

# Home Energy Management System based on Deep Reinforcement Learning Algorithms

**Kahraman, Aysegül; Yang, Guangya**

*Published in:*
Proceedings of 2022 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)

*Link to article, DOI:*
10.1109/ISGT-Europe54678.2022.9960575

*Publication date:*
2022

*Document Version*
Peer reviewed version

# Home Energy Management System based on Deep Reinforcement Learning Algorithms

Aysegül Kahraman
*Department of Wind and Energy Systems*
*Technical University of Denmark*
*Sino-Danish Center for Education and Research*
*University of Chinese Academy of Sciences*
Kgs. Lyngby, Denmark
ayska@dtu.dk

Guangya Yang
*Department of Wind and Energy Systems*
*Technical University of Denmark*
Kgs. Lyngby, Denmark
gyyan@dtu.dk

***Abstract*—With the recent progress in smart grid applications, home energy management system has increased its importance since it allows prosumers to be active participants of the system operation. Operating the smart grid in an efficient way without having a contingency issue has become paramount. The uncertainty of the system inputs, such as renewable energy and load consumption, with the effect of dynamic user behavior, brings the necessity of a more complex control system. In this paper, we introduce three different Deep Reinforcement Learning (DRL) algorithms to minimize the operational cost in the long run and keep the battery state of charge (SoC) between the operable limits. The idea behind applying three different DRLs is to present the powerful and weak sides of the DQN, DDPG, and TD3 algorithms in terms of solving a management problem, even with the continuous state and action space for longer horizons. Experimental results show that the proposed RL algorithms can be employed to solve this and similar management problems. These show that DRL algorithms promise to solve even more complex problems with their uncertainties, but it is difficult to guarantee that they will reach an optimal solution.**

***Index Terms*—deep reinforcement learning, scheduling, home energy management system, prosumer, battery**

## I. Introduction

With the recent increase in integration of uncertain renewable energy sources, energy storage systems and volatile energy demand, it unfolds major changes to handle the challenges such as growing complexity, increasing uncertainty and aggravating volatility [1]. These changes inspire the customers' desire to be more than just a consumer, but also a prosumer. Being a prosumer increases flexibility by including multi-energy systems such as electricity, gas, and combined systems in a cost-effective way. However, this kind of deployment needs different and smarter control and optimization methods outside traditional power system operation models, which impose different requirements on the methods and implementation.

The concept of having a successful complex decision-making system without model information has recently gained attention, since model-based methods often meet challenges for regulating large amount of devices, which in turn makes Reinforcement Learning (RL) type of methods a good way to handle complex problems with less information about the model. It basically interacts with the dynamic environment and observes and improves its performance based on the reward with respect to the action taken in the current state under the current policy. RL can be ensured on both the consumer and network side in terms of reliability and economic efficiency. Deep reinforcement learning (DRL) is a combination of deep learning (DL) and RL to combine the capability of working with data-driven and model-free optimal sequential decision making within a dynamic environment instead of model-based approaches since these kinds of scheduling problems by nature are uncertain and complex.

In recent years, significant progress has been made in using RL and DRL for management problems. The authors in [2] employ fuzzy Q-learning to solve the continuing decision problem by including smart and conventional consumers into a peer-to-peer trading system with the use of the energy storage system to maximize the benefit for each participant. Furthermore, it is also possible to schedule home appliances with RL by continuously learning the behaviour of the users. In [3] deep Q-Learning (DQN) and deep double Q-learning (DDQL) methods are used, and it is proved that DDQL is more appropriate method to use with cost minimization for the users for better integrating of electric vehicles into the main grid. In [4], the authors use DQN to optimize peak load and operational cost for home energy management and EV charging by using an additional Long-Short Term Memory (LSTM) layer to generate the load demand forecast for feeding to RL observations by discretizing the continuous action space. When the number of houses, interacted agents, or components is huge, it may result in dimensionality curse which is a common problem for these kind of problems. With the uptrend of EVs, using RL for EV charging scheduling has become a more common topic. The paper [5] applies constrained DRL to guarantee EVs' scheduling which will be fully charged before departure and not necessarily design a penalty term. Similarly, EV charging scheduling is completed by the Q network by extracting the features from the LSTM network to deal with the uncertainty of electricity prices and user behaviour [6]. Similarly, battery scheduling solutions with RL are becoming

The work is funded under Sino-Danish Center for Education and Research.

more common, and different algorithms such as PPO, DDPG, and A2C are applied to tackle problems with continuous action space [7].

All these literature studies show that using RL for control and smart grid applications is getting more attention. In this study, we use the three different RL algorithms to schedule the next three days by minimizing the total operational cost and keeping the SoC level within the operational limits for the prosumer operation. Rhe main contributions are 1) we model the home energy management problem as a Markov decision process (MDP) and 2) create and compare three different model-free DRL architectures in MATLAB and simulink environment to schedule both the power grid and the storage unit to minimize the total operational cost.

## II. Reinforcement Learning and Mathematical Background

The RL uses a reward signal to learn the behaviour function which is policy of the agent and maximize the value function. The RL is a sum of the expected reward for every state and actions $\mathbb{E}_{\tau\sim P_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t,a_t)\right]$ for finite horizon $T$, where the goal of the RL is to find the $\theta$ to define the policy which maximizes the expected value of the sum of rewards through the time over the state and action pairs, represented by $\mathbb{E}_{s_1\sim p_{(s_1)}}\left[\left[\mathbb{E}_{\ a_1\sim\pi(a_1|s_1)}Q(s_1,a_1)|s_1\right]\right]$ when it is written iteratively. Q function depends on policy $\pi$ and can be written as follows:

$$Q^{\pi}(s_t,a_t)=\sum_{t'=t}^{T}\mathbb{E}_{\pi_\theta}\left[r(s_{t'},a_{t'})|(s_t,a_t)\right] \tag{1}$$

Q function gives the total expected reward based on starting from state $s_t$ and taking action $a_t$, then by following the policy $\pi$. It should be underlined that every policy brings different Q-value. Q function can also be referred as the summation of current reward $r(s_t,a_t)$ and expected value of reward in the future ($\sum_{t'=t+1}^{T}\mathbb{E}_{\pi_\theta}\left[r(s'_t,a'_t)|s_t,a_t\right]$) by starting from the $t+1$ and goes to the $T$ since current reward can be calculated based on the current state $s_t$ and action $a_t$. The future value expectation is also equal to the value function on the next state $V^{\pi}(s_{t+1})$. Therefore, Q function becomes the sum of current reward and value function on the next state.

The expected value over all the actions in state $s_t$ with the current policy $\pi$ of the Q value gives the value function. The difference is value function depends on only state not action. And can be written as:

$$V^{\pi}(s_t)=\sum_{t'=t}^{T}\mathbb{E}_{\pi_\theta}\left[r(s_{t'},a_{t'})|(s_t)\right] \tag{2}$$

There are four main RL algorithms; policy-based method seeks to find a derivative of the following objective $\theta^*=\underset{\theta}{argmax}\ \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t,a_t)\right]$ with respect to $\theta$ and executes a grading descent procedure using this derivative. Value-based method estimates the value function or Q function which can be represented by neural networks to improve and eventually reach the optimal policy. Actor-critic methods are a combination of value and policy based methods. It learns the value function and use it to improve the policy and checks the updated value function based on updated policy and continues in this way. The last method is model-based RL which estimates a transition model to use it for directly improving the policy.

### A. Solution Algorithms

Deep Q-learning is a value-based, model-free, off policy RL agent and it can be used with only discrete action space and discrete or continuous observation space. As declared before RL works with the iterations. DQN uses the value iteration algorithm to obtain the policy and Q function is given with the following equation.

$$Q^{\pi}(s,a)=r(s,a)+\gamma\mathbb{E}\left[V(s')\right] \tag{3}$$

$\gamma\in[0,1]$ is discount factor which is necessary to converge to a reward sooner, especially with the non-episodic problems which are not terminate at a certain point. The $V(s)$ can be obtained by taking right action to maximize Q-value $Q(s,a)$. Based on the explanation in the the previous part, we can write the DQN training process as follows [8]:

1) Take an action $a_i$ and observe the dataset $(s_i,a_i,s'_i,r_i)$ and add to the experience buffer
2) take a random mini batch $(s_j,a_j,s'_j,r_j)$ from the buffer and $j$ can be different than $i$
3) compute target value $y_j$ for each element with the following formula

$$y_j=r_j+\gamma max_{a'_j}Q_\phi(s'_j,a'_j)$$

4) update the parameters by taking the gradient for the $y$

$$\phi=\phi-\alpha\sum_j\frac{dQ_\phi}{d\phi}(s_j,a_j)(Q_\phi(s_j,a_j)-y_j)$$

5) update $\phi'$ after every N steps, and repeat the process.

As a summary, value-based methods might have some problems about learning the policy explicitly, these methods are capable to learn the value function or Q function with the help of Q-table or neural networks. With the help of a value function, it is possible to obtain a policy by using the $argmax$ or any other policy. Another potential trouble is the necessity of using continues action space. Although theoretically it is possible to evaluate the Q value for every possible action to find the best action, it brings several issue such as taking the $argmax$ during the training process causes buffer size and taking the maximum can cause overestimation during the computing $y_j$. Here, the solution becomes training a second neural network to perform this operation which is called as DDPG that is an actor-critic method is proposed to deal with these issues.

Deep Deterministic Policy Gradient (DDPG) is model free, online, off-policy and actor critic algorithm. While the action

space is continuous, the observation space can be continuous or discrete. Actor-critic is the hybrid version of deterministic policy actor $\pi(s)$ and Q-value function critic $Q(s,a)$. DDPG training procedure can be written as follows [9]:

1) Take an action $a_i$ and observe the dataset $(s_i, a_i, s_i', r_i)$ and add to the experience buffer (very similar to DQN)
2) take a random mini batch $(s_j, a_j, s_j', r_j)$ from the buffer and $j$ can be different then $i$
3) compute target value $y_j$ for each element

$$y_j = r_j + \gamma Q_{\phi'}(s_j', \mu_{\theta'}, s_j')$$

instead of using $argmax$, $\mu_\theta{}'$ has been introduced as one of the target network.

4) update the parameters by taking the gradient for the $y$

$$\phi = \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(s_j, a_j)(Q_\phi(s_j, a_j) - y_j)$$

5) update the parameters for the other network

$$\theta = \theta + \beta \sum_j \frac{d\mu}{d\theta}(s_j) \frac{dQ_\phi}{da}(s_j, \mu(S_j))$$

6) update $\phi'$ and $\theta'$ after every N steps, and repeat the process.

DRL allows us to use two different or one shared neural network to represent the value ($V^\pi_{\phi(s)}$) and policy function $\pi_\theta(a|s)$. While policy function returns either the parameters of a continues probability distribution for continues action space or a softmax for discrete action space, value function gives a scalar value as an output.

Twin Delayed DDPG (TD3) is a model-free, online, off policy and actor critic algorithm. TD3 is a successor version of DDPG and the main difference is that TD3 has two Q value function (critic). It performs better by reducing the overestimation problem which is based on taking the maximum during the finding of the $y_j$ target value. The two critic models can have a different structure. However, each critic starts with different parameters even though they have the same structure. Therefore, it can unsurpassed when they have the same structure [10].

## III. Problem Formulation

Here, the main problem is an energy storage scheduling problem for the home energy management system since it includes several uncertainties and might help to decrease the grid contingency in smart grid applications. In recent years, using machine learning techniques is becoming more and more common in scheduling and control problems. In a similar way, RL is also expanding its implementation area. In this study, we create a simple storage unit case for charging/discharging scheduling with the power grid as a part of the home energy management system. In this regard, we formulate the problem as a Markov Decision Process (MDP) to minimize the operational cost over the horizon and manage the uncertainties.

### A. Markov Decision Processes MDPs

It is difficult to successfully obtain the optimal solution for the problems which includes some uncertainties. Scheduling the storage units has unknown components in terms of users' behaviour. Electricity prices and renewable generation are the main components that bring uncertainty to the system. Reinforcement learning (RL) can solve this problem in an iterative way since it has a Markov property that allows the problem formulation as an MDP with a state space, action space, transition probability, and reward $\{S, A, T, r\}$ to define the interaction between the agent and its environment. State space, action space, and observation space are created and the reward function is constituted to fulfill the necessity of the objective function and the constraints of the problem. The uncertain external inputs, such as load demand and photovoltaic (PV) energy generation and day-ahead electric prices, are fed with the current State of Charge (SoC) to find the actions as an output of the policy network.

The states, observation, and objective function can be created in different ways based on the dynamics of the problem, and these differences can cause performance changes.

*1) State:* State is current received information from the environment.

$S_t = (SoC_t, SoC_{t-1}, L_t, PV_t, G_t, cgb_t, cgs_t)$

where,

$SoC_t$ = State of charge (SoC) of the battery at time t
$SoC_{t-1}$ = State of charge (SoC) of the battery at time t-1
$L_t$ = Load demand at time t
$PV$ = Renewable generarion at time t
$G_t$ = Grid energy at time t
$cgb_t$ = Price of energy buying from grid at time t
$cgs_t$ = Price of energy selling to grid at time t

*2) Action:* Action is the decision of the agent based on the current observations.

$A_t = (B_t)$

where

$B_t$ = battery storage unit energy at time t

The environment of an RL agent can be defined as everything except the agent itself. In this problem, it is about the main two components, which are battery state of charge and net grid load. Net grid load is defined as a slack variable to meet the deficit or surplus electricity in every circumstance.

The battery energy can be described as follows:

$$B_t = \begin{cases} B_t \geq 0 & , B_t . \frac{1}{\eta_{disch}} \\ B_t < 0 & , -B_t . \eta_{ch} \end{cases} \tag{4}$$

and the grid power can be as slack variable in this formulation:

$$G_t = L_t - PV_t - B_t \tag{5}$$

where $G_t$ is the state which represents the grid power for buying or selling, $L_t$ is the load power, $PV_t$ is the renewable energy generation and $B_t$ is the battery charge or discharge power. The update of SoC is based on the following equation:

$$SoC_{t+1} = SoC_t - B_t . \frac{\Delta t}{\eta_{disch} . Q_b} - \frac{B_t . \eta_{ch} . \Delta t}{Q_b} \tag{6}$$

The representative efficiency decrease has been inserted into the model. It decreases both in the cases where the storage unit is charging beyond 80% and discharging less that 20%.

*3) Reward:* Reward is a scalar value which is taken from the environment as a consequences of the taken action.

The reward includes everything in terms of penalties and rewards to minimize the total operational cost and keep the boundaries safe and is given as follows:

$$\begin{aligned} R = \sum_{t=1}^{T} \left(cgb_t G_t - cgs_t G_t + cbdB_t\right) \\ -10(SoC_{min} - SoC_t)(SoC_t \leq SoC_{min}) \\ -10(SoC_t - SoC_{max})(SoC_t \geq SoC_{max}) \end{aligned} \tag{7}$$

The first part of the reward function represents the main objective in terms of minimizing the operational cost with respect to grid buy sell decisions and symbolic wear and tear cost for the usage of the storage unit. The third and last line represents the following battery constraint:

$$SoC^{min} \leq SoC_t \leq SoCmax \tag{8}$$

Here, it is also possible to keep the SOC level at a constant level if there is no dynamic load. With the dynamic cost, the agent is struggling in terms of both keeping the target SoC level and minimizing the cost by meeting the load demand. The result will be based on the reward function itself, since the agent cannot fulfill both objectives at the same time, which means the priority will be to follow the higher reward or lower penalty. One of the crucial parts of the RL solution is creating the reward function since the agent evaluates the taken action based on the reward itself and learns through time. Fig. 1 shows the general design which is completed in Simulink and represents how the information flows between the RL components, mainly the RL agent and environment.

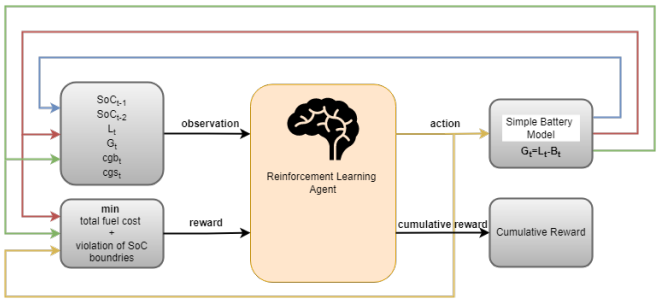


Fig. 1. The overall simulink diagram of the created RL agent-environment relation.

The main idea behind RL is that an agent interacts with its environment iteratively by performing an action based on observed state information $s_t$, which can be partially or completely observable. It receives input in the form of a reward, $R_{t+1}$ in exchange, which it utilizes to enhance its policy and hence raise the future sum of rewards. The transition to the next time step is indicated by the dashed line. In the RL literature, it mostly refers to the reward function to be maximized, but reward maximization and cost minimisation can be used interchangeably.

## IV. Experimental Results

In this section, we compare the performance of the proposed RL algorithms. Each model is trained with 500 episodes, and then simulated with the same inputs. The day-ahead electricity prices are taken from the Nord Pool markets [11]. During the training, a net load vector is extended by generating based on the uniformly distributed numbers in the interval between 25 and 25.

We assumed that the storage unit can be charged and discharged with continuous values in the range of (-50 and 50) with no constraint in terms of range since we solve the problem hourly. One of the main differences between DQN and others is the action space. As DQN needs discrete action space, not continuous, we needed to discrete the action space with 10 kW and have 11 possible actions in terms of charging and discharging decisions. In some cases, especially when the action space is relatively wide, having a discrete action set might improve the performance. The potential problem would be missing the continuous value, which means the result might not be optimal. Fig. 2 shows the net load, charge-discharge action, and grid buy-sell states based on both action and demand with respect to electricity prices. Even though it seems like the smallest total cost between the three models, the model in general does not follow the cost trend, where it mostly based on the load. The dashed red line is input, and the blue line is action. The negative blue line is charging the battery and discharging it when it is positive. The last subplot shows that the model is not violating the declared physical SoC constraints.

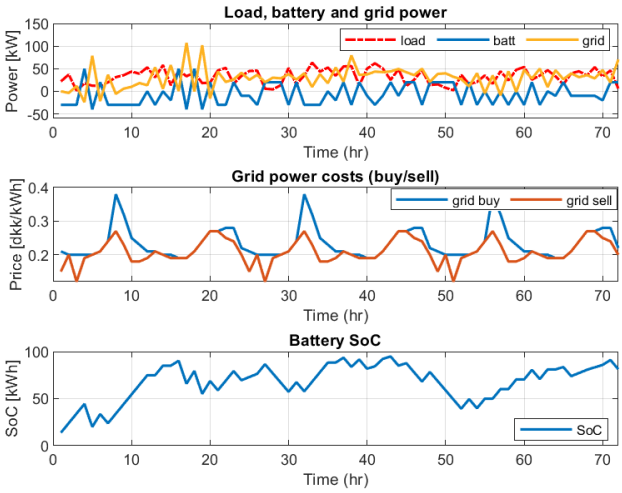


Fig. 2. The results of DQN network.

Fig. 3 illustrates the same scenario with the previous DQN solution in terms of load and electricity costs. Here, the agent takes continuous actions at the same interval and it seems that it is trying to sell power to the grid when the electricity selling cost is relatively high. Certainly it also depends on the

demand. Another observation is that the battery is discharging with a similar trend to demand, which shows the agent gives priority to storage units.

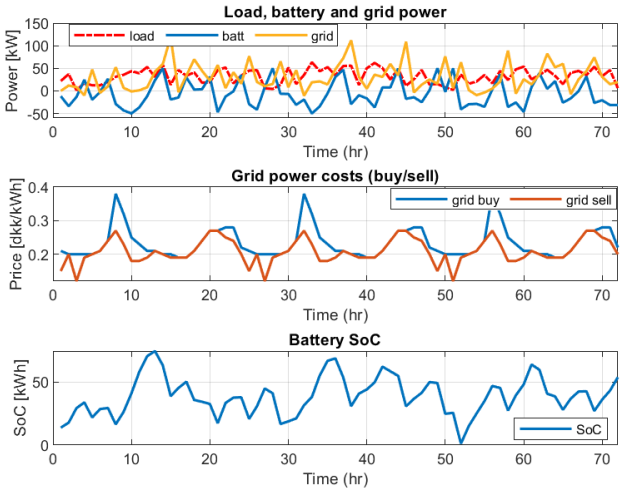


Fig. 3. The results of DDPG network.

The last Fig. 4 shows that the actions and states for the same inputs. Since there is no rate constraint for the battery usage, it changes between minimum and maximum power. TD3 is the improved version of DDPG, and we can see that it tries to use storage units as much as possible since it is a relatively cheap source. Overall, it seems that the battery is charged during the lower electricity cost. Further, it also avoids the less efficient areas, which are after 80% and before 20%.

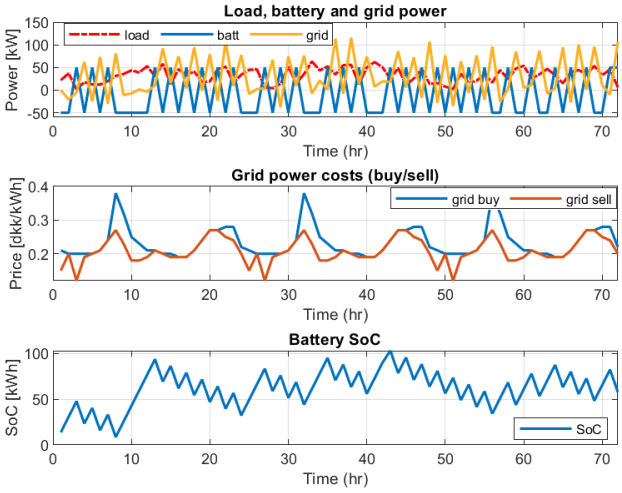


Fig. 4. The results of TD3 network.

It is challenging to reach a determined result for this problem since every simulation can produce slightly different results. Moreover, the change in the number of training sessions will also change the results completely. We stopped at 500 because the reward training plot was not changing, which led us to believe that the parameters will no longer change significantly. It can be observed that the agent learns the constraint, and by increasing the number of episodes and including real data for a year or longer, we can also reduce the operational cost.

## V. Conclusion

In this paper, three different deep RL methods, namely DQN, DDPG, and TD3, are applied to schedule three days for the home energy management system. It should be noted that further research and studies are still needed to improve the solution set. Our solution is focused on solving the problem in the MATLAB environment by following the component constraints.

For comparison, the network design is kept almost the same between these three different algorithms. However, each method might have a different number of layers and any other hyper-parameters in terms of optimizing the architecture. In future study, a crucial next step is to include forecast information for the uncertain inputs. If we feed the forecast data in advance, the agent can also learn the difference between forecast and actual data by time, which will also help to improve operation performance.

## References


[1] X. Chen, G. Qu, Y. Tang, L. Steven and N. Li, "Reinforcement Learning for Selective Key Applications in Power Systems: Recent Advances and Future Challenges," IEEE Transactions on Smart Grid, 2022., in press.

[2] S. Zhou, Z. Hu, W. Gu, M. Jiang, and X. Zhang, "Artificial intelligence based smart energy community management: A reinforcement learning approach," CSEE Journal of Power and Energy Systems, vol. 5, March 2019, pp. 1–10.

[3] Y. Liu, D Zhang, and H. Gooi, "Optimization strategy based on deep reinforcement learning for home energy management," CSEE Journal of Power and Energy Systems, vol. 6, Sept 2020, pp. 572–582. I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] D. Wu, G. Rabusseau, V. François-lavet, D. Precup and B. Boulet, "Optimizing home energy management and electric vehicle charging with reinforcement learning," Proceedings of the 16th Adaptive Learning Agents, 2018.

[5] H. Li, Z. Wan, and H. Haibo, "Constrained EV charging scheduling based on safe deep reinforcement learning," IEEE Transactions on Smart Grid vol. 11, pp. 2427–2439, May 2020.

[6] Z. Wan, H. Li, H. He, and D. Prokhorov, "Model-free real-time EV charging scheduling based on deep reinforcement learning," IEEE Transactions on Smart Grid, vol. 10, Nov 2018, pp. 5246–5257.

[7] B. Huang, and J. Wang, "Deep-reinforcement-learning-based capacity scheduling for PV-battery storage system," IEEE Transactions on Smart Grid, vol. 12, 2020, pp. 2272–2283.

[8] V. Mnih and et al., "Playing Atari with Deep Reinforcement Learning," International conference on machine learning (PMLR), NIPS Deep Learning Workshop, 2013, arXiv preprint arXiv:1312.5602.

[9] T. Lillicrap and et al., "Continuous control with deep reinforcement learning," International conference on machine learning (PMLR), NIPS Deep Learning Workshop, 2015, arXiv preprint arXiv:1509.02971.

[10] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," International conference on machine learning (PMLR), pp. 1587-1596, July 2018.

[11] Nord Pool Market Data, Accessed: 14.04.2022. https://www.nordpoolgroup.com/en/Market-data1//nordic/table.