Design of PV Renewable Energy solution with Deep Reinforcement learning

Submitted as Research Report / Honours in SIT723/SIT724

SUBMISSION DATE
T2-2021

Vincenntius Patrick Tunas
STUDENT ID 217664139
COURSE - Bachelor Of I.t. (Honours)

Supervised by: Dr. Alessio Bonti, Associate Prof Mohamed Abdelrazek

Abstract

Digital twin is a virtual replica of a physical system. It collects and processes real world data from sensors and model them based on how the actual system will operate. It will then output predictions and simulations of how the physical system be affected by certain inputs [11]. In this project digital twin will be used for microgrid. Solar energy is used to provide electricity to Deakin Waurn Ponds Campus. Solar energy is the portion of the sun's energy that is available in the surface of the earth that we can use for our application. The reason why we want to use solar energy is because it is free, it is clean and also abundant in almost all places throughout the year. Fossil fuel cost is high and using it can degrade our atmosphere. By using solar energy to provide electricity and use it to run the appliances in the building, not only that we can save cost, but we can also reduce CO2 emission in the atmosphere. One of the main problems that we are trying to solve is energy optimisation, when too much energy is not used and therefore pushed back to the grid, the grid might be unstable or energy can be wasted, the solar farm gives 7MW to Waurren Ponds campus, the campus only has 2MW of battery. There are many ways to come up with a solution to this problem. In this thesis, we will explore the use of reinforcement learning to develop a charge, discharge, dispatch and export decision making system.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Aim & Objectives

Microgrid is a system that consists of electricity sources such as photovoltaic system, batteries and an electrical load. It has the ability to operate while connected to a centralized grid, It can also work autonomously while disconnected from the main centralized grid.[6]

Microgrids purpose is to use and coordinate with the the electrical load clusters, the distributed generation units and the energy storage systems in order to supply electricity reliably to a connected host power system at distribution level in a single point.[2]

The main centralized grid is used to connect homes and buildings into a central power source (or network). If this main grid is damaged, all the homes and buildings connected to this main centralized grid will not be able to receive energy. For this reason, we need a microgrid. A microgrid can be connected with a distributed generator, battery and a solar farm.[5]

A solar panel is used to absorb sunlight as the source of energy that can be used to generate electricity or heating. The solar panel is made of solar cell that is made of silicon which has conductive layers. Each silicon atom has a strong bond that keeps the electrons in place, no electrons can flow. It has 2 layers the N-type silicon and the p-type silicon. This 2 silicons will meet in the P/N junction. In this junction the electrons can wonder creating a negative charge on one side and a positive charge on the other side. When sunlight hits the solar cell, it can knock an electron from its bond, leaving a hole. Electrons will be drawn to the n-side and the hole will be drawn to the p side. The electrons can then be captured in the form of electric current.[4]

1.2 Solar energy formula

We want to be able to estimate the electricity generated in the output of the photovoltaic system. We can use this formula:

$$E = A * r * H * PR$$

where:

- E = energy (kWh)
- A = Total area of panel (m^2)
- r = solar panel yield given by ratio of the electrical power of one solar panel divided by the area of that one panel
- H = the annual average solar radiation on the tilted panels
- PR = performance ratio

Performance ratio is the constant for losses that usually range between 0.5 to 0.9. the default value is 0.75. There are many losses that can contribute to the PR values. This losses can be from the inverter, the temperature, the DC and AC cables, shading, weak radiations and dust and snow.

The goal of the project is to find the best way to optimise the generator consumption and manage conversion of PV energy system to reduce cost for operating in the microgrid. Energy to be used can be obtained through solar panels, batteries and other resources. Using machine learning, we create an agent that can make actions based on rewards to optimise the constraints and manage energy. The chosen methodology is reinforcement learning.

1.3 Structure

Section 2 reviews the literature. Section 3 presents the research design and methodology. Section 4 describes the approach and the technical details of artefact development. Section 5 evaluates the artefacts, on the basis of research questions (RQs) in Section 5.1 and discusses the RQs in Section 6. Section 7 discusses threats to validity and Section 8 concludes the report.

2 Literature Review

2.1 Reinforcement Learning

Reinforcement learning is a methodology that consist of an agent interacting with an environment in order to learn how to map situations into actions and make decisions in the environment with a goal to maximize a numerical reward signal.[8]

In reinforcement learning, the agent aims to discover a policy of what action is best at the state the agent is in. This is achieved by the agent by learning through exploring the environment with the spaces of possible state and action pairs.An agent learns through rewards and punishments. The agent aims to find the policy that maximizes its reward. [7]

In a reinforcement learning algorithm, there is a markov decision process. Markov decision process is a method that is used to model the sequential decision marking when there are uncertainty based on the action taken by the agent. It has a number of states and action and each state will have a reward associated with it. We can say that it is how reinforcement learning problems are being represented mathematically.

A Transition function is the probability of transitioning to a certain next state from the initial state when the agent takes an action.

The discount factor is used to see if the current reward is more valuable then it is now in the future. If the value is near 0, then the future reward will not be taken into consideration because the future reward is not good but if it is near 1, it shows that the future reward will be great.

A Q-function characterize its control policy, it computes the long term cost of taking a control action when in the state.

A reward function is based of each state transition and evaluated the benefits and the penalties of taking a certain action.[6]

By understanding the policy in the end of the simulation, we can optimize our photovoltaic system better and save more energy and cost.

2.2 The Agent

The agent make use of this system by covering the electrical load with the solar PV generation from the solar panels and also energy from the grid.

The agent's objective is to manage the energy inside the microgrid while covering the load of the system. First, the agent will collect data from the microgrid and determine its current state. Then the agent will take action.

- If the power production is surplus, battery should charge in order to not waste power.
- If the PV source already has enough power to supply the load by itself, then the remaining power should be charged into the battery.

There are 3 scenarios where the battery can help:

- IF Energy surplus AND Battery Charging THEN Absorb remaining power into the battery
- IF Energy surplus AND Battery discharging THEN the agent will not take action.
- IF energy deficit AND Battery discharging, THEN Battery gives the power that is needed

[7]

2.3 The Environment Simulation

Inside a microgrid, the objective is to be able to satisfy the electrical load of the system while maximizing the utility of the solar energy generation received in the photovoltaic system.[6]

2.3.1 Pymgrid

In order to simulate the Microgrid environment, we used a python package library named 'pymgrid'.

Pymgrid enables users to generate and simulate large number of microgrids and the microgrid environment.[3]

There are 3 main levels of microgrid control. In primary control, the voltage and the frequency are controlled in a to sub-second time scale.

In the secondary level of control, it focuses on the steady state of energy control in order to correct voltage and frequency deviation. The last level of control is tertiary control. this is the control level that the pymgrid package mainly focuses on. The tertiary control level focuses on optimizing the operational cost of the micrgorid for long-term usage.

Pymgrid has 3 main components:

1. Data folder: Pymgrid make use of load datasets that comes from DOE OpenEI and is based on TMY3 weather data and pv production datasets based on TMY3. These datasets are made of a year long time-series data with a one-hour time-step for a total of 8760 points. The load and pv data are based on 5 cities from different climate zones in the US.

2. Microgrid generator class: This class is used to generate the microgrids. We first specified the amount of microgrid to generate and it will generate the specified amount of microgrids for us.

   When a microgrid is being generated, the maximum power load is generated by random, the random load file is then selected and scaled down to that value that is previously generated. The architecture of the microgrid is also selected by random.

3. Microgrid simulator class: The microgrid simulator class contains the full implementation of one microgrid, it consists of a time-series data and the specific sizing for 1 microgrid. The microgrid class has 3 main family functions which are: the control loop, the benchmark algorithms, and the utility functions Run() function moves the time-step forward by one. It takes control of the argument dictionary and returns the updated state of the microgrid. The Control dictionary centralizes all the power commands that needs to be passed down into the microgrid to operate each generator at each different time-steps. Reset() function to reset microgrid back to its initial state, empty the tracking data structure and resetting the time-step. [3]

2.4 battery-model

The model of the battery is based on implementing a discrete time system for power flow dynamics over time step interval Dt.

$R_t = R_(t-1) + R_(tstore.charge) + R_(store.charge)$

This means that the battery energy level stored at the time is the sum of energy level from previous timestep with the amount of energy charged and subtracted by the amount of energy discharged.

[6]

2.5 Battery-scheduling

There are 2 variables in the environment: solar power output and consumer load. The value of the 2 variables at timestep t defines the system step:

$s_t^i =< D_t, P_s p >$

The current timestep and the next 3 time steps will define a scenario

$scenario = t+1, t+2, t+3$

At every time step, the decision has to be made about the action to take at times, t, t+1, t+2 and t+3.

The 3-steps- ahead sequence of 4 action is decided at different for battery scheduling

$A_t = [a_t, a_(t+1), a_(t+2), a_(t+3)$

where:

a0 = discharge electricity and supply electricity for consumer's load

a1 = charge the battery and use all the electricity demanded by the consumer to supply solar generator.

The charge level has a maximum and a minimum level. There are rewards for the action taken in the environment.

If the action the agent choose is a1 which is to charge the battery and generator, the reward function is the minimum of the consumer load and the difference between the maximum charge level and the current charge level

If the agent choose a0 which is to discharge, the reward function is the minimum of the output of solar power and the current battery level.

By using a good battery scheduling, we can improve the performance of the microgrid. In order for the agent to optimise the highest amount of reward they can receive, the agent needs to choose actions between a0 and a1 at the right time for the battery scheduling. We can increase the performance by choosing the optimal sequence of actions for the 3 step ahead time period.[6]

In the 3 step markov decision process, each action sequence contains 4 actions in the battery, it can be :

- Charge,charge,charge,discharge
- Discharge, discharge, charge, discharge.

Therefore, there are 16 possible action sequence in a scenario To plan the best action, we need to plan using 3 step q-learning algorithm that take into consideration of the solar power, battery level,and the load and the objective in mind is to reduce the power consumption from the grid. The actions are chosen not for focusing on immediate reward but the future reward as well. For all the 16 possible action values, the Q value is being calculated and after the convergent value is found for all the 16 sequence. The one action sequence that has the maximum Q-value is the best sequence for that scenario.[6]

2.6 Q-learning

In Q-learning, an AI agent explores an environment. The agent will receive rewards based on the action the agent takes in the current state the agent is in. [12]

Each state is being assigned a Q-value. A Q-value is the quality of a certain action in a given state, it is our current estimate of the sum of our future rewards.[12] Based on the action the agent takes in the state the agent is in, the q value will be updated with the estimate of the sum of the future reward that we can expect to accumulate in

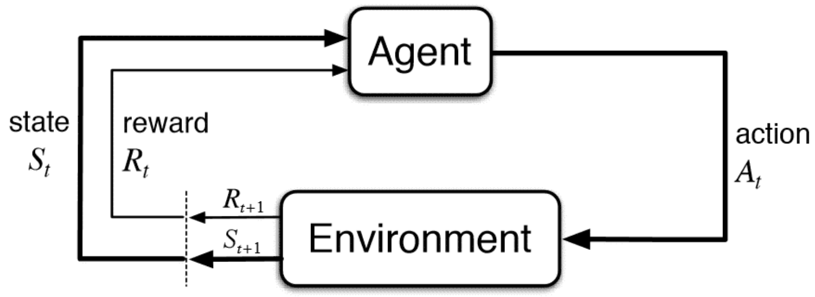


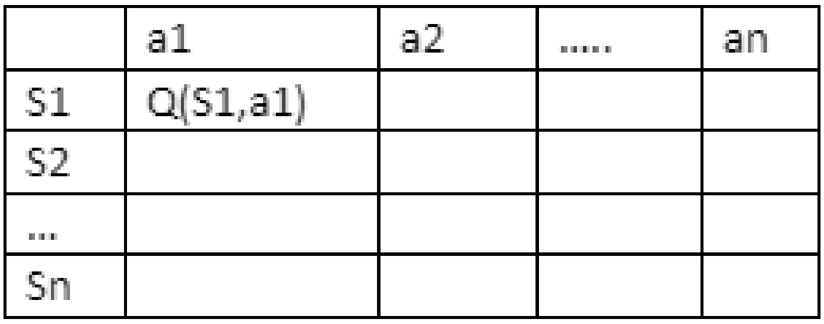
Figure 1: q-learning concept

the long run. Q-values are updated continuously as the states are revisited.[6] Q-values increases as the AI agent gets closer and closer to the highest reward.[12] Each state is assigned a Q-value.

- 1. Agent visits a state and take action to receive reward,
- 2. The reward is used to update the estimate value of action in the long run
- 3. Visit the states infinitely often and the action values are continuously updated until it reaches its limit.

[6]

Q-values are then stored inside a Q-table. Q-table is the AI agent's policy for acting in the current environment. An optimal Q-table contains the values that allows the AI agent to take the best action in any possible state which will provide the agent with an optimal path to the highest reward.[6]



	a1	a2	.....	an
S1	Q(S1,a1)			
S2				
...				
Sn				

Figure 2: Q-table

2.6.1 Temporal Differences

Temporal difference is the method we use to calculate the total amount of Q-value for the action taken in the previous state and how it has changes based on what the agent has learned about the Q-values for the current state's action. [12]

the formula for calculating temporal difference is:

$TD_(S_t, a_t) = r_t + \gamma * maxQ(S_t, a_t)$

where:

- r =reward received in the previous state
- γ = the discount factor
- Q = q value
- s = state
- a =action

The Q-values for the most recent action is always updated with what we have learned about our new state. If the new state has a good reward, then the Q-value for the previous action would be increased.

The discount factor γ enables the agent to discount the sum of future rewards value when receiving the reward in the future is lesser than receiving the same reward today.

The Q-learning agent will use this to make a decision in the net training episode.[12]

2.6.2 Bellman Equation

The Bellman equation is used to tell exactly what is the new value to be used as a Q-value for the action taken in the previous state.

The bellman equation is:

$newQ(S_t, a_t) = oldQ(S_t, a_t) + \alpha * TD(S_t, a_t)$

Inside the bellman equation, the new Q-value for the action taken in the previous state is equal to the sum of the old Q-value for the action taken in the previous state with the temporal difference for the action taken in the previous state multiplied by the learning rate.

[12]

α (learning rate) is a mechanism that is used to control how quickly an AI agent learns. This has an effect on how quick the Q-values are adjusted from the old Q-value of the previous state to the new value the agent has learned in the next state.

3 Research Design & Methodology

3.1 Methodology and Experiment designs

The main goal of using reinforcement learning is to be able to find the control policy that can minimize the sum of cost over the total amount of horizon the simulation is being ran.[1] In the experiment design, a reinforcement learning model is developed. The environment consist of the weather, the battery, the consumption, the generator, the grid and the price.

The microgrid environment where our reinforcement learning agent will operate in is developed with the pymgrid python package and our experiment algorithm we built is based on the Q-learning algorithm made by Total-RD. The link to the code is: https://github.com/Total-RD/pymgrid

3.1.1 Agent design and Action Space

In Q-learning, an agent has to select an action that can maximize the amount of reward received when operating in an environment.[7] An action space is where we define all the actions that are available for the agent to make.

With the solar energy from the sun and the energy load of the model, the agent can

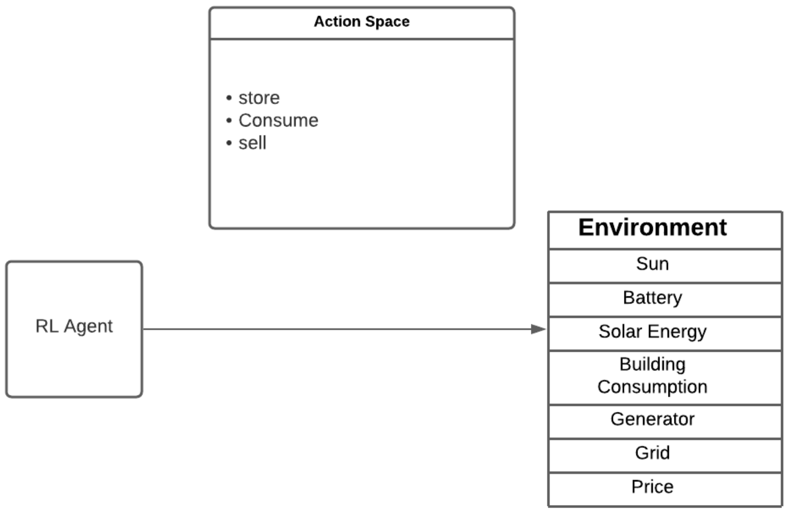


Figure 3: Reinforcement Learning Model Concept

make the decision to store the energy in the battery, consume it into the building or sell it back to the grid. The goal of the agent is to minimize costing [Grid Price] and maximize savings [the amount of electricity x price].

Action Space

The list of actions that can be taken are available in the control dictionary. However, we only set the agent to only be able to take up to 4 actions. The actions are:

- Action 0: Battery Charge
- Action 1: Battery Discharge
- Action 2: Grid Import
- Action 3: Grid Export

3.1.2 formula

There are 2 formula that we will use in our Q-learning Photovoltaic system simulation, the penetration of the microgrid and the net load of the microgrid.

The penetration of the microgrid is the percentage of the pv rated power with the load.

$$Penetration_(microgrid) = (PVRatedpower/load) * 100$$

A high peneration of microgrid is means that a high amount of the electrical load in the system can be satisfied by the solar PV power generation.

The net load of the microgrid is the amount of load in the system that has not been satisfied yet before the agent takes action.

$$Net - load = Load - PV$$

Net-load is the difference of the electrical load of the system at that current timestep with the solar pv generation received in the system during that timestep.

The agent needs to use the actions of importing from the grid and discharging power from the battery in order to satisfy the remaining load that is not covered by the solar photovoltaic generation.

3.1.3 Battery

The battery has 3 operation modes: charging mode, Idle mode and discharging mode. The energy optimization is done by designing charging and discharging battery operation modes.

Battery energy level at every time step

$E_(t + 1) = E_t + nP_(charge\delta_t) - (nP_(discharge\delta_t))/n$

the battery energy at each timestep is the sum of current energy level and the charge coming into the battery multiplied with the charge efficiency and the subtraction of the discharge of energy coming out of the battery divided by its charge efficiency.[1]

When a microgrid is generated, the variables are generated by random:

- battery state of charge
- battery power charge

- battery power discharge
- battery capacity
- battery efficiency
- battery minimum state of charge
- battery maximum state of charge
- battery cost cycle

we will know the battery's power for charging/discharging, its capacity. Its efficiency, its minimum and maximum state of charge and its cost cycle as it will be generated when the microgrid is generated.

action 0: battery charge

There are 2 scenario's when charging into battery:

Scenario 1: IF the battery's power charge is greater than 0 during charging action, THEN the battery will charge the power in and export the remaining maximum power into the grid.

Scenario 2: IF the current battery power charge is lesser than 0, THEN the battery will charge in the net load into the battery and export the remaining energy back into the grid.

action 1: battery discharge

Scenario 1: IF the battery's current discharge power is available, THEN the agent will use the power from the battery to discharge and import the remaining energy from the grid in order to satisfy the load

Scenario 2: if battery's discharge power is negative, THEN It will use the net-load to discharge, and the remaining load will be satisfied by importing from the grid.

Grid

When we initialize the grid, the power for importing and exporting from the grid is generated. In Pymgrid, we get to choose 3 types of grid in the arcithecture. we can use a normal grid, a weak grid or no grid at all.

action 2: Import power from grid

When the agent chooses this Action, the agent import the exact amount of power that is needed to satisfy the net-load(difference between the load and the amount of pv power that is generated) of the system. This decision will increase cost in the operation.

action 3: Export power from grid

When the agent chooses to export power from the grid, it exports the current net load of the system back into the grid. This action lowers down the overall cost. Usually this action is used when the load of the system has been satisfied and the power in the system currently is excessive.

3.1.4 Initialize the Q-table

The Q-table is initialized in the algorithm to store the Q-values. This Q-table will we the agent's policy for acting in the current environment. By achieving an optimal Q-table, the agent can know the best action to be taken in any possible states. This provides the agent with an optimal path to the highest reward. [12] The Psuedocode explains how the Q-table is implemented

```
psuedocode
init_Q_Table(microgrid, action_chosed):
    net_load -> microgrid.forecast_load - microgrid.forecast_pv
    state - > empty
    Q -> empty
    iterate through range from minimum net load to maximum net load
        for all battery state of charge level from min to max:
            round off
            append to state
    for all the states:
        initialize to 0
        for all actions:
            set the state to 0
    return Q
```

Figure 4: psuedocode for Q-table

3.1.5 Exploration exploitation strategy

The strategy that we will be using for our agent is an exploration/exploitation strategy. In the beginning, the agent doesn't know what action is the best to take for each state that the agent encounters. Therefore, the algorithm will need to explore for a certain number of rounds. [7]

An epsilon-greedy action selection behaviour is implemented into our agent. An epsilon-greedy is used in order to balance exploration and exploitation of the agent. [10]

An epsilon value refers to the probability of the agent to explore or to exploit. This epsilon value decrease as the exploration goes further which causes the agent to have a very explorative behaviour in the beginning and becomes more highly exploitative as the training progresses. This is known as the greedy decreasing strategy. [10]

3.1.6 Functions

This functions are used to implement the exploration strategy with a greedy decreasing behaviour.

update epsilon: this function is used to decrease and update the epsilon. As the exploration continues, the epsilon decreases. This leads the agent to exploit more as the training progresses.

epsilon decreasing greedy: we set a random number, if the random number value is above 1-epsilon, then we return the random action that the agent is taking.

max dict : find the maximum value in the dictionary, the function returns the max key and max value after the entire dictionary has been searched through. In this algorithm, we use this function to search in the q table, which state has the highest Q value. If the certain action taken in that state has the highest value out of the rest, the agent will take this action.

3.1.7 Agent Training

We create a function to start training our agent in Q-learning. First, we initialize all the variables that are to be used.

Number of actions = 4

Q = Q-table taking the action and microgrid name as a parameter

Number of states = the total length of Q

Number of episodes = 100

Alpha = learning rate

gamma = discount factor Then, the algorithm Iterate through all the number of episodes and Train the agent every episodes and print the value for the end of each episodes after all the episodes are finish.

Inside each episode, run the training for the total number of horizon(Horizon is the total number of hours we want our agent to go through).

For each horizon the agent goes through, the agent takes an action. The action taken in the microgrid environment changes the status of the environment. The reward is subtracted by the cost of the action. The cost value is added to the total episode cost. The state that the agent arrive in is based on the current net load and the state of charge of the agent. The agent will then choose the action that leads to the best state in the table using max dict() function. Then, we calculate the temporal difference error (td error). Temporal difference error indicates the difference between the agent's current estimate and target value. After all is done, we enter the new Q-table state.

After the total number of horizon is finished and each episode is complete, the epsilon is decreased and the agent moves forward to the next episode. This process is repeated until it has finished through all the defined number of episodes. The q table is then returned in the end of the process.

3.1.8 Agent Testing

After the agent is being trained for the specified number of episodes. Its Q-table has been filled with good Q-values. The agent is then tested in the environment. Through this function, it also displays to us in each time-step, what action did the agent take, what is the state that the agent arrived in and what is the cost of the agent's action.

3.2 Data Collection

By using the reinforcement learning model, the data we collect are the agent's action and its result. A graph will be generated in the end to show the consumption rate and how well the agent has optimized cost and manage energy.

4 Artefact Development Approach

4.1 Data Collection

Efficiency of solar panel is dependent on the temperature of the solar panel. The general range of the solar panel temperature is between 15 and 35 . The temperature standard is 25 . If the temperature goes above 25 , every increase of temperature above 25 will reduce the output power by half. If the weather temperature outside is 40, then the solar panel temperature is roughly 65. The temperature ecoefficiency is -0.5%/after temperature goes above 25 .[9]

This is the reason the first data that are to be collected is weather data. The weather that we plan to monitor is from Waurn Ponds as the deakin campus is in waurn ponds.

The secondary data that we collected are weather data. this data is obtained by using the openweathermap API. The aim is to find the answer of how weather influences the photovoltaic system and how the reinforcement learning agent can adapt based on the weather. The location that we are going to monitor weather data of is Waurn pond since the campus is in there. The data that is stored will be used for our reinforcement learning model.

The data that needs to be obtained are the historical data, the current data and the prediction data.

We create a page where we can identify the current weather in waurn ponds using the Openweathermap API. Using an openweathermap api, we are able to collect data regarding the weather at certain locations at the current day. The weather data consist of weather condition (such as rainy and cloudy, temperature, pressure, humidity and wind data (wind speed, wind direction and wind gust). we store the data into a firebase database. Data will be collected every 4 days Openweathermap provides a forecast feature for 5 days every data every 3 hours. We want to gather this data in order to get an insight on the solar panel's performance for the next five days.

With the Pymgrid package, it enable us to use the PV and load dataset. In this experiment we will be working with 3 datasets each with different PV and Load forecast.

4.2 Artefact development

For our Artefact, we created a baseline model where we can implement our own strategies inside the Q-learning algorithm. When we use our strategy, the agent's policy and decision making in the environment changed.

Since sun is available during daytime and the pv(photovoltaic) generation is at its highest. During Night-time, there is no sun and therefore there is little to no pv generation during nighttime.

Therefore our first strategy would be to charge the battery during daytime when solar PV generation is available and use the battery power to cover the load during nighttime.

The pv generation of the system is available during 7am to 5pm. the hours when pv generation is at its highest is from 9am to 4pm in the afternoon. The strategies that we implemented in the microgrid simulation are the following below:

4.2.1 Strategy 1: Use strategy on the agents training

To avoid energy loss in the system, During the agents training,we will set the agent to use the charging actions every daytime during the hours where pv source are at its highest which are from 9am to 4pm. During the testing stage, the change in Q-table values because of this decision made during training has an impact to the decision that the agent will pick in the next state.

4.2.2 Strategy 2: Use strategy on the agents testing

In this strategy, we force the agent to charge from 9 am to 4 pm everyday. In this decision, the agent is not able to make his own decision everyday between 9am to 4pm in the simulation.

4.2.3 Strategy 3: striving for net load zero

In this strategy, we don't indicate the agents decision based on the time of the day. Instead, we modify the agents action based on the current net load of the system.

IF the net load is greater than or equal to zero, THEN the agent's action is either discharging from the battery or importing energy from the grid

IF the net load is negative, THEN the agent's action is to charge into the battery and export energy into the grid.

4.2.4 Strategy 4: charge into the battery only if net load is negative

Instead of focusing in scheduling actions, in this strategy, we tell the agent during training to charge into the battery only if the net load is currently negative

4.2.5 Strategy 5: export into the grid only if net load is negative

In this strategy, we implement a strategy during the agents training for it to export energy only if the net load is currently negative.

5 Empirical Evaluation

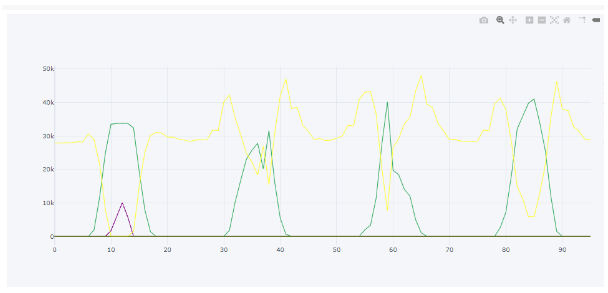
5.1 Research Questions (RQs)

5.1.1 1. How to optimise the electrical power consumption in the microgrid to reduce cost?

We can optimize the electrical power consumption in the microgrid by the use of reinforcement learning methodology. For all the experiments conducted, the reinforcement learning agent performs best and save the most cost when the agent uses its own policy without alterations.

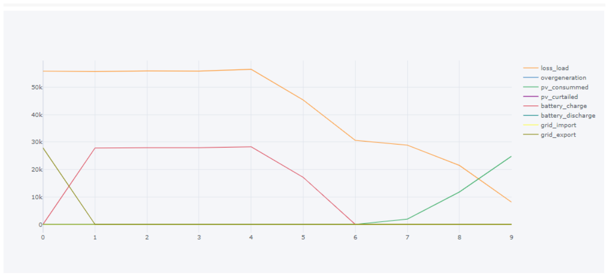
We did strategy tracing and realize that our agent never charges energy into the battery or export any power back into the grid. With the agent's policy, the microgrid is more well managed.

The main reason why our agent's policy works the best is because the simulation can only be run for a maximum of 4 days. Within 4 days, Cost is still low and the agent still have enough power in his battery capacity. This is the reason why within the 4 days of simulation, the agent never chooses to charge power into the battery or export power into the grid and it is still able to minimize cost better than all our strategies.



The graph shows the actual production with the agent's policy

The second graph shows the actual production when we use strategy 2.when we use our strategy, to charge power to the battery during daytime, there is more loss in the load



and the cost increases.

Since our microgrid simulation can only be run in 4 days, inside the microgrid, cost is still low and electrical power inside the battery capacity is still high. Therefore it is not worth it to charge electricity into the battery according to the agent.

When we use our strategy, the agent charges electricity into the battery while the capacity of the battery is still high. This is not good as it can make our battery overcharge and reduce our battery's life cycle.

The best way we can reduce cost in our microgrid simulation is to let our agent use his own decision and make his own policy. After a few simulation days, when the cost gets higher and the battery's capacity is low in power, Then we switch into our strategy that is to charge into the battery during peak solar pv generation hours.

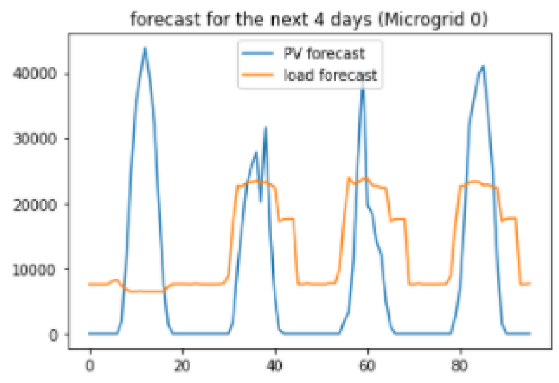
5.1.2 2. How to Manage Battery power?

In order to manage energy in the battery well and improve its lifecycle, the battery needs to charge and discharge at the right time and at the right situation. By using a good battery scheduling, we can improve the performance of the microgrid.[6]

Based on our experiment however, we learn that it is better for us to set a threshold instead of using battery scheduling. A threshold where if the current net-load in the system is negative, then we charge into the battery. If this threshold is not met, the agent should not charge power into the battery.

We conducted an experiment, for this experiment, we're using a different dataset. The dataset we use have the following load and pv forecast.

We run the algorithm normally first. Then we implement strategy 4, strategy 5 and strategy 1.The result is then recorded into the table below: Based on the table, we can make a comparison. Strategy 4 and 5 completely outperforms strategy 1 and 2.



	original	Strategy4	Strategy5	Strategy1
Day1	12862.8 €	1574822.4€	1302767.6€	12862.8 €
Day2	37144.0 €	1764196.6€	1394960.5€	37144.0 €
Day3	65716.6 €	2163669.2€	1442075.1€	1741938.0 €
Day4	88550.0 €	2806114.4€	1843681.1€	4349624.5 €

We are able to save more cost by charging or exporting when the net-load of the system is negative rather than relying on a time schedule of when the agent should charge the battery.

A graph generated below shows the cumulative sum of cost the agent uses when using strategy 4 and strategy 5.

```
mg4.print_cumsum_cost()
```

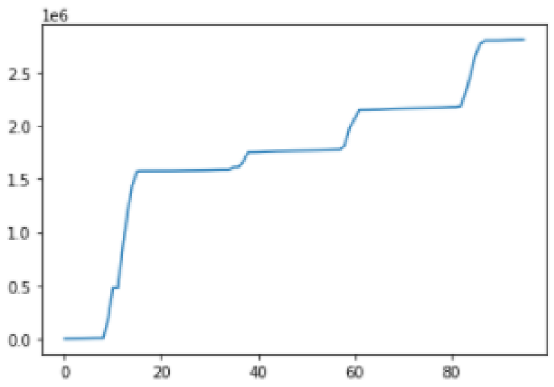


Figure 5: Strategy 4 cost

```
mg4.print_cumsum_cost()
```

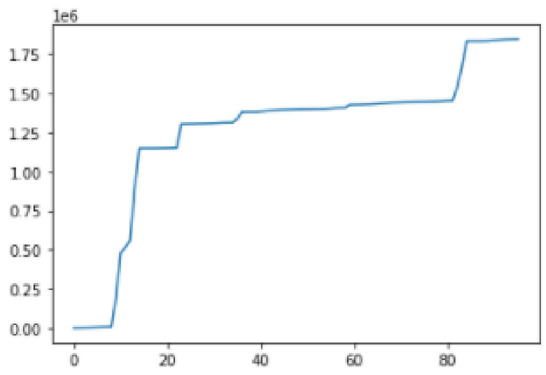


Figure 6: Strategy 5 cost

For strategy 4 and strategy5, we can see the number above the plot 1e6. This is a standard scientific notion. It showed that the overall scale factor for the y axis. This means that the cost in the current number in the y-axis multiplied by 10 to the power of 6.

In the agent's normal policy, the agent relies mostly on importing power from the grid and discharging from the battery action. The agent never charges power into the battery even when the Net-load of the system is negative.

Cost keeps increasing as microgrid works longer. This is normal and is expected. The graph shows the strategy 5 is less costly than strategy 4. This means that exporting energy back to the grid helps reduce cost more than charging the energy into the battery.

The graph for cumulative sum shows that the Agent's policy cost less than our strategies, however, the cost of the agent's policy gradually increases and never stops. If we use the agent's policy for longer period of time , it will be costly.

In strategies 4 and 5 however, there are periods where the agent is able to stop the total cost from increasing. This is shown in the flat lines that occurs over a period in the graph for strategy 4 and strategy 5.

The graph shows that there are periods where the agent is able to stop the cost from increasing when using strategy 4 and strategy 5. When using the agent's policy however, the cost keeps rising every hour the simulation is run. The cost never stops increasing.

In conclusion, if we are going to operate and generate electricity with the microgrid system for over a long period of time, we can't just rely on the agent's policy only. At some point during the operation, we need to switch and operate the microgrid using strategy 4 and strategy 5. However, If we are only going to use the microgrid system for a short period of time, the agent's policy will work better.

6 Results & Discussion

6.1 The first experiment

The first 2 strategies are applied into an algorithms with different datasets. The forecast for the load and PV datasets are below:

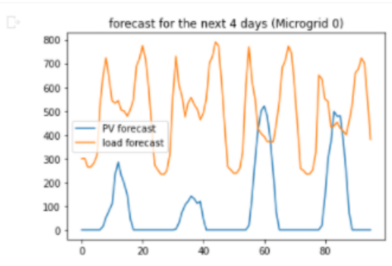


Figure 7: dataset 1

This algorthm is run for 24 hours in the simulation time. When strategy 1 and 2 are implemented in the algorithm with dataset 1, strategy 2 is able to save up more cost than strategy 1. When using strategy 1, the agent used up 83162.3 euro for cost while in strategy 2, The agent used up to 64307.5 euro.

We can assume the reason is because the agent commits fully with the strategy to charge during daytime in strategy 2.

In the original reinforcement learning algorithm, the agent never seems to choose to charge electricity into the battery. The reason for this might be because in the dataset we are using, the pv source never seems to be able to fulfill the load of system at all. Therefore, all the PV energy are being used up for the load and being charged into the battery which will lead to the system having an increase in net load and therefore increase the cost. This is the reason why in the next step, we experiment with a different dataset where the pv source is able to satisfy the load of the system at least during for a minimal range of time.

6.2 The second experiment

We apply strategy 1 and strategy 2 in a different dataset. The load and pv forecast of the dataset is below

In this experiment, we run the algorithm for 4 days instead of just 1. Compared to the previous dataset, this dataset has a higher load and a higher solar pv generation. On the first and second day, there is a positive net load, the amount of solar pv generated

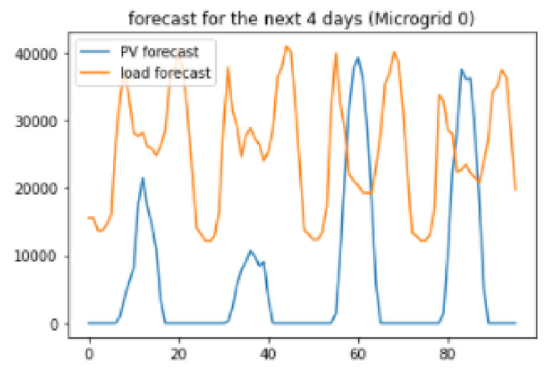


Figure 8: dataset 2

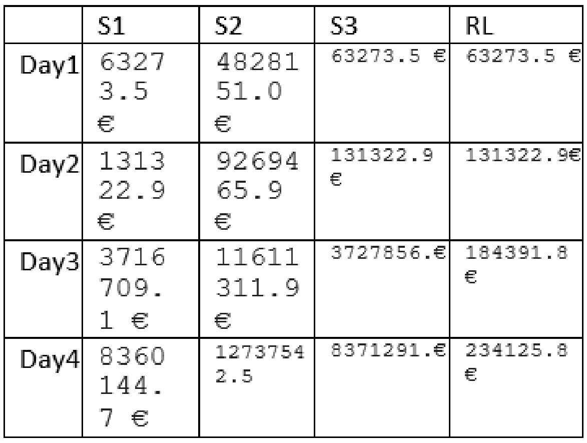
isn't high enough to supply the load of the system. On the third and fourth day however, the solar PV generation is very high that it exceeds the amount required to satisfy the load.When this happens, the agent needs to charge the excess power into the battery in order to save cost.

we record the performance of the agent when the algorithm is run normally, when using strategy 1 and when using strategy 2. Based on each performance, we make a comparison.

We also implemented a new strategy. In strategy 3, we don't indicate the agents to take a certain decision based on the time of the day. Instead, we modify the agents action based on the scenarios. The goal of this strategy is for the agent to strive for a net-load of zero. This means that all the electrical load of the building's system to be satisfied and any excessive unsed power in the system will either be charged into the battery or exported back into the grid. It makes sense that this strategy should be the most expensive as we are aiming to completely satisfy the electrical load of the building fully.

Unlike the first experiment, strategy 2 performs better than strategy 1.

The reason is because in the first experiment, the algorithm is only run for 1 day. The pv generation is always below the load. In this experiment however, the agent works for 4 days in the simulation and the condition changes over the day. There are days where the pv generation exceeds the load of the system. Since the situation changes in different days, the strategy where the agent can adjust to this changes is best. This is the reason why strategy 1 is better than strategy 2, because strategy 1 enables the agent to adapt to the changes in the environment.



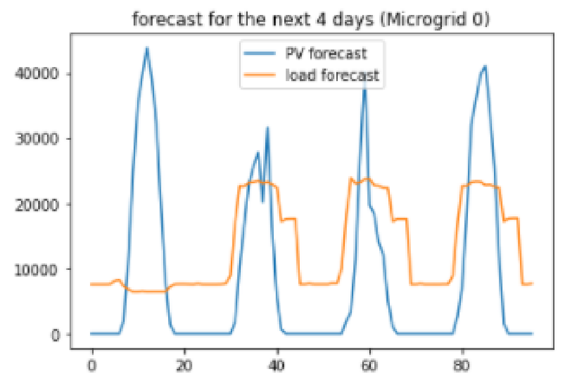
	S1	S2	S3	RL
Day1	63273.5 €	4828151.0 €	63273.5 €	63273.5 €
Day2	131322.9 €	9269465.9 €	131322.9 €	131322.9€
Day3	3716709.1 €	11611311.9 €	3727856.€	184391.8 €
Day4	8360144.7 €	12737542.5	8371291.€	234125.8 €

6.3 The third experiment

For the first and second experiment, our strategy always mainly rely on scheduling. In the third experiment we will not focus on scheduling on which hour is a certain action best taken. Instead, we focus on taking a certain decision if the net-load is negative and there is an unused excess power in the system.

The 2 new strategies are strategy 4 and strategy 5. In strategy 4, the agent will charge into the battery if the net-load value is negative. In strategy 5, the agent will export energy back into the grid if the net-load value is negative.

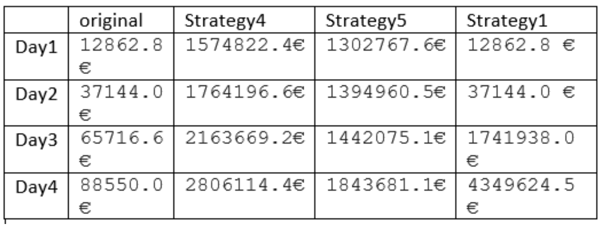
In this experiment, we used a different load and PV dataset. The forecast for the load and PV dataset for this experiment is shown below:



In this dataset, for all the 4 days, the solar PV generation that is obtained is able to satisfy the load of the system and provide the system with excess power.

In this experiment, we run the algorithm normally first. Then we implement strategy

4, strategy 5 and strategy 1.The result is then recorded into the table below: Based



	original	Strategy4	Strategy5	Strategy1
Day1	12862.8 €	1574822.4€	1302767.6€	12862.8 €
Day2	37144.0 €	1764196.6€	1394960.5€	37144.0 €
Day3	65716.6 €	2163669.2€	1442075.1€	1741938.0 €
Day4	88550.0 €	2806114.4€	1843681.1€	4349624.5 €

on the table, we can make a comparison. For all the experiments conducted, the reinforcement learning agent performs best and save the most cost when the agent uses its own policy without alterations.

However, strategy 4 and 5 completely outperforms strategy 1 and strategy 2. We are able to save more cost by charging or exporting when the net-load of the system is negative rather than relying on a time schedule of when the agent should charge the battery.

7 Threats to Validity

One of the threats to validity is that the microgrid simulation we have is not able to run longer than 4 to 5 days at its best. Because of this flaw, we can only experiment on the first few days of the simulation. The agent can only train and learn about the environment for just 4 days and can only build its policy based on just that four to five days of training.

If we are able to run the simulation for a longer period of time, the agent will face a lot more changing situations in the microgrid. By facing more situations, more changes in the variable, the cost will be much higher as the environment runs longer, the electrical power in the battery capacity will keep changing overtime.Therefore the agent's training will improve because it will face more different states where the variables the agent is working with in the environment isn't very good. This will improve the agent's decision making.

By being able to run our microgrid simulation for a longer period of time, we can also see how our strategies effects our agent and its performance.

As explained in our research design, the pymgrid package that we use to simulate our microgrid environment generates load and Solar photovoltaic generation files by random.

The dataset that is used in the environment will change if the code is run after a certain time span (usually a day). If the simulation is run at the same day, the dataset will not change.

When we run the simulation at a different day, the dataset that we will be working with changes and we have no control over it.Therefore the result we receive for running a simulation at a certain day will be different with the result that we receive when running the simulation on the next day.

Since most of our result are received by comparing different performance of different strategies, we can't compare a strategy that we run with the different strategy we run on a different day.

We have to run all strategies we have on the same day in order for us to be able to compare the different strategies fairly because all the strategies are working on the same dataset.

If we are to make changes to our code and to our simulation, the dataset will be different than the dataset we use before the changes is made on our code.

The randomness of the dataset generated by the microgrid also leads to another threat to validity. The dataset that pymgrid package generate for us to use in the microgrid simulation has an effect on the duration of how long the agent can operate in the environment. Often, a certain dataset can cause an agent to not be able to run more than 30 hours in the simulation. Often, a certain dataset that pymgrid generate allows the agent to work for more than 48 hours in the simulation but less than 96 hours.

We are not able to compare different strategies that works at a different time duration. All the strategies has enable the agent to work for the same amount of time in order for it to be able to be compared.

8 Conclusion & Future Work

The conclusion from this research is that setting a threshold for our agent when operating in our microgrid environment simulation is better than using battery scheduling for the agent to make a certain decision. Its better for us to set a threshold, if netload is negative, our agent needs to choose whether to export the excess power into the grid or store it inside the battery in order to prevent electrical power loss.

The problem with battery scheduling approach is that because sunlight is available, doesn't always mean the solar power we receive will be high enough to satisfy the electrical load of the system. Therefore charging this solar PV generation into the battery will make the system not be able to satisfy the load.

It depends on the situation. If we do not have a forecast data on the electrical usage of the system for the next few months and the forecast data on the solar Photovoltaic generation for the next few months, then it is better for us to use battery scheduling and set the agent to charge energy during the hours when solar PV generation are available and is at its highest.

If we have a forecast data on the electrical usage of the system for the next few months and the forecast data on the solar Photovoltaic generation in the next few months, then we need to implement strategy 4 and strategy 5. It is better for us to use the threshold to manage the agent's decision.

The question of which strategy will save more cost and which strategy will be able to manage the energy consumption better depends on the length of the period of time will we operate using the microgrid.

As it was mentioned in our emparical evalation, when we plan on using the microgrid system for a short period of time. Using the normal agent's policy will save more cost and be able to manage the energy consumption better. But in the long run, the agent's policy is not feasible for our usage. This is because we will receive a higher cost and we may ran out of power.

If we want to use the microgrid simulation for a long period of time, strategy 4 and strategy 5 needs to be used as it is able to save more cost by charging excessive and unused power in the system into the battery. By exporting the electricity back into the grid, we will be able to save up more cost.

The optimal strategy will be to switch between strategy 4 and strategy 5. If the battery capacity is already fully charged, charging into the battery will cause it to overcharge and therefore reduces the battery's life cycle, this isn't good. Therefore when battery capacity is full and there are excessive electrical power, the agent should export it back into the main grid and save cost.

We also learn that it is better to save our strategy for the future days rather than using it right at the beginning of the simulation.

The best way to manage energy in the microgrid is to use the agent's original policy for the first few days. Then as the cost gets higher and the power capacity of the battery becomes low, then we change the agent's strategy and switch to our optimal strategy that is mentioned above.

We can set a threshold, if the cost reaches a certain value or the battery power capacity reaches a certain low level, then switch from agents policy into our optimal strategy(implementing strategy 4 and 5).

Through strategy shifting mechanism, shifting from our agent's normal policy to our optimal strategy at the right time and at the right situation is the key to managing electrical power in the microgrid and reduce cost.

8.1 Future Work

If we have more time, we plan on using our strategies and our hypothesis in the real world, specifically in deakin's waurn pond campus. We would like to use our strategy in a real dataset instead of a dataset from pymgrid package. We want to be able to implement our strategies with the deakin's waurn ponds dataset.

We can improve our result in the future by looking for a solution in how we can make the microgrid simulation run longer than just 4 days. We want to be able to run the simulation for at least over a month.

We can also improve our solution in the future by implementing a strategy shift mechanic where we can switch strategy in the middle of the simulation runtime.

Therefore, inside the environment, the agent can start using his own policy and he can make his own decision. After 6 to 8 days in the simulation, as the agent keeps discharging power and importing power from the grid to satisfy the system's electrical load, the cost will increase and the power left in the battery will be low, then we can switch into our optimal strategy.

We can set a threshold, when the battery capacity is full and there is excess electrical power, the agent will switch from strategy 4 to strategy 5.The agent will export the excess power back into the main grid instead of charging it to the battery.

By identifying the right threshold where the strategy has to switch when that threshold is reach, knowing what strategy to switch to and what value in the battery capacity, cost or other unknown variables in the future should trigger the strategy shift. Then we can optimize the microgrid's electrical consumption and minimize cost.

References

[1] G. D. Brida V. Mbuwir, F. Spiessens, Self-learning agent for battery energy management in a residential microgrid, (2018).

[2] A. H. E. C. A. C. R. I. M. K. S. A. H. H. O. G.-B. M. S. R. P.-B. G. A. J.-E. N. D. H. Daniel E. Olivares, Ali Mehrizi-Sani, Trends in microgrid control, (2014).

[3] A. H. R. A. Gonzague Henri, Tanguy Levent and P. Cordier, pymgrid: An open-source python microgrid simulator for applied artificial intelligence research, (November 2020).

[4] R. Komp, How do solar panels work?, Jan 6, 2016. https://www.youtube.com/watch?v=xKxrkht7CpY.

[5] A. Lantero, How microgrids work, (17 June 2014).

[6] R. Leo, R. S. Milton, and S. Sibi, Reinforcement learning for optimal energy management of a solar microgrid, (2014), pp. 183–188.

[7] G. V. . A. I. D. Panagiotis Kofinas, Energy management in solar microgrid via reinforcement learning using fuzzy reward, (2018), pp. 97–115.

[8] D. Pandey and P. Pandey, Approximate q-learning: An introduction, (2010), pp. 317–320.

[9] D. V. N. Rahul Chauhan, Write white paper seamless, (2021).

[10] samishawl, Epsilon-greedy algorithm in reinforcement learning, (2020). https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/.

[11] K. Shaw and J. Fruhlinger, What is a digital twin and why it's important to iot.

[12] D. D. Soper, Foundations of q-learning, 2020. https://www.youtube.com/watch?v=___t2XRxXGxI.