV} (shown in orange). The battery operation value mean and uncertainty is obtained after 10 Monte Carlo ensembles. The analysis of the results was conducted over a 72-hour testing period,

but the data was compressed into a 24-hour timeframe to evaluate the operational variations spanning the three-day duration. The results are analysed over 10 Monte Carlo runs, hence we are considering the uncertainty in every hour, by considering 30 samples.

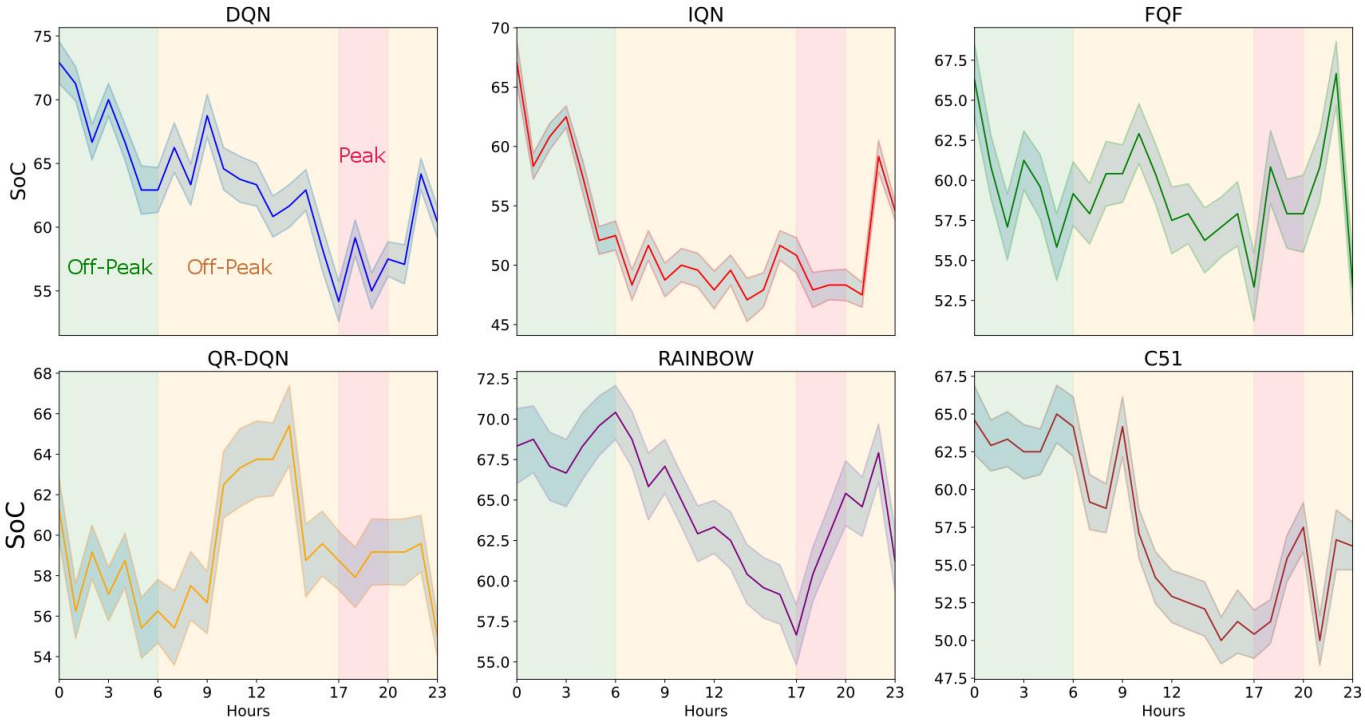

Figure 7: The state of charge (% SoC) of the battery during different time of the day for different deep and distributional RL algorithms for the. The results are analyzed over 24-hour period even though the testing period is for 72 hours, so that the operational nature can be analyzed for different ToU tariff schemes and hence the uncertainty in operation is analyzed accordingly. The results are obtained after 10 Monte Carlo ensemble runs with the testing load demand and solar power data set as per the scheme shown in Figure 3.

We also see that episodic FQF penalties are higher even after convergence is achieved. The penalties are attributed to approximating the quantile distribution using a staircase distribution while deploying the Heavyside step function in (44). While comparing the timings as per Figure 5c, we observe that DQN and C51 algorithms take the lowest and the RAINBOW algorithms take maximum time. DQN, C51 and rainbow algorithms have simpler loss functions to compute. However, due to the presence of different agents, we find the training time of rainbow algorithms is much higher than DQN and C51. We also observe that the mean time for IQN, FQF and QR-DQN algorithms are similar. For the FQF algorithm, the variance of the simulation time is higher with Monte Carlo ensembles. The higher variance can be attributed due to the poor approximation of the quantile distribution using staircase functions. Thus, we can conclude that the type of loss function plays an essential role in the algorithmic convergence and overall training time.

If we compare the training and testing losses amongst the algorithms as shown in Figure 5a and Figure 5b, we find that the training and validation losses are converging to a higher positive value for QR-DQN as compared to IQN and FQF even though they employ the same Huber loss function as given in (32). The higher value is attributed because IQN and FQF distort the policies using a risk factor and a staircase function, respectively. The staircase function approximation causes the loss value negative and the distorted expectation value in (37) , which causes the loss function to converge close to 0. The loss functions for the C51 and RAINBOW algorithm are similar because Kullback Leibler divergence is utilized while computing the loss function. Hence the loss function is very similar to the loss function we experience for the DQN algorithms, representing the loss over the epochs for a feedforward neural network.

5.3 Battery Operations on the Testing Dataset

After the deep *Q*-network training, we test the trained model with the solar power and load demand dataset which is considered for 3 days as shown in Figure 3 while analyzing the system characteristics like battery operation, and agent rewards in the given scenario. The analytics have been shown in Figure 6, Figure 7, Figure 8a and 8b. We observe the results of battery operation in Figure 6 for the different randomized trained *Q*-network for the same input dataset. The testing results are analyzed for the best-trained model, i.e. with integer battery actions, prioritized experience replay and corrected actions with penalized rewards, as shown in Figure 6.

We utilize specific metrics to analyze the results as per the metrics in the residential photovoltaic analysis (Tervo et al., 2018), such as self-sufficiency ratio (SSR), which represents the percentage of the residential load met by PV and battery systems. As we observe in Figure 6, the algorithms that did not perform optimally during training, i.e. IQN, FQF, rainbow, operate the battery in charging mode during the sunny intervals. However, the rainbow algorithm acts more conservatively than the other two, where the

battery power is fluctuating within a narrow band most of the time. However, for the algorithms like QR-DQN, DQN and C51, where the training performance was good, the battery operates in both charge and discharge mode during the sunny period. We also observe that QR-DQN operation is more conservative than DQN and C51.

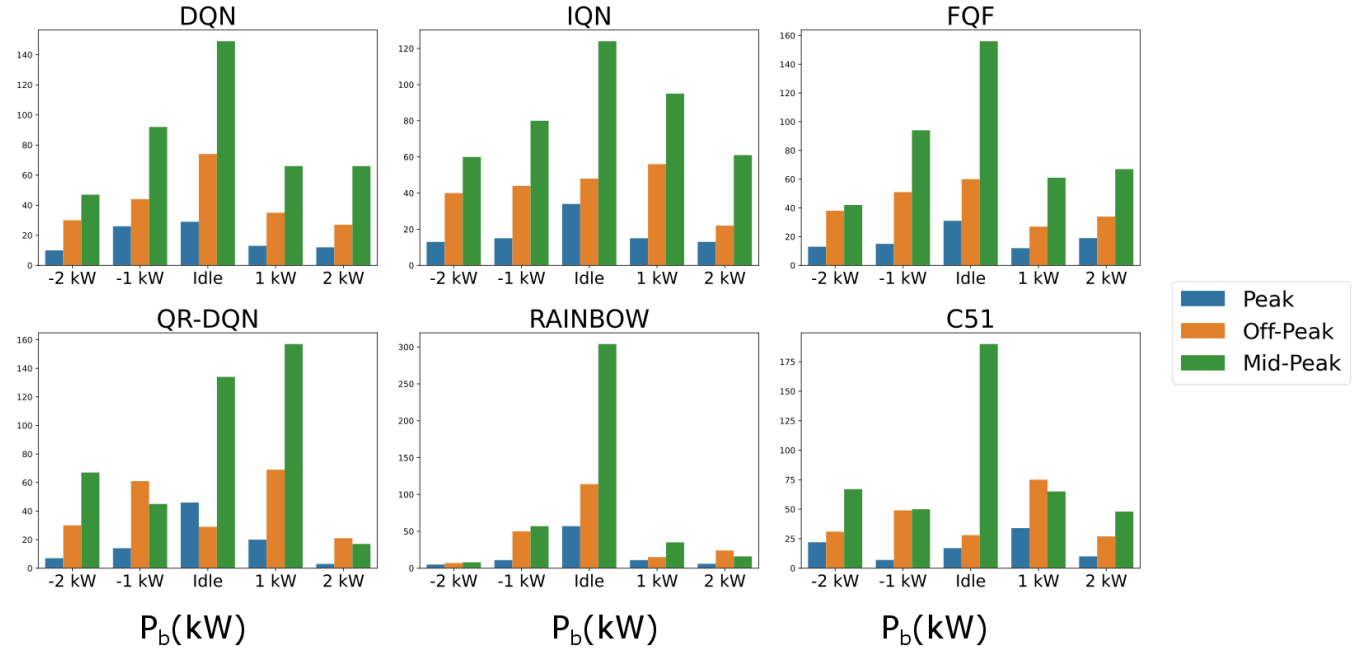

Figure 8a: The bar plot of the frequency of battery operations with 5 action spaces during different tariff periods for deep and distributional reinforcement learning algorithms obtained after 10 ensemble runs with the testing load demand and solar power dataset.

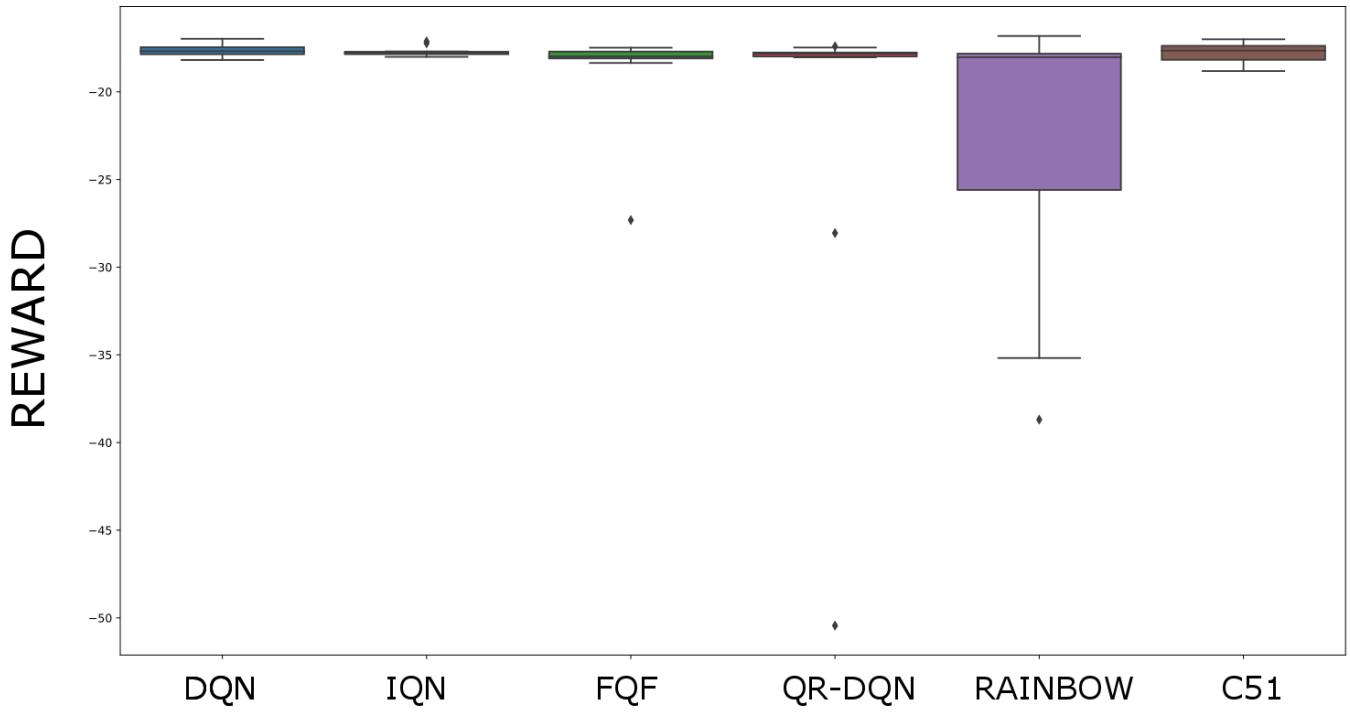

Figure 8b: The box plot of the cumulative rewards obtained on the testing dataset for different deep and distributional RL algorithms obtained after 10 Monte Carlo ensemble runs.

We observe in Figure 6 the battery power uncertainty is lower when operated under the trained rainbow agent algorithm and very high for the IQN algorithm. We observe during the peak period that QR-DQN, which performs optimally during the training period, conservatively operates the battery. Battery SoC throughout the day is shown in Figure 7, which is averaged over the 24 hours. We observe that in some cases like the DQN and IQN, the state of charge shows a discharging trend during the peak period in the evening. The battery primarily charges up when there is solar power, except for the QR-DQN algorithm. The battery

operation hovers around 50 per cent SoC for the IQN algorithm when the solar power is present during the mid-peak period. Other than the DQN and FQF algorithm, the battery operation remains either stagnant or in charge mode during the peak period. Even though the training performance is optimal for the QR-DQN algorithm, the SoC remains relatively constant during the peak period. The battery operation shows that good training performance leads to conservative operation of the battery on the testing dataset. FQF shows good discharge performance during the peak period, suggesting that the same battery operation can act as a peak demand shaving device without any change in the cost function.

While the battery profiles may appear visually similar, there are distinctions in the operational variability among different distributional RL approaches. To provide a more quantitative assessment, we have incorporated the numerical metrics from Figure 6 into Figure 8 in the form of a bar chart. This representation highlights the variations in battery operations, enabling us to make informed observations about diverse battery utilization patterns.

Figure 6 provides an intuitive representation of the operational attributes of the battery at different times throughout the day. The graph presents the average performance and the degree of variability in battery operations. It is apparent from this visualization that the employment of RAINBOW algorithm results in the least uncertainty and minimal fluctuations in battery operations. However, for a more detailed examination of the battery's performance, we have included evaluation indicators in Figure 8a. These indicators are depicted as a bar plot, illustrating the distribution of charging, discharging, and idle operations at various tariff regimes during the day. We observe that for the mid-peak period, the battery is utilized optimally when the algorithm is DQN, IQN and FQF, as we observe a multimodal distribution structure across all the power values. However, for QR-DQN, the battery is not charged fully during the mid-peak period. We also observe a high peak across the 0kW and 1kW, confirming a conservative operation during the mid-peak period. If we observe the mid-peak pattern for the rainbow algorithm, the battery is unutilized most of the time during the mid-peak period. We also observe a similar phenomenon for the C51 algorithm. However, the utilization rate is better than the rainbow algorithm. The battery is operated uniformly at all power levels under the IQN algorithm. However, we observe discharge operation mainly during the peak period for QR-DQN algorithms and not full charge operation, unlike other algorithms.

As far as reward is concerned, the mean value is relatively similar for all the algorithms as shown in Figure 8b. However, the reward variance varies across different algorithms. The rainbow algorithm has the highest variance, mainly due to a mixture of agents in the algorithm. Even though the IQN training algorithm is not optimal, the variance is the minimum out of all. We also find that QR-DQN, which performs optimally in training, enforces penalties on the testing dataset as observed in the outlier of the reward values. Thus, we can conclude that the best-trained algorithm agent might not perform optimally with the testing data. In the previous literature, more emphasis has been on training performance and converged rewards, not much on the performance on the testing dataset. However, unlike those cases, even though the algorithm performs best on the training dataset, the performance is not optimal for testing ones. The worst performing algorithm in the training dataset is the most trustworthy one for the testing dataset.

5.4 Battery Technology Implications

As far as battery technology is concerned, lead-acid generally dominates due to lower production costs. Lead-acids are of two types: floating valve-regulated lead-acid (FRLA) and open flooded ones. Undercharged condition leads to the sulphation of the negative electrode, where major failure can occur long-term. Hence, if the SoC of the battery is relatively low, then using a lead-acid battery is risky. The SoC of the battery can get low if the decision is taken at lower time intervals.

As far as the Vanadium redox battery is concerned, the efficiency based on known climatic conditions and load requirement is discussed (Guggenberger et al., 2012). Lead-acid batteries contain low energy density even at high mass; however, Vanadium redox batteries are better in terms of high efficiency and scalability, fast response, long life and low maintenance, making them ideal for power smoothing for energy systems having stochastic fluctuations. The performance characterization of Vanadium redox batteries in terms of discharge and output power efficiency is discussed in (Nguyen et al., 2014). We observe that the discharge efficiency is relatively low when underutilized, and it gets better with a discharge power of 1 kW. The battery operation for the rainbow algorithm in Figure 8 suggests that it is largely underutilized. Hence, the discharge efficiency of the Vanadium redox battery will be affected when rainbow algorithms are utilized. Similar observations can also be made for output power efficiency. However, the output power efficiency is lower with an increased state of charge operation. As we see in Figure 7, the battery state of charge hovers around 50 when IQN is utilized. As we observe in the exact figure, when operated using rainbow algorithms, the SoC of the battery is relatively high, which is around 70 per cent which can lead to lower voltage and discharge efficiency.

Thus, we can say that the integrated agent in the rainbow is not optimal for battery operation in the long run.The output power efficiency for the Vanadium redox batteries will be higher when IQN is utilized, which is contrary to the training performance we have observed in Figure 4, based on episodic rewards, where IQN was not the best algorithm. Hence, in terms of battery operating characteristics, the training performance of the algorithms has a minor role in terms of long-term operating characteristics. The IQN algorithm has the component of modifying the policies to incorporate risk-seeking behaviour. The behaviour is also responsible for the optimal battery operation, and rewards on the testing dataset observed so far. In (McKenna et al., 2013) , we have seen that the Coulombic efficiency is high for SoC below 70%, and it falls off faster with time. Based on the SoC characteristics of the battery for different algorithms in Figure 7, we say that the SoC is kept below 70%. However, for rainbow and DQN algorithms, due to high charging actions during the off-peak period, the SoC goes beyond 70%.

6. CONCLUSION

We have demonstrated the performance of deep and deep distributional reinforcement learning algorithms for battery management in a residential PV residential environment. The deep and distributional reinforcement learning algorithms are trained and validated on training and validation datasets. The training is performed for different algorithms in different environments, which involves varying the action space, uncorrected penalty and random experience replay. The best-trained system is tested on load demand and solar power datasets, where the battery actions are based on its during different tariff periods and the rewards obtained on the testing dataset. We find the following results can be summarized from the given study:

- The convergence performance is better under prioritized experience replay, with corrected penalty and lower action spaces. We obtain a stable loss convergence under the given conditions.
- Out of all algorithms, the QR-DQN algorithm performs optimally during training out of all the given benchmarks, while rainbow and IQN suffer higher penalties with lower convergence rates.
- The rainbow algorithm takes maximum simulation time for convergence, while FQF has the highest variability in simulation time over the ensembles.
- The rainbow algorithm operates the battery conservatively, keeping it idle most of the time on the testing dataset. IQN performs periods. The rainbow algorithm also provides the highest variability in the rewards over the ensembles. Hence, we infer that the algorithm which performs optimally during training and validation might not perform the best on the testing dataset.
- With IQN algorithm the battery can operate at higher voltage while under rainbow and DQN algorithm, the discharge efficiency is higher.

6.1 Methodological Novelty over State of the Art of Automated Microgrid Management

In the realm of traditional RL, the primary objective is to derive a policy that prescribes the best course of action in response to a given state. This policy is designed to establish a mapping between states and actions with the goal of maximizing cumulative rewards. Conventionally, the policy is determined through the utilization of a value function that corresponds to specific state-action pairs. This value function is typically stored in tabular form for discrete state spaces or as a continuous function for continuous state spaces. However, as real-world systems become increasingly intricate, characterizing the values associated with state-action pairs can become a formidable challenge. Consequently, it becomes imperative to extract this knowledge using deep neural networks (DNNs). DNNs enable estimating values for state-action pairs within complex, realistic systems. Furthermore, deep RL effectively addresses the dimensionality problem that arises when working with continuous state spaces. For instance, in our specific study presented in this paper, the battery state of charge and grid demand represent continuous real-world variables. Attempting to derive an optimal policy for these variables using a discretized state representation would have been infeasible.

In the conventional framework of RL, the value function is updated by considering the expected rewards across an entire episode. The distributional approach to RL, on the other hand, diverges from the value-based approach by centering on the probability distribution of the obtained rewards rather than their expected values. This approach has been proved to be especially advantageous for RL algorithms operating within contexts where states exhibit random variations, as is the case in our study involving the unpredictability of solar PV output and load demand. It is worth noting that the fundamental process of calculating rewards remains unchanged; however, the methodology for updating the value function differs significantly. Specifically, our approach incorporates a distributional Bellman operator, in contrast to the typical Bellman operator employed in traditional RL settings.

Also, the distributional reinforcement learning diverges from the traditional approach by focusing on the probability distribution of obtained rewards rather than solely relying on expected reward values. It is important to distinguish distributional reinforcement learning from distributed reinforcement learning, where the computation and updating of value functions are executed in a decentralized manner. These are two distinct methodologies with different principles and objectives.

Moreover, in the traditional approach to optimization, whenever we receive new data regarding demand and solar power, we are required to re-optimize to determine the operational cost. However, when utilizing RL, the process is streamlined. We only need to train the RL algorithm to establish an optimal policy that dictates the charging and discharging actions for the battery in response to incoming data concerning load demand and solar power. Consequently, there is no need for recurrent algorithm re-execution, as the trained RL model autonomously selects optimal actions from the incoming dataset.

In prior research, as indicated in Table 1, the primary focus has largely revolved around assessing the rewards generated by reinforcement learning algorithms during their training phase, aiming to deduce the optimal strategy for managing batteries in microgrids. These studies have predominantly favored the algorithm that yields the highest rewards without comprehensive system-level analysis. However, this paper takes a different approach by incorporating a system-level analysis in the context of both vanilla and distributional deep reinforcement learning algorithms. This analysis encompasses several novel aspects, as demonstrated in Figure 5a. Firstly, it scrutinizes the impact of experience replay, reward structure, and the number of actions during the training phase, a departure from previous literature. Secondly, it evaluates battery charging and discharging performance on the testing dataset, while also considering the analysis in terms of charge and discharge cycles during various tariff periods.

The findings indicate that, although the training performance of vanilla and distributional RL appears similar, the QR-DQN (Quantile Regression) within the distributional RL framework outperforms the traditional vanilla DQN algorithm in the testing dataset. It effectively optimizes battery charging during periods of solar power abundance and demonstrates discharge operations during peak demand periods, as evidenced in Figure 7. Consequently, the distributional RL approach, specifically the QR-DQN aspect, surpasses the traditional vanilla DQN in terms of battery management performance.

6.2 Scope of Future Research

This work can also be extended to incorporate multiple battery energy systems operated using multi-agent RL algorithms to satisfy various business and operational constraints. The operations related to the battery can be substituted with actions aimed at managing the power consumption of appliances, with the goal of reducing the consumer's utility expenses. This approach allows for the application of RL in the context of demand response within the solar microgrid.

This current research can also be extended to encompass a community microgrid, equipped with its own battery energy storage, the problem can readily be scaled up to accommodate a multi-agent environment. In this setup, each distinct community functions as an individual agent, with the primary objective of maximizing its own rewards. Consequently, we find ourselves dealing with multiple agents concurrently managing the battery to optimize their respective utility functions. This naturally evolves into a multi-agent scenario where each agent possesses its own utility function.

Acknowledgement:
This work was supported by ESIF ERDF Cornwall New Energy project, project number: 05R16P00282. SD was partially supported by the ERDF Deep Digital Cornwall project number: 05R18P02820.

Author Contribution:
DP – Methodology, Software, Formal analysis, Investigation, Data Curation, Writing - Original Draft, Visualization; OT – Conceptualization, Validation, Software, Formal analysis, Investigation, SD – Conceptualization, Resources, Validation, Writing - Review & Editing, Supervision, Project administration, Funding acquisition, MA - Validation, Conceptualization, Validation, Writing - Review & Editing, Project administration, Supervision, Funding acquisition.

REFERENCES

Abedi, S., Yoon, S.W., Kwon, S., 2022. Battery energy storage control using a reinforcement learning approach with cyclic time-dependent Markov process. International Journal of Electrical Power and Energy Systems. 134, 107368.

Ali, K.H., Sigalo, M., Das, S., Anderlini, E., Tahir, A.A., Abusara, M., 2021. Reinforcement Learning for Energy-Storage Systems in Grid-Connected Microgrids: An Investigation of Online vs. Offline Implementation. Energies. 14, 5688.

Bellemare, M.G., Dabney, W., Munos, R., 2017. A Distributional Perspective on Reinforcement Learning, in: Precup, D., Teh, Y.W. (Eds.), Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR, pp. 449–458.

Braun, M., Büdenbender, K., Magnor, D., Jossen, A., 2009. Photovoltaic self-consumption in Germany: using lithium-ion storage to increase self-consumed photovoltaic energy, in: 24th European Photovoltaic Solar Energy Conference (PVSEC), Hamburg, Germany.

Bui, Y.-H., Hussain, A., Kim, H.-M., 2020. Double Deep Q-Learning-Based Distributed Operation of Battery Energy Storage System Considering Uncertainties. IEEE Transactions on Smart Grid. 11, 457–469.

Chang, Y., Mao, X., Zhao, Y., Feng, S., Chen, H., Finlow, D., 2009. Lead-acid battery use in the development of renewable energy systems in China. Journal of Power Sources. 191, 176–183.

Cheng, Y., Qian, C., Crow, M.L., Pekarek, S., Atcitty, S., 2006. A Comparison of Diode-Clamped and Cascaded Multilevel Converters for a STATCOM With Energy Storage. IEEE Transactions on Industrial Electronics. 53, 1512–1521.

Dabney, W., Ostrovski, G., Silver, D., Munos, R., 2018a. Implicit Quantile Networks for Distributional Reinforcement Learning, in: Dy, J., Krause, A. (Eds.), Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR, pp. 1096–1105.

Dabney, W., Rowland, M., Bellemare, M., Munos, R., 2018b. Distributional reinforcement learning with quantile regression, in: Proceedings of the AAAI Conference on Artificial Intelligence.

Dhaene, J., Kukush, A., Linders, D., Tang, Q., 2012. Remarks on quantiles and distortion risk measures. European Actuarial Journal. 2, 319–328.

Divya, K.C., Østergaard, J., 2009. Battery energy storage technology for power systems—An overview. Electric Power System Research 79, 511–520.

Du, Y., Li, F., 2020. Intelligent Multi-Microgrid Energy Management Based on Deep Neural Network and Model-Free Reinforcement Learning. IEEE Transactions on Smart Grid. 11, 1066–1076.

Fan, D.D., Agha-mohammadi, A., Theodorou, E.A., 2022. Learning Risk-Aware Costmaps for Traversability in Challenging Environments. IEEE Robotics and Automation Letters. 7, 279–286.

Foruzan, E., Soh, L.-K., Asgarpoor, S., 2018. Reinforcement Learning Approach for Optimal Distributed Energy Management in a Microgrid. IEEE Transactions on Power Systems. 33, 5749–5758.

GB National Grid Status [WWW Document], n.d. URL https://www.gridwatch.templar.co.uk/.

Glavic, M., Fonteneau, R., Ernst, D., 2017. Reinforcement Learning for Electric Power System Decision and Control: Past Considerations and Perspectives. IFAC-Pap. 50, 6918–6927.

Green Energy UK Tariff. [WWW Document], n.d. URL https://www.theguardian.com/money/2017/jan/03/green-energy-uklaunches-first-time-of-day-electricity-tariff

Guggenberger, J.D., Elmore, A.C., Tichenor, J.L., Crow, M.L., 2012. Performance Prediction of a Vanadium Redox Battery for Use in Portable, Scalable Microgrids. IEEE Transactions on Smart Grid. 3, 2109–2116.

Guo, C., Wang, X., Zheng, Y., Zhang, F., 2022. Real-time optimal energy management of microgrid with uncertainties based on deep reinforcement learning. Energy. 238, 121873.

Open AI Gym Environment.

Harrold, D.J.B., Cao, J., Fan, Z., 2022. Data-driven battery operation for energy arbitrage using rainbow deep reinforcement learning. Energy. 238, 121958.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D., 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. Proceedings on AAAI Conference Artificial Intelligence. 32.

Hua, H., Qin, Y., Hao, C., Cao, J., 2019. Optimal energy management strategies for energy Internet via deep reinforcement learning approach. Applied Energy. 239, 598–609.

Kintner-Meyer, M.C., Balducci, P.J., Jin, C., Nguyen, T.B., Elizondo, M.A., Viswanathan, V.V., Guo, X., Tuffner, F.K., 2010. Energy Storage for Power Systems Applications: A Regional Assessment for the Northwest Power Pool (NWPP) (No. PNNL-19300, 991590).

Kolodziejczyk, W., Zoltowska, I., Cichosz, P., 2021. Real-time energy purchase optimization for a storage-integrated photovoltaic system by deep reinforcement learning. Control Engineering Practice. 106, 104598.

Kottick, D., Blau, M., Edelstein, D., 1993. Battery energy storage for frequency regulation in an island power system. IEEE Transactions on Energy Conversion. 8, 455–459.

Lei, L., Tan, Y., Dahlenburg, G., Xiang, W., Zheng, K., 2021. Dynamic Energy Dispatch Based on Deep Reinforcement Learning in IoT-Driven Smart Isolated Microgrids. IEEE Internet of Things Journal. 8, 7938–7953.

Leung, K.K., Sutanto, D., 2003. Storage power flow controller using battery storage. IEE Proceedings - Generation Transmission and Distribution. 150, 727.

Liang, Y., Guo, C., Ding, Z., Hua, H., 2020. Agent-Based Modeling in Electricity Market Using Deep Deterministic Policy Gradient Algorithm. IEEE Transactions on Power Systems. 35, 4180–4192.

Liu, X., Zhu, T., Jiang, C., Ye, D., Zhao, F., 2022. Prioritized Experience Replay based on Multi-armed Bandit. Expert Systems with Applications. 189, 116023.

Mahmud, K., Hossain, M.J., Town, G.E., 2018. Peak-Load Reduction by Coordinated Response of Photovoltaics, Battery Storage, and Electric Vehicles. IEEE Access. 6, 29353–29365.

Mbuwir, B., Ruelens, F., Spiessens, F., Deconinck, G., 2017. Battery Energy Management in a Microgrid Using Batch Reinforcement Learning. Energies. 10, 1846.

McClelland, J.L., McNaughton, B.L., O'Reilly, R.C., 1995. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. Psychology Reviews. 102, 419–457.

McKenna, E., McManus, M., Cooper, S., Thomson, M., 2013. Economic and environmental impact of lead-acid batteries in grid-connected domestic PV systems. Applied Energy. 104, 239–249.

Min, K., Kim, H., Huh, K., 2019. Deep Distributional Reinforcement Learning Based High-Level Driving Policy Determination. IEEE Transactions on Intelligent Vehicles. 4, 416–424.

Ming-Tsung Tsai, Chin-E Lin, Wen-Inne Tsai, Ching-Lien Huang, 1995. Design and implementation of a demand-side multifunction battery energy storage system. IEEE Transactions on Industrial Electronics. 42, 642–652.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K., 2016. Asynchronous Methods for Deep Reinforcement Learning, in: Balcan, M.F., Weinberger, K.Q. (Eds.), Proceedings of The 33rd International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR, New York, New York, USA, pp. 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature. 518, 529–533.

Nakabi, T.A., Toivanen, P., 2021. Deep reinforcement learning for energy management in a microgrid with flexible demand. Sustainable Energy Grids and Networks. 25, 100413.

Newsham, G.R., Bowker, B.G., 2010. The effect of utility time-varying pricing and load control strategies on residential summer peak electricity use: A review. Energy Policy. 38, 3289–3296.

Nguyen, T.A., Qiu, X., Guggenberger II, J.D., Crow, M.L., Elmore, A.C., 2014. Performance characterization for photovoltaic-vanadium redox battery microgrid systems. IEEE Transactions on Sustainable Energy. 5, 1379–1388.

Panda, D.K., Das, S., 2021. Economic operational analytics for energy storage placement at different grid locations and contingency scenarios with stochastic wind profiles. Renewable and Sustainable Energy Reviews. 137, 110474.

Pudjianto, D., Djapic, P., Aunedi, M., Gan, C.K., Strbac, G., Huang, S., Infield, D., 2013. Smart control for minimizing distribution network reinforcement cost due to electrification. Energy Policy. 52, 76–84.

Rösler, U., 1992. A fixed point theorem for distributions. Stochastic Processes and Their Applications. 42, 195–214.

Ruelens, F., Claessens, B.J., Quaiyum, S., De Schutter, B., Babuska, R., Belmans, R., 2018. Reinforcement Learning Applied to an Electric Water Heater: From Theory to Practice. IEEE Transactions on Smart Grid. 9, 3792–3800.

Sebastián, R., Alzola, R.P., 2010. Effective active power control of a high penetration wind diesel system with a Ni–Cd battery energy storage. Renewable Energy. 35, 952–965.
Shuai, H., He, H., 2021. Online Scheduling of a Residential Microgrid via Monte-Carlo Tree Search and a Learned Model. IEEE Transactions on Smart Grid. 12, 1073–1087.
Shuai, H., Li, F., Pulgar-Painemal, H., Xue, Y., 2021. Branching Dueling Q-Network-Based Online Scheduling of a Microgrid With Distributed Energy Storage Systems. IEEE Transactions on Smart Grid. 12, 5479–5482.
Song, H., Liu, Y., Zhao, J., Liu, J., Wu, G., 2021. Prioritized Replay Dueling DDQN Based Grid-Edge Control of Community Energy Storage System. IEEE Transactions on Smart Grid. 12, 4950–4961.
Subramanya, R., Sierla, S.A., Vyatkin, V., 2022. Exploiting Battery Storages With Reinforcement Learning: A Review for Energy Professionals. IEEE Access. 10, 54484–54506.
Sutton, R.S., 1988. Learning to predict by the methods of temporal differences. Machine Learning. 3, 9–44.
Tervo, E., Agbim, K., DeAngelis, F., Hernandez, J., Kim, H.K., Odukomaiya, A., 2018. An economic analysis of residential photovoltaic systems with lithium ion battery storage in the United States. Renewable and Sustainable Energy Reviews 94, 1057–1066.
Totaro, S., Boukas, I., Jonsson, A., Cornélusse, B., 2021. Lifelong control of off-grid microgrid with model-based reinforcement learning. Energy. 232, 121035.
Tsantekidis, A., Passalis, N., Toufa, A.-S., Saitas-Zarkias, K., Chairistanidis, S., Tefas, A., 2021. Price Trailing for Financial Trading Using Deep Reinforcement Learning. IEEE Transactions on Neural Networks and Learning Systems. 32, 2837–2846.
Tsitsiklis, J.N., Van Roy, B., 1997. An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control. 42, 674–690.
Van Hasselt, H., Guez, A., Silver, D., 2016. Deep Reinforcement Learning with Double Q-Learning. Proceedings of AAAI Conf. Artificial Intelligence. 30.
Von Neumann, J., Morgenstern, O., 2007. Theory of games and economic behavior, in: Theory of Games and Economic Behavior. Princeton university press.
Wan, Z., Li, H., He, H., Prokhorov, D., 2019. Model-Free Real-Time EV Charging Scheduling Based on Deep Reinforcement Learning. IEEE Transactions on Smart Grid. 10, 5246–5257.
Wang, B., Li, Y., Ming, W., Wang, S., 2020. Deep Reinforcement Learning Method for Demand Response Management of Interruptible Load. IEEE Transactions on Smart Grid. 11, 3146–3155.
Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N., 2016. Dueling Network Architectures for Deep Reinforcement Learning, in: Balcan, M.F., Weinberger, K.Q. (Eds.), Proceedings of The 33rd International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR, New York, New York, USA, pp. 1995–2003.
Watkins, C.J.C.H., Dayan, P., 1992. Q-learning. Machine Learning. 8, 279–292.
Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., Su, Y., Su, H., Zhu, J., 2022. Tianshou: A highly modularized deep reinforcement learning library. Journal of Machine Learning Research. 23, 1–6.
Xie, J., Sun, W., 2022. Distributional Deep Reinforcement Learning-Based Emergency Frequency Control. IEEE Transactions on Power Systems. 37, 2720–2730.
Yaari, M., 1987. The dual theory of choice under risk Econometrica 55 (1): 95-115. Authors' Addresses Francesca Beccacece Bocconi Univ. Inst. Quant. Methods Viale Isonzo 25, 20135.
Yang, D., Zhao, L., Lin, Z., Qin, T., Bian, J., Liu, T.-Y., 2019. Fully parameterized quantile function for distributional reinforcement learning. Advanced Neural Information Processing Systems. 32.
Yang, Z., Shen, C., Zhang, L., Crow, M.L., Atcitty, S., 2001. Integration of a StatCom and battery energy storage. IEEE Transactions on Power System. 16, 254–260.
Zeng, J., Zhang, B., Mao, C., Wang, Y., 2006. Use of Battery Energy Storage System to Improve the Power Quality and Stability of Wind Farms, in: 2006 International Conference on Power System Technology. Presented at the 2006 International Conference on Power System Technology, IEEE, Chongqing, China, pp. 1–6.
Zhang, Q., Saad, W., Bennis, M., 2022. Millimeter Wave Communications With an Intelligent Reflector: Performance Optimization and Distributional Reinforcement Learning. IEEE Transactions on Wireless Communications. 21, 1836–1850.
Zhang, Y., Li, Q., Li, Z., 2021. Intelligent demand response resource trading using deep reinforcement learning. CSEE Journal Power Energy Systems.
Zhou, Kunshu, Zhou, Kaile, Yang, S., 2022. Reinforcement learning-based scheduling strategy for energy storage in microgrid. Journal of Energy Storage. 51, 104379.
Zhuang, Y., Li, Y., Cheng, L., Wang, C., Lin, E., 2022. Online Scheduling of PV and Energy Storage System Based on Deep Reinforcement Learning, in: 2022 IEEE International Conference on Power Systems Technology (POWERCON). Presented at the 2022 IEEE International Conference on Power Systems Technology (POWERCON), IEEE, Kuala Lumpur, Malaysia, pp. 1–6.