Explainable AI
Deep Reinforcement Learning Agents for Residential Demand Side Cost Savings in Smart Grids

Hareesh Kumar
IIT Bombay
hareesh@iitb.ac.in

Abstract

Motivated by the recent advancements in deep Reinforcement Learning (RL), we develop an RL agent to manage the operation of storage devices in a household designed to maximize demand-side cost savings. The proposed technique is data-driven, and the RL agent learns from scratch on how to efficiently use the energy storage device under variable tariff-structures âĂŤ contracting the concept of the "black box" where the techniques learned by the agent are ignored. We explain the learning progression of the RL agent, and the strategies it follows based on the capacity of the storage device.

Keywords Deep Reinforcement Learning, Demand Side Cost Reduction, Demand Response, Energy Management System, Explainable AI

1 Introduction

Renewable energy is the fastest growing source of energy, accounting for more than half of the energy supply and is set to penetrate the global energy system more quickly than any fuel previously in history. [10]

But, there is a mismatch in the hours of renewable energy generation and the hours when there is actual demand for energy. Renewable energy sources like Solar PV have a high potential for energy during the noon hours, but the higher demand in the residential buildings is during the evening hours as most of the people spend the day hours in the office. This imbalance of the demand and supply in the modern (smart) grids has forced utilities to go for variable pricing methods such as Time of Day (ToD) or Dynamic Pricing to motivate the consumers to consume energy during times of higher availability. The price of the electricity is usually less during the high supply hours, and is high during the high demand hours. The utilities indirectly reward users who consume during the high supply hours and penalize the consumers who consume during the high demand hours. Some utilities also directly pay consumers to reduce energy consumption during the high demand hours. Such Demand Response(DR) occasions call for intelligent decision making by the entities involved, helping reduce energy costs for both consumers and producers.

In some developing countries, when there is high demand but less supply, utilities adopt rolling blackouts where specific regions suffer from blackouts for a few hours [4]. Frequent blackouts have led to a large number of households purchasing battery-based energy storage devices as backup power supplies when energy from the grid is not available. Over the years, the efficiency of the batteries has increased [22] while cost reduction has been felt by the consumers. This has further fueled the adoption of batteries.

Demand Side Energy Management[3, 17, 18] offers a set of techniques to improve building energy consumption via load shifting or load reduction. However, the deferrable loads are limited to dishwashers, washing machine, air-conditioners, plug-in electric vehicles, etc. With a significant fraction of non-deferrable loads, it is easier to meet the energy needs using energy storage systems, under variable energy pricing. Modeling an efficient Energy Management System which can optimize the usage of battery can lead to cost savings for the consumers. This could also help in increasing the stability of the grid by decreasing the demands during energy deficit hours. Traditional methods rely on rule-based techniques to optimize battery operation[1, 19]. But, these methods need to be flexible and must adapt to changes in the tariff methods.

Reinforcement Learning(RL) has gathered a lot of popularity especially in the gaming domain, where it has been shown that it can surpass human level decision making in most of the games[9, 16]. In this paper, we model a deep reinforcement learning agent which can learn to work under a variable pricing regime to provide cost savings to the consumers of energy. The goal of the RL agent is to maximize cost savings, i.e., to reduce the cost that has to be paid to the utility. The cost savings of the agent is calculated using the formula below

$$cost\_saving = \left(1 - \frac{Cost(RLAgent)}{Cost(BaseLine)}\right) * 100 \qquad (1)$$

where $Cost(BaseLine)$ is the baseline cost, i.e., the cost that the consumer would have paid in the absence of the RL agent, $Cost(RLAgent)$ is the cost incurred by the trained RL agent.

2 Related Work

Extensive studies have been carried out on the optimal control of energy storage systems. Many of them [2, 7, 12]try to

formulate it as an optimization problem where the decisions are taken beforehand. For instance, in [7] a linear integer program is solved to determine the battery operation to improve the profitability over a period of 20 years. Moreover these papers do not consider the uncertainty in consumption and also rely on accurate prediction and information. Few papers [6] considering the uncertainty in information cannot handle problems with large state space. Advances in RL offer an opportunity to solve problems with large state space. Also, when compared to traditional rule-base approaches, RL algorithms learn the best control policy by itself. Works such as [5, 11, 14, 15] try to solve problems in energy domain with new RL techniques. However, many of them are formulated for the purpose of better battery utilization along with the renewables and presented the agent as a black box [5, 11]. Although solar PV deployments are not common in households, most of them adopt battery storage systems for managing energy shortage situations. None of the current RL approaches propose an intelligent decision-making agent for battery storage operations (excluding renewables) in households under various electricity tariff structures.

3 Modeling the Energy Management System

Reinforcement Learning is the study of decision making over time in complicated environments. Fig 1 shows the RL agent, at time t, the agent is in state s_t, performs an action a_t from the list of possible actions which is executed in the Environment, and the reward r_t for taking that action along with the next state s_{t+1} is returned to the agent. In our case, the agent is the controller box that performs actions, and the environment is the utility. State s_t is the environment's private representation, action a_t is the combination of charge/discharge operations that the agent performs. The rewards are computed based on how good the chosen action is. The rewards are computed by the Environment and are returned to the Agent.

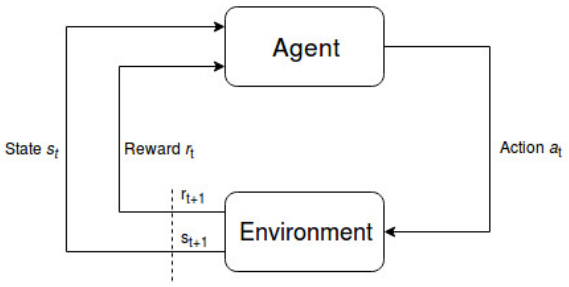

Figure 1. Reinforcement Learning

3.1 Observation State

Observation state is represented as a continuous array which contains, Time of the Day Pricing or the forecasted price for the next n hours along with the status of the battery and the current load. Other data points (if any) could also be added to this list.

$$s_t = [t_1, t_2, t_3....t_n, battery\_status, current\_load, o_1, o_2..o_n] \tag{2}$$

where $t_1, t_2, t_3....t_n$ are the Time of the Day Pricing or the forecasted price for n hours, and $battery\_status$ is the status of the battery, $current\_load$ is the current demand and $o_1, o_2..o_n$ are any other information which could be appended. The observation state is not labeled, the information about what the values in the list signify is not known to the agent.

3.2 Action Space

Action Space is the set of possible actions that the agent can perform in a State. There are three possible actions that the Reinforcement Agent can choose.

- Grid: The Smart Grid fulfills the consumer demand for energy
- Battery Discharge+Grid: The Energy Storage device fulfills the maximum possible consumer demand, and the Smart Grid fulfills the rest of the consumer demand.
- Battery Charge + Grid: The Smart Grid fulfills the total consumer demand, as well as the energy storage device demand.

3.3 Reward Function

The goal of the Reinforcement Learning agent is to maximize the expected cumulative reward.

$$R = \sum_{t=0}^{\infty} \gamma^t r_t \tag{3}$$

where γ with value between 0 and 1 is the discount factor; the larger the value of γ the more importance is given to the future reward and r_t is the reward at time t.

Episodic Reward: In this case, we have a starting and an ending point, and the reward is computed and returned to the agent at the ending point. In the case of episodic reward the reward could be given as the negative of the total cost that the user will have to pay to the utility.

Continuous Reward: The is an immediate reward given for every action that the agent performs. E.g., let us assume we are training a robot to walk, we can give rewards in the form of linear/exponential function when it balances itself and walks. If the robot crashes then we can give a high negative reward and terminate.

Here we model the reward as a continuous reward as it intuitively feels that the agent can learn better if continuous reward is given. The reward at time t is computed as the cost to be paid to the utility to consume from the grid

$$R(t) = -Cost\_electricity(t) \times demand(t) + penalty(demand(t)) \tag{4}$$

where $R(t)$ is the reward at time t. In some of the cases, there is a penalty that has to be paid for exceeding the maximum

demand limit, which is calculated by penalty(x). The incentives in the case of Demand Response can also be integrated with the penalty function. Negation is used here as we want the RL agent to minimize the cost. The function which is used to calculate the rewards, incentive or the real-time pricing is not known to the agent, but the result is known i,e the agent will know the cost of the price at a different time but doesn't know how the values are computed.

4 Framework for the RL agent

We use Q-Learning to learn a policy which will help the RL agent to perform optimal action given a state. Given a state s and a action a, $Q(s, a)$ denotes how good/bad is it to take action a being in state s. Q-Value of a state is computed using the Bellman Equation, Eq. 5 by updating the Q-Values until convergence.

$$Q^{new}(s_t, a_t) = (1-\alpha)*Q(s_t, a_t)+\alpha*(r_t+\gamma*max_a Q(s_{t+1}, a)) \tag{5}$$

where α is the learning rate, $Q^{new}(s_t, a_t)$ is the new value for state $Q(s_t, a_t)$, γ is the discount factor and $max_a Q(s_{t+1}, a)$ is the maximum possible reward given s' and all possible actions at that state.

Deep Q-Learning: Here we use Deep Neural Network to approximate the Q-Values, which would have been obtained by Eq. 5. Neural Network takes the state s as input and outputs the Q-Value for every action that could be taken. The agent chooses the action which results in the maximum Q-Value at each step. Rectified Linear unit(ReLU) is used as an activation function for the hidden layers in the neural network.The target $Q^*(s_t, a_t)$ values in neural network is estimated using equation

$$Q^*(s_t, a_t) = r_{t+1} + \gamma max_{a_t+1} - Q(s_{t+1}, a_{t+1}) \tag{6}$$

The loss function for the network is computed as the Mean Squared Error(MSE) of the computed Q-Value of the neural network to the target Q value. Mini-batch gradient descent method is used to update the parameters in the neural network $\omega = \alpha\triangle\omega$ where α is the learning rate, and the $\triangle\omega$ is the derivative of the loss function $\triangle\omega = \frac{\partial L}{\partial w}$.
The loss function is calculated as

$$L = \frac{1}{2n}\sum_{1}^{n}[(Q^*(s_t, a_t) - Q(s_t, a_t))^2] \tag{7}$$

4.1 Prioritized Experience Replay(PER)

We use Prioritized Experience Replay(PER) [13] where the experiences(state, action, reward, next_state) are stored based on the priorities, and the experiences are chosen based on the priority. The priority of the experience is set based on the predicted value and the target value, higher the difference higher the priority. Probability of being chosen for a reply is given by stochastic prioritization

$$P(i) = \frac{p_i^a}{\sum_k p_k^a} \tag{8}$$

where $p(i)$ is the priority value of i being selected, hyperparameter a is used to introduce randomness in the experience selection for replay buffer. If a=0, then it is pure randomness, if a=1 then select only the experiences with highest priorities. Updates to the network are weighted with Importance Sampling weights(IS), to account for the change in the distribution.The IS weights are updated by reducing the weights of the often seen samples.

$$IS = \left(\frac{1}{N} * \frac{1}{P(i)}\right)^b \tag{9}$$

where N is the size of the Replay buffer size, P(i) is the sampling probability and b is to control how the sampling weights affect the learning process. If $b = 0$ then no importance sampling, and $b = 1$ then it is full importance sampling.

4.2 Fixed Q-targets

We use the idea of fixed Q-targets introduced by Mnih et al. [9], by using a separate network with a fixed parameter for estimating target value and at every T step, we copy the parameters from DQN network to update the target network. This improves the stability of the Neural Network as the updates are not done to the target function after every batch of learning.

$$\triangle w = \alpha[(R + \gamma max_a \hat{Q}(s', a, w^-)) - \hat{Q}(s, a, w)]\nabla_w \hat{Q}(s, a, w) \tag{10}$$

where $\triangle w$ is the change in weights, α is the learning rate, $\nabla_w \hat{Q}(s, a, w)$ is the gradient of the predicted Q-value, $\hat{Q}(s', a, w^-)$ is the current predicted Q-value and $(R + \gamma max_a \hat{Q}(s', a, w^-))$ is the maximum possible Q-value for the next state predicted from the target network.

4.3 Double DQN

Double DQN introduced by Hado van Hasset[20] is used to handle the problem of overestimation of the Q-values. The accuracy of the predicted Q-Value depends on the actions that are explored. Taking the maximum Q-value will be noisy during the initial phase of the training if non-optimal actions are regularly given high Q-value than the optimal best action, then the learning will be complicated. The best action to take for the next state (the action with the highest q-value) is computed from the DQN network, and the target Q Value of taking that action at the next state is computed from the target network.

$$Q(s, a) = r(s, a) + \gamma Q(s', argmax_a Q(s', a)) \tag{11}$$

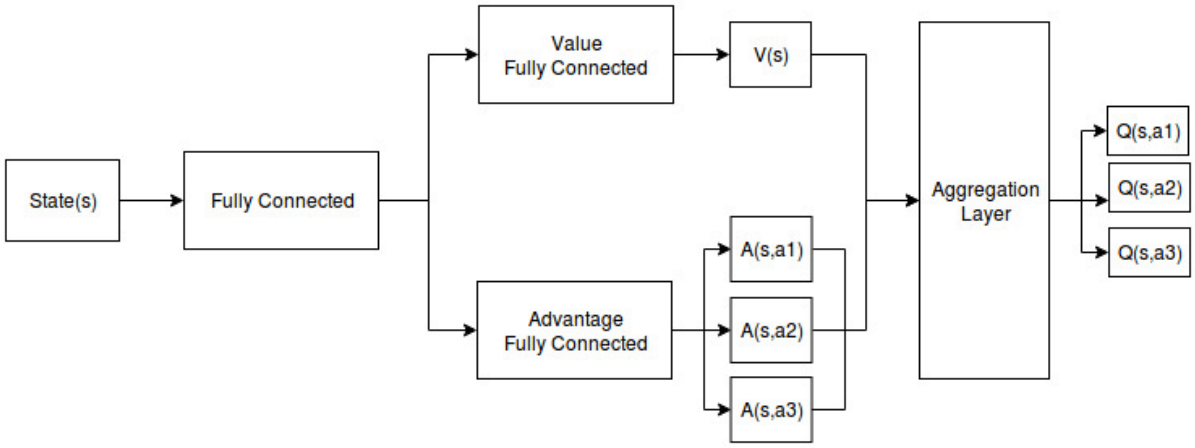

Figure 2. Double Dueling Deep Q-Learning Network (DDDQN)

$Q(s,a)$ is the Q-target, $r(s,a)$ is the reward of taking that action at that state and $\gamma Q(s', argmax_a Q(s',a)$ is the discounted max q value amoung all possibles actions from next state

4.4 Dueling Double DQN

In a Dueling Double Deep Q- Learning Neural Network (DDDQN) [21], the value of Q(s, a) is computed as the sum of the value of being in that state $V(s)$ and the advantage of taking action at that state $A(s,a)$. By decoupling the estimation, intuitively our DDQN can learn which states are (or are not) valuable without having to learn the effect of each action at each state since its also calculating V(s). This helps in not choosing the local minima as the advantage of the taking the action is also considered.

$$Q(s,a,\theta;\alpha,\beta) = V(s;\theta,\beta)+(A(s,a;\theta,\alpha)-\frac{1}{A}\sum_{a'} A(s,a';\theta,\alpha)) \tag{12}$$

θ is the common network parameters, α advantage stream parameters, β value stream parameters, $\frac{1}{A}\sum_{a'} A(s,a';\theta,\alpha)$ is the average advantage. This architecture helps in boosting the training as we can calculate the value of the state without calculating the value for $Q(s,a)$ for each action at that state, it also helps us to find the reliable Q-Values for each action as the value and the advantage are decoupled.

Fig 2 shows the DDDQN architecture,. The *Value Fully Connected* is used to calculate the value function of the state and the *Advantage Fully Connected* is used to calculate the advantage of the taking the action. The aggregation layer performs the Eq. 12

5 Experiments and Results

We use the dataset of high-rise residential building[8], Fig 3 shows the consumption of the apartment which was considered under the study. The blue line represents the avg. consumption of the apartment for a month and the the yellow line shows the consumption of the apartment for a day.

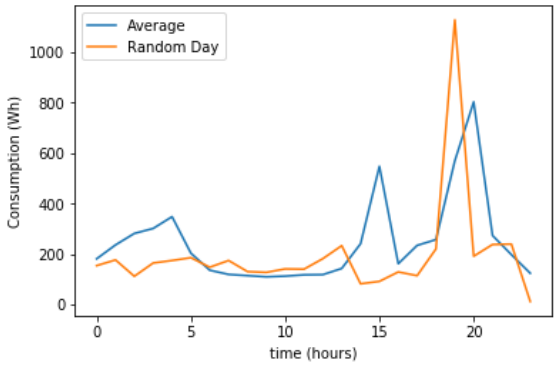

Figure 3. Sample Consumption

5.1 Naive RL agent

During the training phase, the model was saved for every ten iterations. Then saved model was evaluated on the test dataset, we then explain the learnings of the agent.

Observation Space The observation space consists of the Time of Day(ToD) pricing for the next 24 hours along with the battery status and the current load. Time of Day pricing was modeled as shown in Table 1

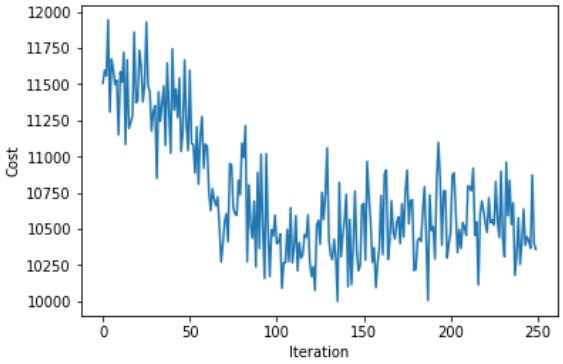

Figure 4. Battery Status

ToD - Time Slot	Cost (x/kWh)
00:00 Hours to 08:00 Hours	1
08:00 Hours to 16:00 Hours	3
16:00 Hours to 24:00 Hours	2

Table 1. ToD Pricing

The Energy storage used here is battery with the capacity of 900Wh and the maximum discharge/charge rate set to 300W.

Rewards The rewards at time t are calculated using the Eq 4 and value for penalty is assumed to be null.

Hyper-parameters

Hyperparameter	Value
mini batch size	32
replay memory size	10240
discount factor	0.96
learning rate	0.00025
initial exploration	1.0
final exploration	0.1

Table 2. Hyper-parameters

RL agent was trained on a 30day residential dataset. The hyper-parameters which were used during the training is mentioned in the Table 2. Fig 4 shows the gradient descent of the RL agent, where it tries to reduce the energy cost for the residence.

5.1.1 Learnings of RL agent

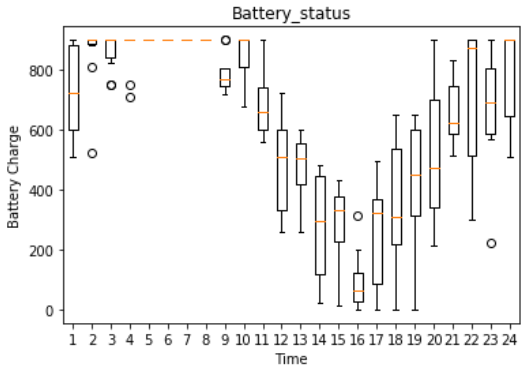

Figure 5. RL agent learning when to consume from the battery

Around 50 iterations: The agent learns insights on the data given to it and how it affects the gradient ascent. The Agent learns when the cost of the energy is high, and consuming energy from the battery during those hours will help in reducing the energy cost. Fig 5 shows agent discharging the battery during the 8-16th hour, which are the hours when the cost of the energy is high.

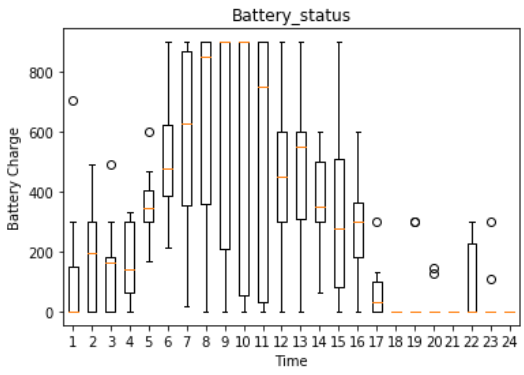

Figure 6. RL agent learning when to charge the battery

Around 100 iterations: The agent learns when the cost of the energy is cheap, and charging during those hours will help in reducing the energy cost. Fig 6 shows agent charging the battery during the 0-8th hour when the cost of the energy is low. It should also be observed that the agent sometimes performs random charging and discharging in the slots, and these do not affect end cost. The agent also learns not to perform any action during the 17-24 hours as this does not increase or decrease the cost savings.

ToD - Time Slot	Rs/kWh
06:00 Hours to 09:00 Hours	0.00
9:00 Hours to 12:00 Hours	0.50
12:00 Hours to 18:00 Hours	0.00
18:00 Hours to 22:00 Hours	1.00
22:00 Hours to 06:00 Hours	-0.75

Table 3. Tata ToD Tariff (Base Price 5Rs)

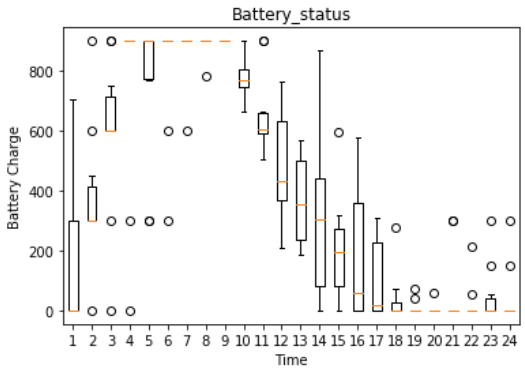

Figure 7. RL agent learning Time of Day

Around 200 iterations: Fig 6 shows the agent charging the battery as soon as the cost of the energy is low, learns to discharge during the high hours, and does not do anything during the other hours.

5.2 Case Study : Mumbai, India

We model the environment based on Tata Power Tariff, High Tension Residential consumer(Housing Society). We have also included the Time of the Day pricing which is not applicable for the residential loads but mandatory for most of the consumers in the High Tension load and is optional for a few of the consumers. High Tension residential consumers are charged at 5Rs per kWh. Table 3 shows the additional ToD pricing which is followed by Tata Power (Base price of 5Rs has to added to the ToD specified). The cost of the electricity is lowest from 22.00-06.00 hours with the cost of 4.25Rs/kWh and the cost of the electricity is highest during 18.00-22.00 hours with the cost of 6Rs/kWh.

5.2.1 Hyperparameters

Observation space used here is similar to the one described in the experiment 4.1. The architecture used in the experiments is described in section 2. The hyper-parameters which were used during the training is mentioned in the Table 4. The capacity of the battery was varied in the range from 5kWh to 30kWh. The models were initialized with random weights initially. Since deep charge or discharge of the battery reduces the lifetime of the battery, it was ensured that the battery could maximum charge up to 90% of its total capacity, and the maximum discharge of the battery was limited to 10% of the total capacity. The charging capacity and the discharging rate was set to 70% of the battery's capacity. The loss occurred during the charging, and discharge was ignored. The cost for the battery and the lifetime of the battery is also not taken into consideration.The model was trained on one month and was tested on the next month of the residential dataset. All the models were trained for 500 epochs.

Hyperparameter	Value
mini batch size	32
replay memory size	10240
agent history length	15 days
target network update frequency	5 days
discount factor	0.96
learning rate	0.00025
initial exploration	1.0
final exploration	0.1

Table 4. Hyper-parameters

5.2.2 Results

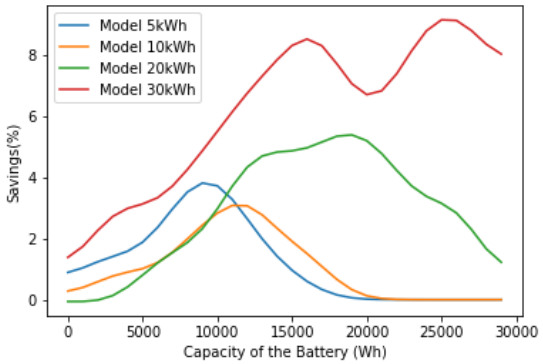

Figure 8. Storage Capacity vs Cost Savings

Fig 8 shows lower capacity batteries fail to perform when tested on high capacity batteries, but the vice-versa is not true. This is because the agents trained on the lower capacity batteries have not seen states that are experienced by the high capacity batteries, but the high capacity batteries have seen the states that are seen by the low capacity batteries.

5.2.3 Low vs High Energy Storage devices

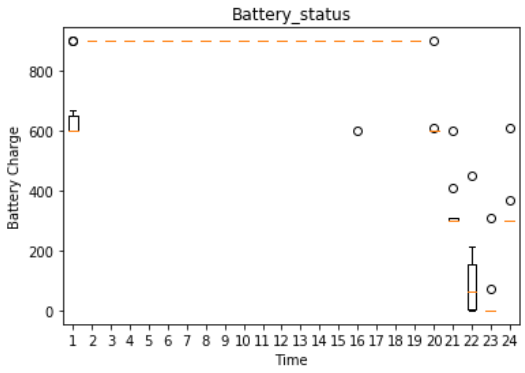

Figure 9. RL agent on small capacity storage devices

Fig 9 shows the common pattern which was observed when the Reinforcement Agent was trained on the lower capacity batteries. It also shows that the agent chooses to charge the battery when the cost of the energy is low and chooses to discharge when the cost of the energy is high. The agent also does not charge/disharge the battery in the rest of the hours.

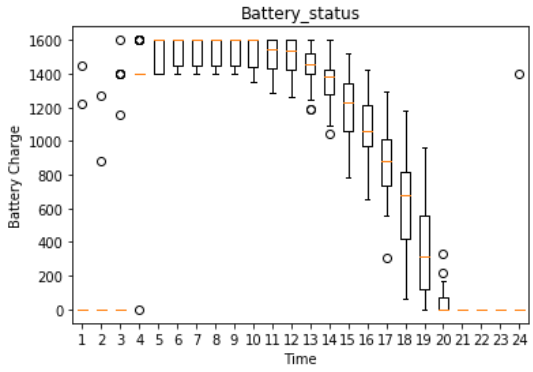

Figure 10. RL agent on large capacity storage devices

Fig 10 shows the trained RL agent on high capacity batteries, it can be observed that the agent chooses to completely charge the battery when the cost of the energy is low(22.00-06.00 hours) and discharge the battery continuously as soon the cost of the energy increases.

5.2.4 Demand Response

In situations like demand response, there is maximum demand limit that is imposed on the consumers, consuming above the maximum demand limit results in heavy penalties imposed by the utility. We model the Demand Response by setting a maximum demand limit per day as 700Wh for the consumer along with the tariff as mentioned in Table 3 and the consumer is penalized adding 2Rs for every unit exceeding the maximum demand limit.

Neural Network used for Demand Response were initialized with the results of the earlier models and was fine-tuned to work for demand response.

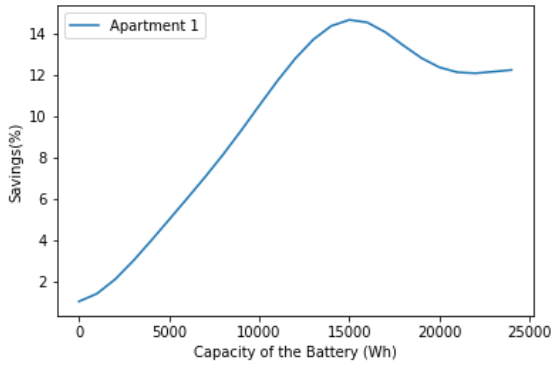

Figure 11. Demand Response

Fig 11 shows the savings of the agent with the ToD pricing and ToD pricing along with the Demand Response Scenario. Fig 11 shows increase in the savings percent when the capacity of the battery is increased, it can be seen that after 15000Wh, the savings trend to flatten around 12%. Here using the battery of 15000Wh is the optimal battery that can be used and it can achieve in 12-14% of cost reduction in this scenario.

Our results show savings of 6-8% under the ToD tariff method as specified in Table 3. Savings of 12-14% which can be obtained if the utility follows the ToD pricing along with the rewards from the DR program. The capital cost for the energy storage system along with the efficiency, life time of the device and other factors has to be considered to calculate the payback period.

6 Conclusion

This paper presents a deep reinforcement learning based data-driven approach to control an energy storage system. Our results show increment in savings when the capacity of the storage device is varied up to a certain capacity, and then the savings remains constant. This can be used to calculate the optimal capacity of the storage that can be installed at the residence. We also show the learnings of the RL agent through the course of training and the strategies followed by the agent when the capacity of the storage device is varied. Future work includes considering other parameters of the storage system like cost, lifetime, etc. which have been ignored in this study. Payback period of the battery can be calculated after these parameters are taken into consideration.

Acknowledgments

I want to thank my guide, **Prof: Krithi Ramamritham** and Priyanka, for their invaluable guidance during the project. I would also like to thank the members of the Smart Energy Informatics Lab, IIT Bombay, for their immense support.

References

[1] Diego Arcos-Aviles, Julio Pascual, Luis Marroyo, Pablo Sanchis, and Francesc Guinjoan. 2016. Fuzzy logic-based energy management system design for residential grid-connected microgrids. *IEEE Transactions on Smart Grid* 9, 2 (2016), 530–543.

[2] Oytun Babacan, Elizabeth L Ratnam, Vahid R Disfani, and Jan Kleissl. 2017. Distributed energy storage system scheduling considering tariff structure, energy arbitrage and solar PV penetration. *Applied energy* 205 (2017), 1384–1393.

[3] Marc Beaudin and Hamidreza Zareipour. 2015. Home energy management systems: A review of modelling and complexity. *Renewable and sustainable energy reviews* 45 (2015), 318–335.

[4] Farhan Beg. 2013. Integrating Wind And Solar Energy In India For A Smart Grid Platform. (2013).

[5] Heider Berlink, Nelson Kagan, and Anna Helena Reali Costa. 2015. Intelligent decision-making for smart home energy management. *Journal of Intelligent & Robotic Systems* 80, 1 (2015), 331–354.

[6] Chenxiao Guan, Yanzhi Wang, Xue Lin, Shahin Nazarian, and Massoud Pedram. 2015. Reinforcement learning-based control of residential energy storage systems for electric bill minimization. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 637–642.

[7] Fiodar Kazhamiaka, Patrick Jochem, Srinivasan Keshav, and Catherine Rosenberg. 2017. On the influence of jurisdiction on the profitability of residential photovoltaic-storage systems: A multi-national case study. *Energy Policy* 109 (2017), 428–440.

[8] Priyanka Mary Mammen, Hareesh Kumar, Krithi Ramamritham, and Haroon Rashid. 2018. Want to Reduce Energy Consumption, Whom should we call?. In *e-Energy*. 12–20.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[10] BP Energy Outlook. 2019. *BP Energy Outlook*. https://www.bp.com/content/dam/bp-country/es_es/spain/documents/downloads/PDF/bp-energy-outlook-2019_book.pdf.

[11] Xin Qiu, Tu A Nguyen, and Mariesa L Crow. 2015. Heterogeneous energy storage optimization for microgrids. *IEEE Transactions on Smart Grid* 7, 3 (2015), 1453–1461.

[12] Elizabeth L Ratnam, Steven R Weller, and Christopher M Kellett. 2015. An optimization-based approach to scheduling residential battery storage with solar PV: Assessing customer benefit. *Renewable Energy* 75 (2015), 123–134.

[13] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

[14] Shinya Sekizaki, Tomohiro Hayashida, and Ichiro Nishizaki. 2015. An intelligent home energy management system with classifier system. In *2015 IEEE 8th International Workshop on Computational Intelligence and Applications (IWCIA)*. IEEE, 9–14.

[15] Guang Shi, Derong Liu, and Qinglai Wei. 2017. Echo state network-based Q-learning method for optimal battery control of offices combined with renewable energy. *IET Control Theory & Applications* 11, 7 (2017), 915–922.

[16] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[17] Goran Strbac. 2008. Demand side management: Benefits and challenges. *Energy policy* 36, 12 (2008), 4419–4426.

[18] Olle Sundström and Carl Binding. 2010. Optimization methods to plan the charging of electric vehicle fleets. In *Proceedings of the international conference on control, communication and power engineering*. Citeseer, 28–29.

[19] Jeroen Tant, Frederik Geth, Daan Six, Peter Tant, and Johan Driesen. 2012. Multiobjective battery storage to improve PV integration in residential distribution grids. *IEEE Transactions on Sustainable Energy* 4, 1 (2012), 182–191.

[20] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[21] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581* (2015).

[22] Chao Zhang, Yi-Li Wei, Peng-Fei Cao, and Meng-Chang Lin. 2018. Energy storage system: Current studies on batteries and power condition system. *Renewable and Sustainable Energy Reviews* 82 (2018), 3091–3106.