sensors

MDPI

Article

# Maximum Power Point Tracking of Photovoltaic System Based on Reinforcement Learning

**Kuan-Yu Chou *, Shu-Ting Yang and Yon-Ping Chen**

Institute of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan; ystfc72.eed02@nctu.edu.tw (S.-T.Y.); ypchen@cc.nctu.edu.tw (Y.-P.C.)

* Correspondence: eric210092.ece01g@nctu.edu.tw

Received: 24 October 2019; Accepted: 14 November 2019; Published: 19 November 2019


**Abstract:** The maximum power point tracking (MPPT) technique is often used in photovoltaic (PV) systems to extract the maximum power in various environmental conditions. The perturbation and observation (P&O) method is one of the most well-known MPPT methods; however, it may face problems of large oscillations around maximum power point (MPP) or low-tracking efficiency. In this paper, two reinforcement learning-based maximum power point tracking (RL MPPT) methods are proposed by the use of the Q-learning algorithm. One constructs the Q-table and the other adopts the Q-network. These two proposed methods do not require the information of an actual PV module in advance and can track the MPP through offline training in two phases, the learning phase and the tracking phase. From the experimental results, both the reinforcement learning-based Q-table maximum power point tracking (RL-QT MPPT) and the reinforcement learning-based Q-network maximum power point tracking (RL-QN MPPT) methods have smaller ripples and faster tracking speeds when compared with the P&O method. In addition, for these two proposed methods, the RL-QT MPPT method performs with smaller oscillation and the RL-QN MPPT method achieves higher average power.

**Keywords:** maximum power point tracking (MPPT); photovoltaic (PV) system; reinforcement learning; Q-learning; Q-network

## 1. Introduction

Sustainable energy such as solar energy is often seen as one of the solutions to reduce pollution caused by thermal power generation. A photovoltaic (PV) module is able to convert solar energy into electrical energy without generating greenhouse gases and coal dust, and it is wildly used since the deployment is relatively easy compared to other sustainable energy sources such as tidal energy and biogas energy. However, low efficiency is the main drawback of a PV system. Therefore, several maximum power point tracking (MPPT) methods are proposed in order to extract maximum power from the PV module. The perturbation and observation (P&O) method is one of the most common MPPT methods and can be implemented model-free [1,2]. However, a large size perturbation setting will lead to large oscillations near the maximum power point (MPP), while a small step size perturbation setting will slow down the tracking speed. Therefore, several adaptive methods are proposed to improve the P&O method [3–5]. The adaptive P&O method basically modifies the step size based on the amount of the power difference between two perturbations. However, to achieve the best performance, the ratio between the step size and power difference needs to be tuned according to the actual model. Additionally, several fuzzy P&O methods are proposed to perform the MPPT [6,7]. Although fuzzy logic control is a model-free control method, expert knowledge is required when designing the fuzzy parameters.

Capable of performing model-free control, reinforcement learning (RL) [8] is widely used in solving control problems because it can learnt by interacting with the system without prior knowledge of the system model. There are two similar MPPT methods based on RL for PV system proposed in [9] and [10], and a Markov decision process (MDP) is used as the framework to describe the problem. The states are defined by the moving direction and the position of the operating point relative to the MPP. The action is the choice of variable step sizes, and the reward is defined as the power difference before and after the action is taken. Kofinas et al. [11] also proposed a solar MPPT method based on RL. The discretized current and voltage value and a parameter of judging whether the operating point is at MPP are used as the system description. The perturbation step size can be chosen appropriately according to the interacting experience with the system. The MPPT method in [11] is similar to one of the proposed methods, reinforcement learning-based Q-table maximum power point tracking (RL-QT MPPT), however, it is designed to be learnt online, which causes several oscillations during the tracking process. In addition, the above mentioned solar MPPT methods approached by RL are all implemented by constructing the Q-table, which may lead to the problem of generalization representation. Using a neural network as an approximation of the Q-table was first introduced by [10], and the experience replay technique is also proposed by [12]. Mnih et al. [13] proposed a fixed target Q-network technique to stabilize the training process. In this paper, two RL MPPT methods are proposed by using the Q-learning algorithm. One constructs the Q-table (RL-QT MPPT) and the other adopts the Q-network (RL-QN MPPT). These two proposed methods do not require the information of an actual PV module in advance and can track the MPP through offline training in two phases, the learning phase and the tracking phase. From the simulation and experimental results, it is expected to outperform the P&O method since the step size can be chosen according to the learned perturbation experience.

At first, this paper will introduce the model of the PV module in Section 2. Then the P&O method and proposed reinforcement learning-based MPPT methods are described in Section 3. In Section 4, the simulation and experimental results show the comparison of the traditional P&O method and the proposed methods to prove its performance. Finally, the conclusion is given in Section 5.

## 2. Model Description of PV Module

When a PV module is exposed to the sunlight, the electrons inside it will absorb the energy and jump to a higher energy state. Some free mobile electrons will be released through a connected wire and thus form a current. This phenomenon is known as the photovoltaic effect [14].

A solar cell is a device to generate electrical power based on the photovoltaic effect. Figure 1 shows the single-diode model [15] of a solar cell, where $I_{ph}$ is the photo-generated current, $I_{D_S}$ is the current through the diode $D_s$, $V_{D_S}$ is the voltage across $D_s$. $I_{sh}$ is the current through shunt resistor $R_{sh}$, and $I$ is the current through series resistor $R_s$. In addition, $I$ and $V$ are the output current and output voltage of the solar cell, respectively.

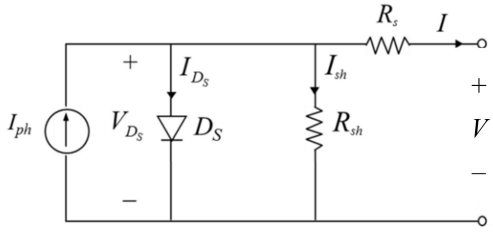



**Figure 1.** Single-diode model of a solar cell.

According to Kirchhoff's current law, the output current can be expressed as

$$I = I_{ph} - I_{D_S} - I_{sh} = I_{ph} - I_{D_S} - \frac{V + IR_S}{R_{sh}}. \quad (1)$$

The current $I_{D_S}$ can be expressed by the Shockley equation below:

$$I_{D_S} = I_0\left(e^{\frac{qV_{D_S}}{\eta KT}} - 1\right) = I_0\left(e^{\frac{q(V+R_SI)}{\eta KT}} - 1\right) \tag{2}$$

where $I_0$ represents the reverse saturation current, $q$ is the elementary charge, $K$ is Boltzmann's constant, $T$ is the temperature and $\eta$ is the diode ideality factor. Note that $\eta = 1 \sim 2$ and $\eta = 1$ for an ideal diode [16].

Substituting (2) into (1) yields the *I-V* characteristic expression of a solar cell model shown as

$$I = I_{ph} - I_0\left(e^{\frac{q(V+R_SI)}{\eta KT}} - 1\right) - \frac{V + IR_S}{R_{sh}}. \tag{3}$$

The photo-generated current $I_{ph}$ is affected by solar irradiance $S$ and environment temperature $T$ as below:

$$I_{ph} = \frac{S}{1000}[I_{scr} + K_i(T - T_r)] \tag{4}$$

where $I_{scr}$ is the short circuit current, $K_i$ is the temperature coefficient of the short circuit current, and $T_r$ is the reference temperature.

A PV module usually consists of several solar cells in series or in parallel, i.e., composed of $M$ rows and $N$ columns of solar cells as depicted in Figure 2. The output voltage $V_{sm}$ of $M$ solar cells in series and the output current $I_{sm}$ of $N$ solar cells in parallel are respectively given as

$$V_{sm} = MV \tag{5}$$

$$I_{sm} = NI. \tag{6}$$

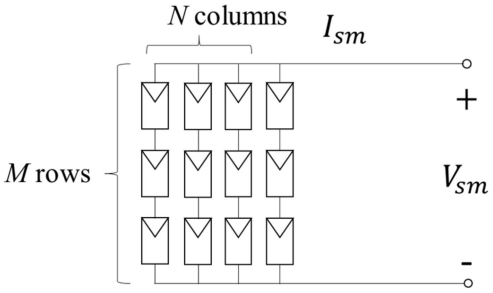


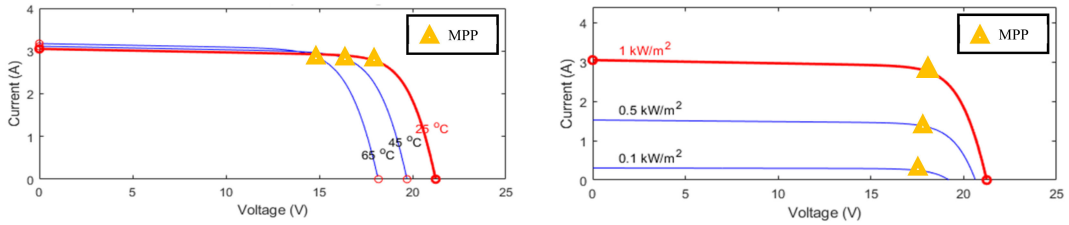

**Figure 2.** Equivalent circuit of a photovoltaic (PV) module.

Substituting (5) and (6) into (3) yields:

$$I_{sm} = NI_{ph} - NI_0\left(e^{\frac{q(\frac{V_{sm}}{M} + R_s\frac{I_{sm}}{N})}{\eta KT}} - 1\right) - \frac{\frac{N}{M}V_{sm} - I_{sm}R_s}{R_{sh}}, \tag{7}$$

which is the *I–V* expression of an $M \times N$ PV module.

As described in (7), the *I–V* curve of a PV module is strongly affected by environmental factors, including the variance of the solar irradiance and the module temperature. Figure 3a shows the *I–V* curves with different temperatures under the same irradiance condition. As the temperature rises, the curve moves left and vice versa. In addition, the plot of *I–V* curves under several distinct irradiance conditions with a constant temperature is presented in Figure 3b. It is obvious that the escalation of solar irradiance will cause a raise in the *I–V* curve, correspondingly. From Figure 3a,b, the maximum power point changes as the irradiance and module temperature vary, thus the variance of irradiance and module temperature will be considered in the proposed MPPT.




**(a)** **(b)**

**Figure 3.** *I–V* curves of a PV module with (**a**) different temperature and (**b**) different irradiance.

## 3. Maximum Power Point Tracking (MPPT) Control

As previously mentioned, the input impedance of the converter can be tuned by varying the duty ratio. If a PV module's output impedance is matched with the converter's input impedance, the maximum power can be extracted. As illustrated in Figure 4, the operating point of a PV module is the intersection of the *I–V* curve and the load line. The purpose of maximum power point tracking (MPPT) is to reach the operating point where the PV module has maximum power output. In this paper, the MPPT is implemented by adjusting the duty ratio of the converter. The P&O method will be discussed in Section 3.1, and the RL MPPT methods will be proposed in Section 3.2.

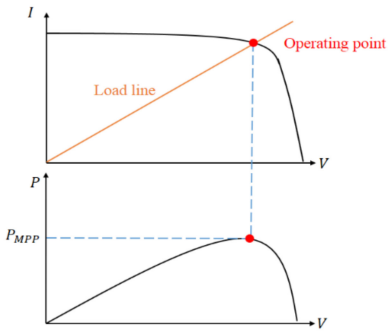



**Figure 4.** Load line and operating point.

### 3.1. Perturbation and Observation (P&O) Method

Figure 5 briefly illustrates the concept of the perturbation and observation method (P&O method), which is one of the well-known tracking methods. At time $t$, a fixed-sized perturbation is performed according to the measured power difference $\Delta P_{t-1}$ and voltage difference $\Delta V_{t-1}$. Then the change of the power and the voltage will be observed. With a new power difference $\Delta P_t$ and a new voltage difference $\Delta V_t$, the system can be perturbed accordingly.

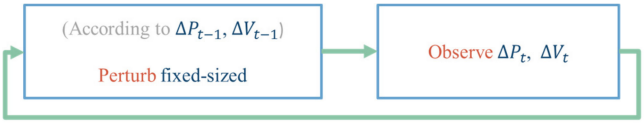
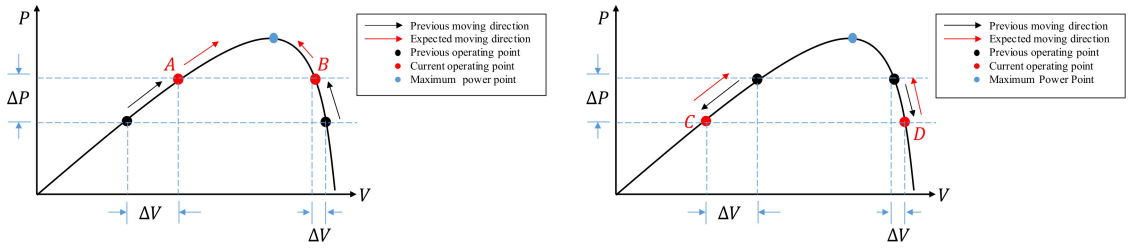



**Figure 5.** Concept of the perturbation and observation (P&O) method.

To drive the operating point toward the MPP, the system first measures the present power $P(n)$ and voltage $V(n)$ and then calculates the difference of power and the difference of voltage. If $\Delta P > 0$, there are two cases, operating points A and B, as depicted in Figure 6a. To move toward the MPP, the duty ratio should be decreased for point A with $\Delta V > 0$, and increased for point B with $\Delta V < 0$.




**(a)** **(b)**

**Figure 6.** (**a**) Positive power difference condition; (**b**) negative power difference condition.

On the other hand, if $\Delta P < 0$, there are two cases, operating points $C$ and $D$, as depicted in Figure 6b. To move toward the MPP, the duty ratio should be decreased for point $C$ with $\Delta V < 0$, and increased for point $D$ with $\Delta V > 0$.

### *3.2. Design of the Reinforcement Learning MPPT System*

Sequential decision-making is a common problem in real life, for example, an infant trying to walk by stretching a leg forward and move his body. By taking a series of actions, the infant will have a chance to reach his goal (keep moving forward). The Markov decision process (MDP) provides a systematic framework for describing this sequential decision-making problem. To solve an MDP, a reinforcement learning (RL) method is proposed by [8]. Learning by interacting with the environment is a human's intuitive learning skill. When facing an unknown system, the human will interact with it according to their own understanding of the system, and then receive feedback. This feedback signal provides a criterion for judging how "good" or "bad" an action is under a specific circumstance. The actions with better outcomes will have a larger chance to be chosen, i.e., they are reinforced, while the actions with worse outcomes will have smaller chance of being done in the future.

With the description of RL provided above, the concept of the variable step size tracking method based on RL is shown in Figure 7. The perturbation size can be chosen according to the observed system state and the previous experiences. Once the perturbation is done, the system change including the change of the state and the feedback signal will be observed and the interaction experience will be learned.

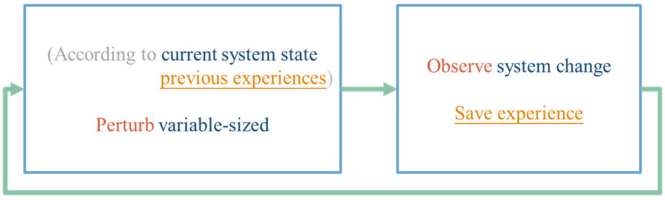



**Figure 7.** The concept of reinforcement learning-based maximum power point tracking (RL MPPT).

To provide a detailed description of the RL MPPT, the background knowledge of RL will be described from Section 3.2.1 to Section 3.2.5, including the description of the system, the interaction, and the evaluation of the interaction experience. Finally, the design of the RL MPPT methods, including the RL-QT MPPT method and the RL-QN MPPT method will be proposed in Section 3.3.

#### 3.2.1. Markov Decision Process and Reinforcement Learning Problem

A Markov decision process (MDP) [17,18] consists of $S$, $A$, $T$, and $R$, where $S$ represents the set of environment's state description, and $A$ is the set of available actions the agent can take. $T$ is the transition function, which indicates the system's probability distribution of jumping from any state to all possible states after applying all available actions, and it is denoted as $T : S \times A \times S \rightarrow [0, 1]$. For example, the probability of jumping from the state $s \in S$ to state $s\prime \in S$ after applying the action $a \in A$

can be written as $T(s, a, s\prime)$. For an MDP, the Markov properties hold. Consider a discrete-time series $t = 1, 2, \ldots$, the next state $s_{t+1}$ only depends on the current state $s_t$ and the current action $a_t$, as shown below:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots) = P(s_{t+1}|s_t, a_t) = T(s_t, a_t, s_{t+1}). \quad (8)$$

A reward $R$ is a scalar numerical signal received from doing an action at a state, and the reward function $R$ can be formally represented as $R : S \times A \times S \to \mathbb{R}$.

In RL, the one who actively takes actions that affect the environment is called an agent, and the environment is an object that reacts passively to the agent. The interaction between the agent and the environment can be described under the MDP framework. The agent–environment interface is shown in Figure 8. The agent and the environment interact discretely in a time series $t = 0, 1, 2, 3$. For each time step $t$, the description of the environment's current condition $s_t$ is obtained by the agent. The agent will decide which action $a_t$ should be taken based on some rules. Then a numerical reward signal $r_{t+1}$ and a new state $s_{t+1}$ are brought out by the environment.

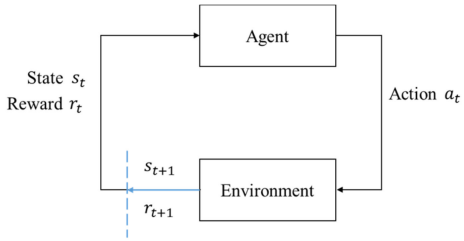



**Figure 8.** The agent–environment interface.

Figure 9 is a common representation of describing the interaction between the agent and the environment. The white circles represent the states and the black dots are the actions. The top white circle is the current state, and the middle black dots are all of the available actions. The possible succeeding states are the white circles in the middle, and the corresponding available actions are listed at the bottom of the diagram.

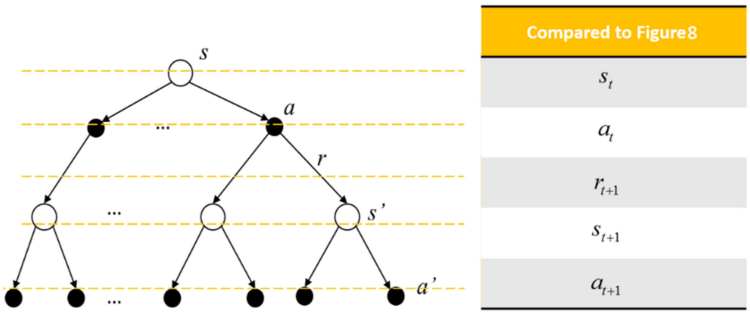



**Figure 9.** Sequential agent–environment interaction diagram.

The rule followed by an agent is called a policy. Formally, an agent's policy $\pi$ is the mapping of the current state to the selection probability of the available actions. Hence, the probability of selecting $a_t$ when the agent is at the state $s_t$ can be denoted as $\pi_t(a_t|s_t)$. For example, under a policy $\pi$, the probability of selecting $a_1$ at state $s$ can be written as $\pi(a_1|s)$, as shown in Figure 10. With the experience (such as the tuple $(s_t, a_t, r_{t+1}, s_{t+1})$) gathered, the policy can be modified. The goal of reinforcement learning is to obtain a policy, such that the received cumulative rewards are maximized.

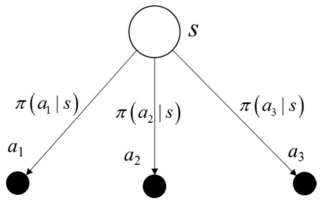



**Figure 10.** Example of the policy.

#### 3.2.2. Long Term Rewards

The expected return $G_t$ is the sum of the discounting rewards the agent expected to receive at time $t$, as below:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = r_{t+1} + \gamma G_{t+1} \tag{9}$$

where $\gamma$ is the discounting parameter, $0 \leq \gamma \leq 1$. The rewards are discounted in order to avoid infinite cumulated rewards. With a smaller $\gamma$, the agent focuses on the recent rewards more, while a larger $\gamma$ will make the agent be farsighted, and thus long term rewards will be considered. Also, the expected return can be written as an iterative form, which indicates that the current return is equal to the sum of the immediate reward $r_{t+1}$ and the successor state return $G_{t+1}$.

#### 3.2.3. Action Value and Optimal Action Value

The action value is the agent's expected return starting from the state $s$, with the action $a$ chosen and thereafter following the policy $\pi$, as described in (10):

$$q_\pi(s,a) = E_\pi[G_t|s_t = s, a_t = a] = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi\left(a'\middle|s'\right) q_\pi(s',a')\right] \tag{10}$$

where $r$ is the immediate reward received by doing action $a$, and $\gamma \sum_{a'} \pi(a'|s')q_\pi(s',a')$ is the discounted action value starting from all possible next states $s'$.

The backup diagram of the action value is shown in Figure 11. The top white circle and black dot are the state $s$ and the action $a$ being chosen under the state $s$, respectively. $r$ is the reward received consequently. The white circles in the middle are the successor states $s'$ the agent may obtain after doing action $a$. After that, the agent does actions according to the policy $\pi$, and the black dots in the bottom are the possible actions $a'$ being chosen.

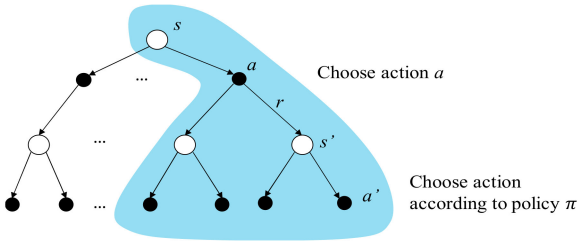



**Figure 11.** The backup diagram of action value function.

To an action value function, a policy $\pi$ is better than the other policy $\pi'$, or $\pi > \pi'$, if and only if $q_\pi(s,a) \geq q_{\pi'}(s,a), \forall s \in S, a \in A$. There always exists a policy that is better or equal to all the other policies and that is an optimal policy. The optimal action value function is defined as the expected return starting from the state $s$ with the action a chosen, thereafter following the optimal policy, such that the expected returns in the following states are maximized, as shown in (11)

$$q_*(s,a) = \max_\pi q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} q_*(s',a')\right] \tag{11}$$

where, $\max_{a'} q_*(s', a')$ is the optimal action value function of the next state $s'$ with optimal action $a'$ chosen. The backup diagram of the optimal action value function is depicted in Figure 12 with the blue background. The arc between different $a'$ is a representation of choosing the optimal $a'$ such that the return starting from $s'$ is maximized.

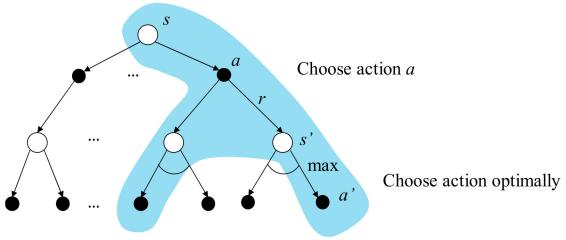



**Figure 12.** The backup diagram of the optimal action value function.

3.2.4. Introduction to Q-learning

Q-learning [19] is a model free temporal difference (TD) method to perform RL. A Q-table is constructed through bootstrapping to store the optimal action value of any state-action pair. The one-step update of Q-learning is shown as below:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a\_ \in A(s')} Q(s', a\_) - Q(s,a)\right] \tag{12}$$

where $Q(s,a)$ directly approximates the optimal action value $q*$ and is independent of the policy followed. For the state $s\prime$, an optimal action $a\_$ is expected to be selected within action set $A(s')$ so that the Q value at $s\prime$ can be maximized, i.e., $\max_{a\_ \in A(s')} Q(s', a\_)$. The updating rate of the $Q$ value is $\alpha$, $0 \leq \alpha < 1$, and $Q$ has been proven to converge to $q*$ with a probability 1 by [19]. The backup diagram of Q-learning is depicted in Figure 13 with a blue background. Through actual interactions, the experiences $(s, a, r, s\prime)$ can be acquired. According to the interaction experiences, the $Q$ values can be updated by (12) and stored in a tabular form, which is called a Q-table.

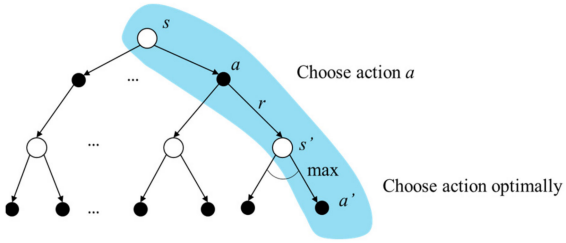



**Figure 13.** The backup diagram of Q-learning.

Once the Q-table is fully constructed, the optimal policy can be extracted by greedily choosing the action with the largest optimal action value for each state, i.e., $\operatorname{argmax}_{a \in A(s)} Q(s,a)$. Table 1 is an example of policy extraction from a well-constructed Q-table. The agent will take the action $A3$ when it is at the state $S1$ since $\operatorname{argmax}_{a \in \{A1, A2, A3\}} Q(S1, a) = A3$. Similarly, $A1$ should be taken at states $S2$ and $S4$, and $A2$ should be taken at state $S3$.

**Table 1.** Q-table example.

| $Q(s,a)$ | $A1$ | $A2$ | $A3$ |
|---|---|---|---|
| $S1$ | $Q(S1,A1) = 0.5$ | $Q(S1,A2) = 1$ | $Q(S1,A3) = 1.5$ |
| $S2$ | $Q(S2,A1) = 3$ | $Q(S2,A2) = -1$ | $Q(S2,A3) = 2$ |
| $S3$ | $Q(S3,A1) = 3$ | $Q(S3,A2) = 5$ | $Q(S3,A3) = 0.5$ |
| $S4$ | $Q(S4,A1) = -1$ | $Q(S3,A2) = -5$ | $Q(S4,A3) = -2$ |

One of the issues in RL is the exploration–exploitation trade-off [8]. For some RL problems, the policy changes with time, thus off-line learning is not suitable for them. Therefore, for each time step, the system is designed to be $\varepsilon$-greedy, i.e., choosing actions randomly to explore new possible rules in a probability $\varepsilon$ and otherwise following the learned policy and always choosing the action with the largest Q value.

The algorithm of Q-learning is shown in Algorithm 1. First, the elements in the Q-table are initialized. The $\varepsilon$-greedy, the learning rate $\alpha$ and the discount factor $\gamma$ are also initialized. Note that for a non-episodic MDP, there is no ending state, therefore, $\gamma$ should be less than 1 to avoid infinite expected return.

The agent observes the current state first. With a probability $\varepsilon$, the agent will randomly choose the available actions, otherwise, it will choose the action with the largest Q value according to the Q-table. After applying the action, the environment will generate the reward signal $r_{t+1}$, and then a new state $s_{t+1}$ can be observed. With $(s_t, a_t, r_{t+1}, s_{t+1})$ obtained, the element $Q(s_t, a_t)$ in the Q-table can be updated using (34). Finally, the current state $s_t$ is replaced by the new state $s_{t+1}$, and one step of an update is completed.

**Algorithm 1**: Non-episodic Q-learning algorithm

Initialize $Q(s,a), \forall s \in S, a \in A$
Initialize $\varepsilon, \alpha, \gamma, 0 \le \varepsilon \le 1, 0 \le \alpha < 1, 0 < \gamma < 1$
Observe $s_t$
**Repeat** (for each time step $t$)

$$\begin{cases} \text{randomly choose } a_t & \text{probability } \varepsilon \\ a_t = \underset{a\_ \in A(s_t)}{\text{argmax}} Q(s_t, a\_) & \text{otherwise} \end{cases}$$

Apply $a_t$, observe $r_{t+1}$ and $s_{t+1}$

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_{a\_ \in A(s_{t+1})} Q(s_{t+1}, a\_) - Q(s_t,a_t)\right]$$

$s_t \leftarrow s_{t+1}$

An example of the agent–environment model with the Q-table is illustrated in Figure 14. The state representation generated by the environment is required to be discretized to perform table lookup on the Q-table, which is impractical in some cases, such as the state space is too large or the state space is continuous. Therefore, an approximation of the Q-table using a neural network, named Q-network [13], has been implemented in this paper.

#### 3.2.5. Q-Learning with Neural Network Approximation

The Q-table in Figure 14 can be approximated as the Q-network in Figure 15. The state representation can be directly used as the input of the Q-network without discretization. The number of input nodes is the dimension of the state representation. The output of the Q-network $Q(s,a;\theta)$ is used to approximate the optimal action value $q*$, where $\theta$ is the weight of the network, i.e.,$Q(s,a;\theta) \approx q_*(s,a)$.

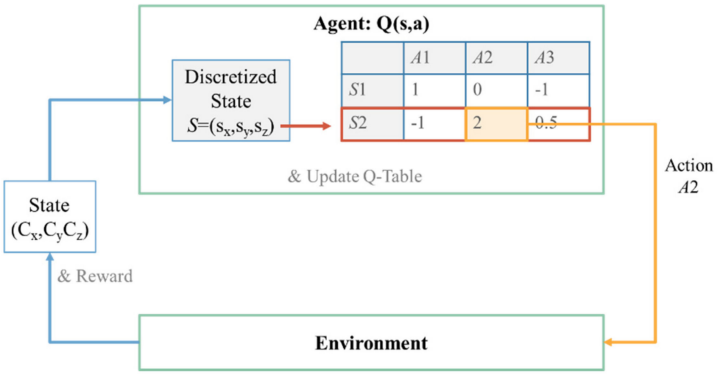



**Figure 14.** An example of agent–environment model with Q-table.

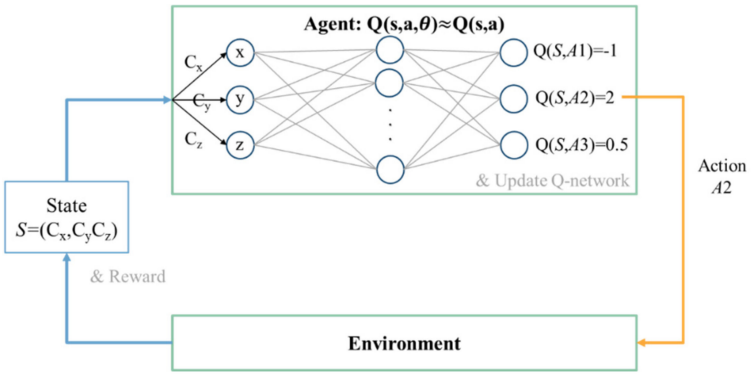



**Figure 15.** An example of the agent–environment model with the Q-network.

The loss function, also known as the cost function [20], of the Q-network in iteration $i$ is defined as:

$$L_i(\theta_i) = \left( \underbrace{y_i}_{\text{target Q value}} - \underbrace{Q(s,a;\theta_i)}_{\text{estimate Q value}} \right)^2 \tag{13}$$

where $y_i = r + \gamma \max_{a'} Q(s',a';\theta_{i-1})$ is the target Q value based on the current reward r and the maximum Q value at the next step $S'$ generated by the Q-network in iteration $i-1$, and $Q(s,a;\theta_i)$ is the estimated Q value provided by the Q-network in iteration $i$. For every $i$, (13) should be minimized in order to approximate the estimate Q value to the target Q value. The target Q-network parameter of the previous iteration, $\theta_{i-1}$, should hold fixed during the training in iteration $i$. This may cause the oscillation or divergence of the policy since the target Q value is affected immediately after updating the Q-network for every iteration.

To stabilize the learning process, a fixed target Q-network technique is proposed by [13], and the loss function is written as

$$L_i(\theta_i) = (y_i - Q(s,a;\theta_i))^2 = \left( \underbrace{\left( r + \gamma \max_{a'} \hat{Q}(s',a';\theta^-) \right)}_{\text{target Q value}} - \underbrace{Q(s,a;\theta_i)}_{\text{estimate Q value}} \right)^2 \tag{14}$$

where $\theta^-$ is the old parameter several iterations before, and it will update to the current value for every $C_T$ iterations, $C_T > 1$. Therefore, the frequent update of the Q-network $Q$ will have less effect on the target Q value $\hat{Q}$, and the training process will be more stable.

To break down the correlation between training samples, the experience replay technique is proposed by [12]. For each time step t, an experience sample is gathered and stored by the agent into a data set $E = \{e_1, \ldots e_t\}$, where $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$. For each learning iteration, several samples are taken randomly as a mini-batch $\left(s_j, a_j, r_{j+1}, s_{j+1}\right)$ to perform mini-batch gradient descent [21], and the loss function can be rewritten as

$$L_j(\theta) = \left(y_j - Q\left(s_j, a_j; \theta\right)\right)^2 = \left(\underbrace{\left(r_{j+1} + \gamma \max_{a\_} \hat{Q}\left(s_{j+1}, a\_; \theta^-\right)\right)}_{\text{target Q value}} - \underbrace{Q\left(s_j, a_j; \theta\right)}_{\text{estimate Q value}}\right)^2 \tag{15}$$

Finally, a Q learning algorithm using a Q-network as the function approximation is shown in Algorithm 2. The parameter of $Q$ and $\hat{Q}$, $\theta$ and $\theta^-$, are initialized. The target $Q$ parameter update period $C_T$, the experience data set $E$, the $\varepsilon$-greedy policy, the learning rate $\alpha$, the discount factor $\gamma$ and the experience replay threshold $p$th are also initialized. The agent then observed the current state $s_t$. With a probability $\varepsilon$, the agent will do the action randomly, otherwise, it will choose the action with the largest $Q$ value according to the output of the Q-network, i.e., $a_t = \underset{a\_}{\text{argmax}} Q(s_t, a\_; \theta)$. After applying an action, the agent observed the reward and the representation of the next state, and $(s_t, a_t, r_{t+1}, s_{t+1})$ will be stored into $E$. After cumulating enough experiences, an experience replay can be performed by sampling a mini-batch of experiences $\left(s_j, a_j, r_{j+1}, s_{j+1}\right)$ from E randomly. The mini-batch target Q values $y_j$ are calculated by the $\hat{Q}$ instead of Q because of the performing of the fixed target Q-network technique, and then a mini-batch gradient descent [21] is applied to minimize the loss function shown in (37). The target Q-network parameter $\theta^-$ will be updated for every $C_T$ iterations. Finally, the state is updated, and a new iteration will begin.

**Algorithm 2**: Q-learning using Q-network approximation with fixed target Q-networks and experience replay

Initialize $C_T$, $E$,$\theta$, $\theta^-$, $\varepsilon$, $\alpha$, $\gamma$, $pth$, $0 \le \varepsilon \le 1, 0 \le \alpha < 1, 0 < \gamma < 1$
Observe $s_t$
**Repeat** (for each time step $t$)

$$\begin{cases} \text{randomly choose } a_t & \text{probability } \varepsilon \\ a_t = \underset{a\_ \in A(s_t)}{\text{argmax}} Q(s_t, a\_; \theta) & \text{otherwise} \end{cases}$$

Apply $a_t$, observe $r_{t+1}$ and $s_{t+1}$
Store $(s_t, a_t, r_{t+1}, s_{t+1})$ in $E$
If $t > pth$
Randomly Sample mini-batch $\left(s_j, a_j, r_{j+1}, s_{j+1}\right)$ from $E$
Calculate loss function as (15)
Perform mini-batch gradient descent to optimize the loss function
Every $C_T$ step update $\hat{Q} \leftarrow Q$
$s_{t+1} \leftarrow s_t$

### *3.3. Design of an RL MPPT System*

To perform MPPT based on RL, the system must be able to be described by MDP. The element needed in the RL MPPT system is defined in Table 2. The PV module and the converter can be seen as the environment, and the controller is the agent as depicted in Figure 16. The goal of the agent is to reach the MPP through interacting with the environment.

**Table 2.** RL MPPT system element selection.

| Parameter Needed to Perform RL | Parameter Selection in Solar MPPT System |
|---|---|
| Environment | PV module and converter |
| Agent | controller |
| State | (irradiance, temperature, duty ratio) |
| Action | $\Delta D$ |
| Reward | $\Delta P = P' - P$ |

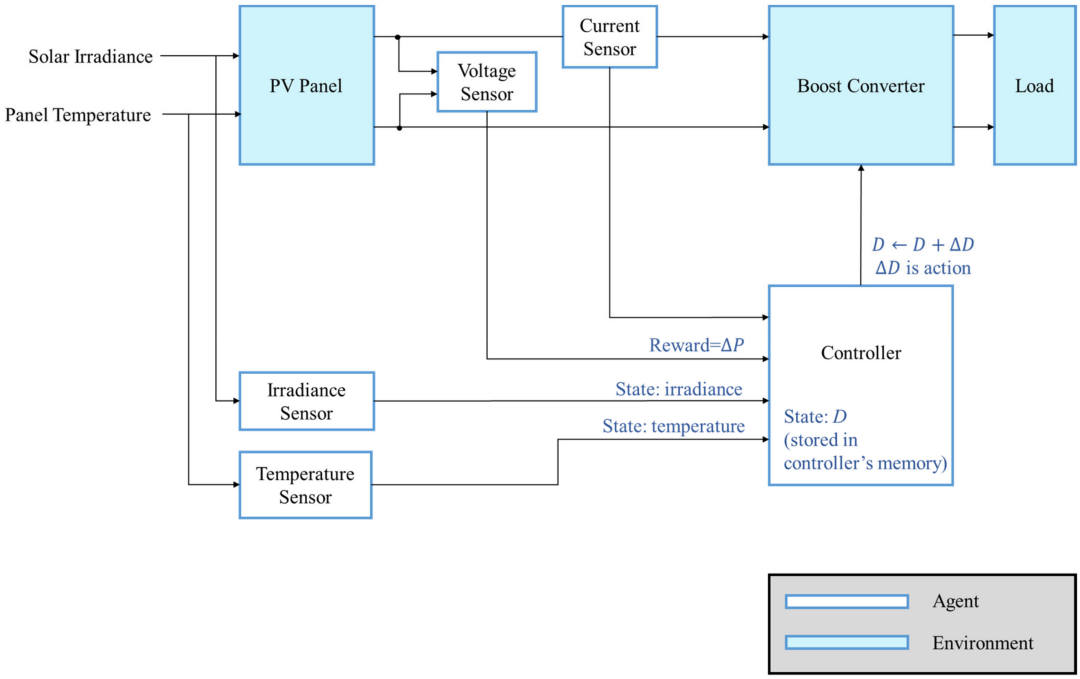



**Figure 16.** Agent and environment defined in the RL MPPT system.

The system's condition, the state, is described by the solar irradiance, the module temperature and the duty ratio D since the *I-V* curve is affected by the solar irradiance and the module temperature as mentioned previously. The system's operating point is at the intersection of the *I*–*V* curve and the load line, and the load line is controlled by the duty ratio of the converter.

In this study, the action is defined as a set of duty ratio changing step $\Delta D$ with different step sizes. Therefore, the tracking progress can be seen as a sequential decision-making problem, i.e., the MPP of the system can be reached by applying a series of variable step sizes $\Delta D$ appropriately.

The reward is a numerical signal that helps the agent judge how "good" or "bad" an action is. The action that moves the operating point close to the MPP is better than the action that moves the operating point away from the MPP. Therefore, the power difference $\Delta P = P' - P$ is defined as the reward since it provides not only the moving direction of the operating point but also the numerical scaling representation of the effect caused by applying the action, for example, a larger step size may lead to a larger power difference.

Also, the Markovian property holds since the current state is only affected by the state and the action taken one step before. Through combining the elements of the MDP model designed in Table 2 and the concept of the RL MPPT shown in Figure 7, the detail of the RL MPPT can be described as Figure 17.

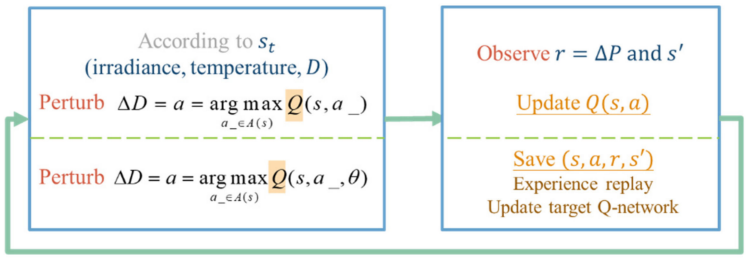



**Figure 17.** Simple workflow of the RL MPPT.

The perturbation step size is chosen according to the Q value of the current irradiance, temperature, and duty ratio $D$. After applying the change of $D$, the power difference $\Delta P$ and the new state description $s'$ can be observed. In the Q-table approach, the experience will be used to update the Q value immediately, but in the Q-network approach, it will be stored to perform experience replay. With the simple workflow of the RL MPPT described as above, the flowcharts of the RL MPPT using Q-table (RL-QT MPPT) and the RL MPPT using Q-network (RL-QN MPPT) are shown in Figures 18 and 19. The flowcharts essentially follow the Q-learning algorithms provided in Algorithm 1 and Algorithm 2, respectively.

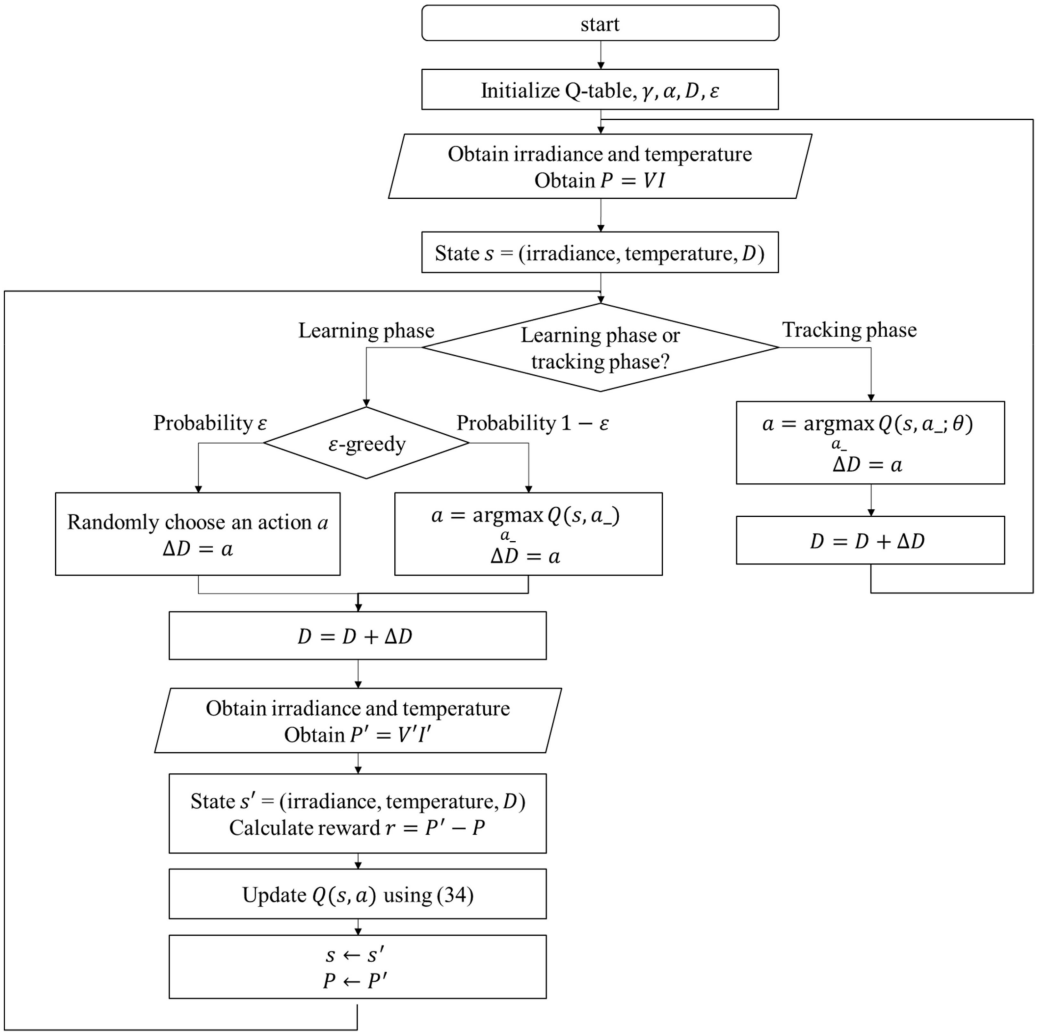



**Figure 18.** Flowchart of the RL MPPT using the Q-table (RL-QT MPPT).

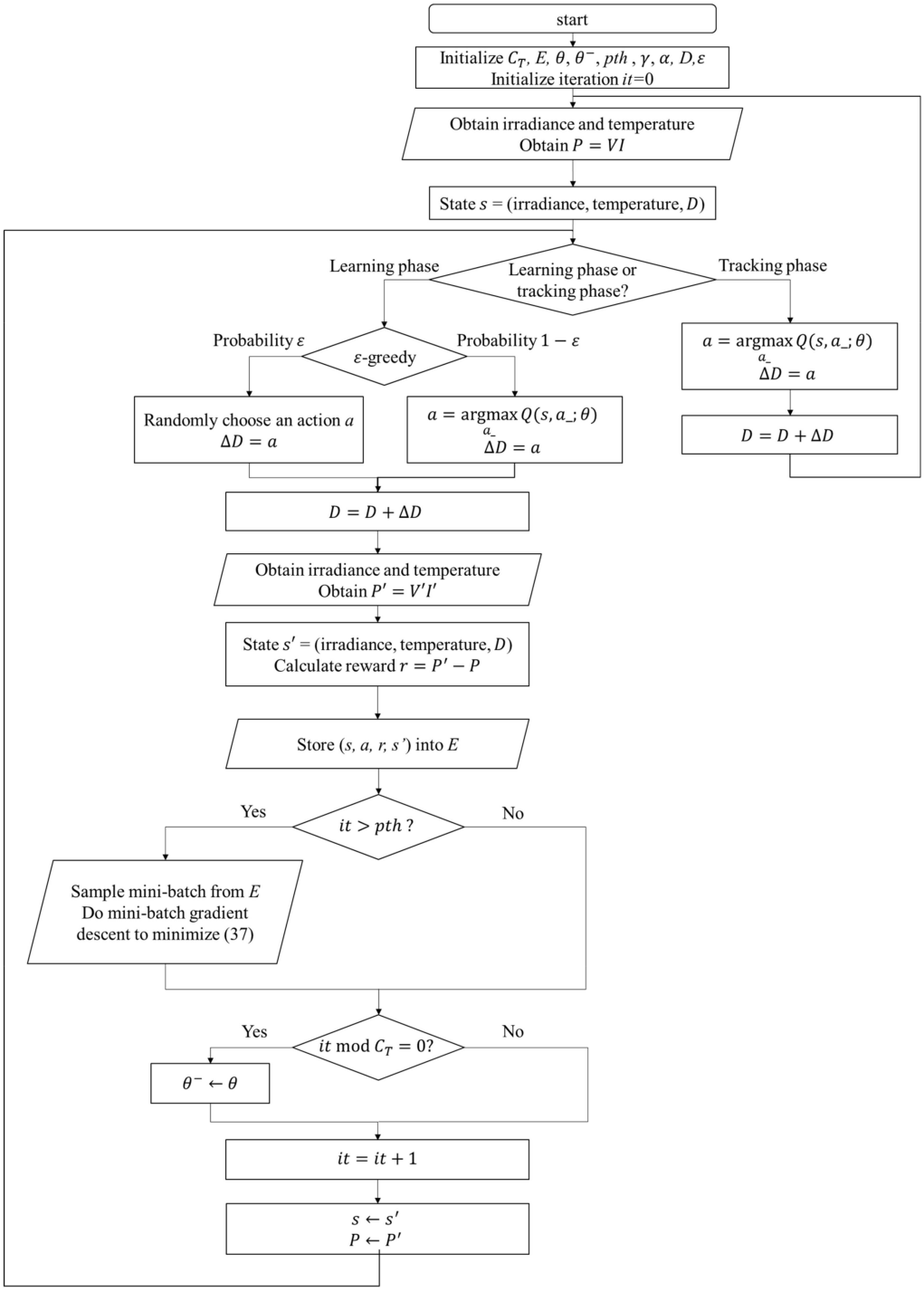



**Figure 19.** Flowchart of the RL MPPT using the Q-network (RL-QN MPPT).

The proposed RL MPPT methods in this paper include two phases, the learning phase and tracking phase. In the learning phase, the experiences are learned, i.e., the Q-table or the Q-network is updated. However, to speed up the tracking speed, the update process is skipped in the tracking phase. The description of the learning phase and the tracking phase of the RL-QT MPPT and RL-QN MPPT are shown below:

(a) Learning Phase of RL-QT MPPT

First, the Q-table, the discount factor $\gamma$, the learning rate $\alpha$, the $\varepsilon$-greedy policy, and the duty ratio $D$ are initialized, and the solar irradiance and the temperature of the PV module are sensed to form

the state representation (irradiance, temperature, $D$). The output power of the PV module is sensed and stored as $P$. With a probability of $\varepsilon$, the agent will randomly choose an action and change the perturbation step size. Otherwise it will follow the Q-table and choose the action with the largest Q value. After applying the new duty ratio, the irradiance, the temperature and the output power $P\prime$ are sensed again. Therefore, the succeeding state $s\prime$ is obtained, and the reward $r$ can be calculated as $P' - P$. Finally, with $(s, a, r, s')$, the corresponding element in the Q-table, i.e., $Q(s, a)$, can be updated by (34).

(b) Tracking Phase of RL-QT MPPT

In the tracking phase, the agent selects the action by looking up the Q-table, and $D$ is changed accordingly. Then the iteration ends without updating the Q-table.

(c) Learning Phase of RL-QN MPPT

The RL-QN MPPT is similar to the RL-QT MPPT. First, the fixed target Q-network update iteration $C_T$, the experience dataset $E$, the network parameters $\theta$ and $\theta^-$, the experience replay threshold pth, the discount factor $\gamma$, the learning rate $\alpha$, the $\varepsilon$-greedy policy, and the duty ratio $D$ are initialized, and the iteration counter is also initialized. Then the state (irradiance, temperature, $D$) is obtained by acquiring the solar irradiance data and the module temperature data. The output power of the PV module is calculated as $P = VI$. Under the $\varepsilon$-greedy policy, the action will be chosen randomly in a probability $\varepsilon$, otherwise the action with the largest Q value will be taken. The subsequent state representation $s'$ and the reward $r$ can be obtained after changing the duty ratio of the converter. The experience $(s, a, r, s')$ is stored in $E$, and if the experiences in $E$ are enough, the experience replay techniques will be performed. For every $C_T$ step, the fixed target Q-network weight $\theta^-$ will be updated, and finally, it will be increased to move onto the next iteration.

(d) Tracking Phase of RL-QN MPPT

In the tracking phase, the agent follows the policy approximated by the Q-network, i.e., always choose the action with the largest Q value. Then $D$ is modified by $\Delta D$. The system's operating point is changed, but the experience is not stored to speed up tracking. Then a new iteration will begin.

## 4. Results

### *4.1. System Configuration*

The environment configuration of both simulation and experiment is shown in Table 3, including the description of the PV module and the parameter of the DC–DC boost converter. The agent configuration is shown in Table 4, which is identical for both simulation and experiment. The corresponding hardware structure is depicted in Figure 20. For the environment described in Table 3, the configuration of the PV module, the DC–DC boost converter and the resistive load are identical for both simulation and experiment, while two large resistor $R_1$ and $R_2$ are added into the actual circuit as a voltage divider to measure the voltage, as shown in Figure 20.

**Table 3.** Environment configuration of simulation and experiment.

| **PV Module** | **Power = 50 *W*, Voc = 21.24*V*, Isc = 3.05*A*, (Test Condition: 25 °C, 1000 *W*/*m*$^2$)** |
|---|---|
| ***L*** | 1.1 $mH$ |
| ***C*** | 330 $\mu F$ |
| $R_{Load}$ | 100 Ω |
| $C_{in}$ | 1000 $\mu F$ |

**Table 4.** Agent–configuration of simulation and experiment.

| | RL-QT MPPT | RL-QN MPPT |
|---|---|---|
| ***D* range** | 0.2~0.9 | |
| **Sampling time** | 1 s | |
| **Δ*D* (action)** | {0, ±0.01, ±0.05, ±0.1} | |
| **State** | (irradiance, temperature, *D*) | |
| **Reward** | Δ*P* | |
| $\varepsilon$ | 1 | |
| $\gamma$ | 0.3 | |
| **Q value storing type** | Q-table<br>4260*7 | Q-network<br>3-40-40-40-7 |
| $\alpha$ | 0.9 | 0.0001 |
| $C_T$ | N/A | 100 |
| ***E*** | | No limit |
| ***pth*** | | 16 |

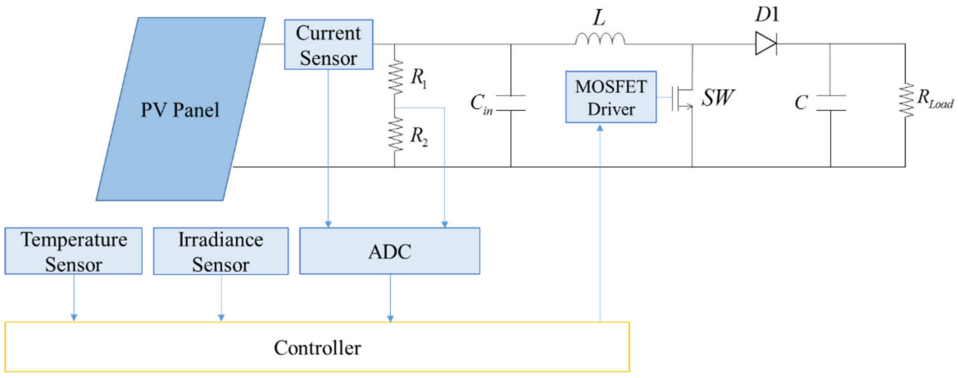



**Figure 20.** Hardware structure of the proposed system.

As for the configuration of the agent, the range of the duty ratio is set between 0.2 and 0.9 due to the limit of the hardware. The P&O method and the RL MPPT methods are performed one time per second for both simulation and experiment. $\Delta D$ of the P&O method is fixed at 0.05, while the RL MPPT methods provide a set of actions with different step sizes, i.e., $A = \{\Delta D = 0, \Delta D = \pm 0.01, \Delta D = \pm 0.05, \Delta D = \pm 0.1\}$. The state, reward, $\varepsilon$-greedy, and the discount factor $\gamma$ are the same in the RL-QT MPPT and RL-QN MPPT. However, the discretization of the state is needed to perform in the RL-QT MPPT method. For the RL-QT MPPT method, the irradiance is discretized into 10 levels between 0 and 1000 $W/m^2$, and the temperature is divided into 6 levels, which are below 20 °C, 20~30 °C, 30~40 °C, 40~50 °C, 50~60 °C, and beyond 60 °C, respectively. The level of $D$ is discretized by 0.01, the minimum value of non-zero $\Delta D$, and is limited between 0.2~0.9 as mentioned before. Therefore, the size of the Q-table can be obtained by calculating the size of the state space and the action space, which is 4260*7.

A multi-layer perceptron is used as the Q-network in this study. After several attempts and adjustments, a 5-layer structure with 3 hidden layers is applied in both simulation and experiment. The input layer consists of 3 neurons, which are the irradiance, the temperature, and duty ratio input, respectively. Each hidden layer is constructed by 40 neurons, and the output layer has 7 neurons, which represent the Q value of 7 actions,$\Delta D = 0, \Delta D = \pm 0.01, \Delta D = \pm 0.05, \Delta D = \pm 0.1$, respectively. The learning rate $\alpha$ is 0.9 in RL-QT MPPT method and 0.0001 in RL-QN MPPT method. $C_T$, $E$ and $pth$ are also initialized for the Q-network method to perform the experience replay and the fixed target

Q-network techniques. The total amount of experiences gathered by RL-QN MPPT in the learning phase are 18,646 and 26,838 for the simulation and the experiment, respectively.

### *4.2. Simulation Result*

The performance of the P&O, RL-QT MPPT, and RL-QN MPPT methods are simulated by MATLAB and Simulink R2017b with AMD Ryzen Threadripper 1920X processor, 3.50 GHz, 64 GB of DRAM memory and Microsoft Windows 10 operating system. With the irradiance and the temperature signal and the simulation configuration provided in Figure 21, Tables 3 and 4, the *P-t* graph and the *D-t* graph of the P&O method, RL-QT MPPT and RL-QN MPPT methods are shown in Figures 22–24, respectively. For the RL MPPT methods, larger step sizes are chosen when the operating point is far away from the MPP, and the smaller step sizes are selected when the operating point is near the MPP. Consequently, the RL MPPT methods outperform the P&O method since both of the RL-QT MPPT and the RL-QN MPPT provide smaller ripples and faster tracking speeds.

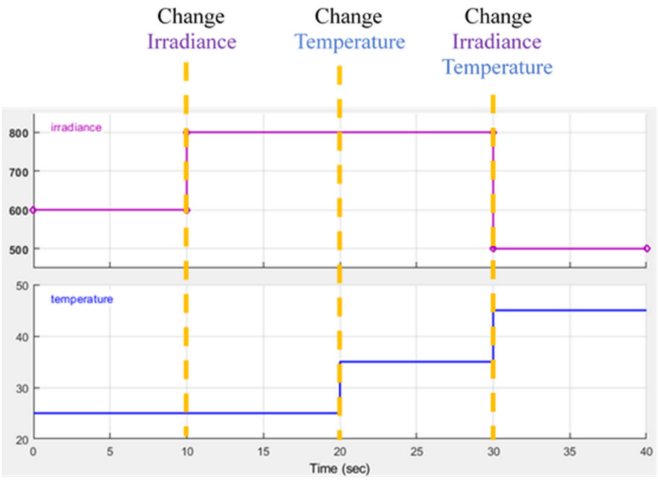



**Figure 21.** Irradiance and temperature simulation conditions.

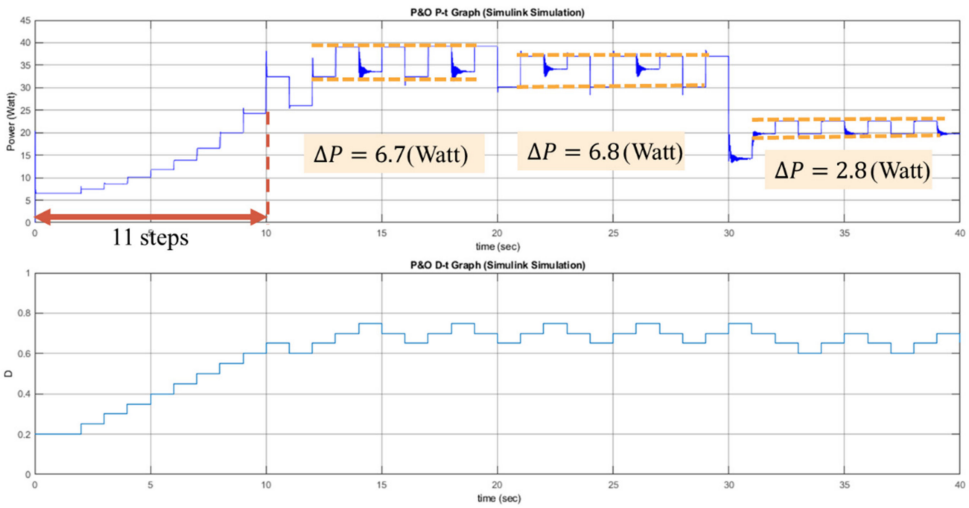



**Figure 22.** *P-t* graph and *D-t* graph of the P&O simulation result.

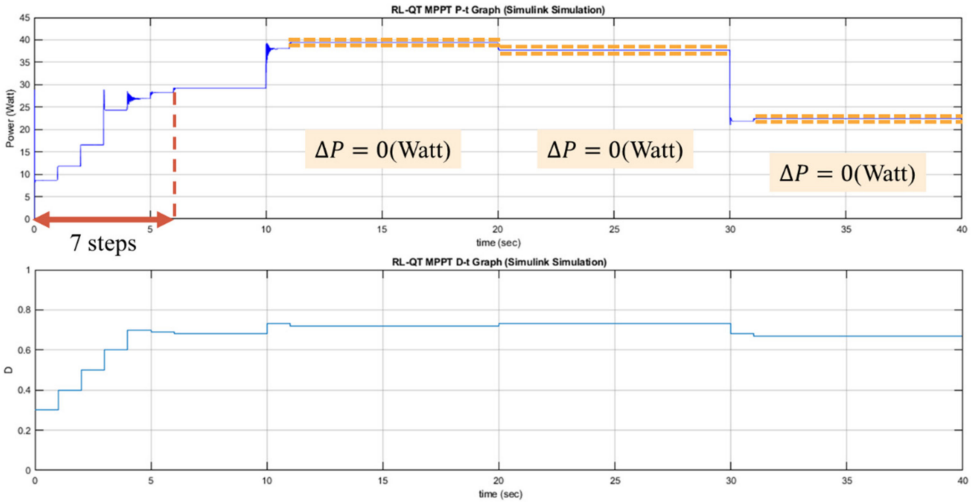



**Figure 23.** *P-t* graph and *D-t* graph of the RL-QT MPPT simulation result.

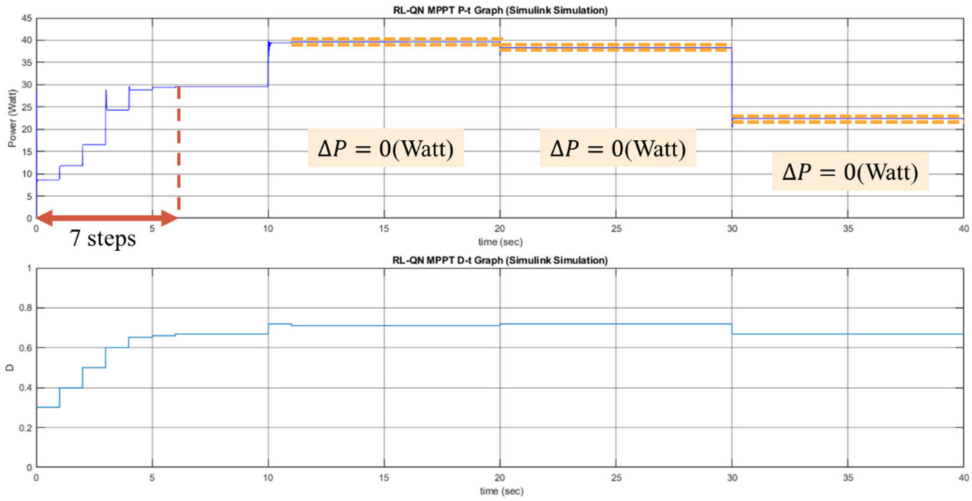



**Figure 24.** *P-t* graph and *D-t* graph of the RL-QN MPPT simulation result.

### *4.3. Experimental Result*

The hardware specification for the actual experiment setup are listed in Table 5. An analog to digital converter (ADC) is needed since there is no built-in ADC in Raspberry Pi 3 Model B [22]. The voltage is measured by the ADC, and the analog output of the current sensor is transferred into the digital signal by the ADC as well. An ambient light sensor module MAX44009 GY-49 is used to measure the illuminance, which can be converted to the solar irradiance with a ratio of 0.0079 $W/m^2$ per lux [23]. To measure the temperature, an infrared thermometer non-contact module MLX90614 GY-906 is used to measure the surface temperature of the PV module. The power MOSFET IRF840 is driven by the photocoupler TLP250. With a high current capability, 1N5408 is an appropriate choice for the diode D1 in the boost converter. Finally, the actual experiment setup is shown in Figure 25.

**Table 5.** Hardware specification of the experiment.

| Hardware Specification | |
|---|---|
| **Development board** | Raspberry Pi 3 Model B |
| **ADC** | ADS1115 |
| **Voltage sensor** | ADS1115 |
| **Current sensor** | ACS723 |
| **Irradiance sensor** | MAX44009 GY-49 |
| **Temperature sensor** | MLX90614 GY-906 |
| **MOSFET driver** | TLP250 |
| **Diode D1** | 1N5408 |

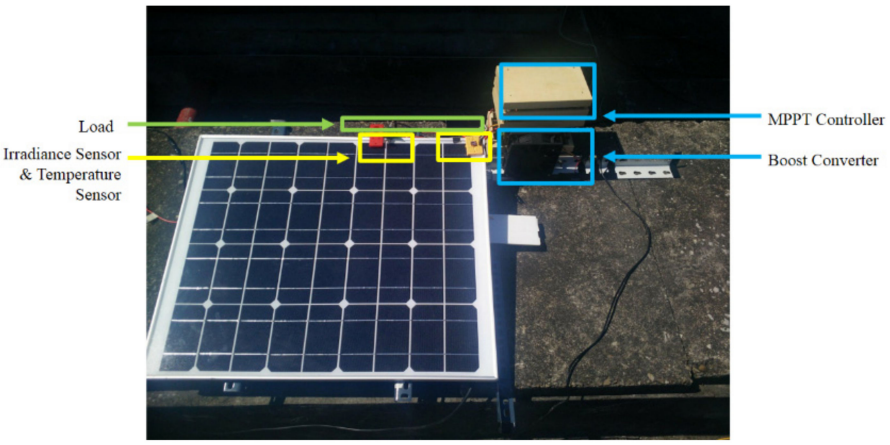



**Figure 25.** Overview of the hardware setup.

The experiments of the P&O method, the RL-QT MPPT method, and the RL-QN MPPT method were conducted under similar environmental conditions. The irradiance was about 650 $W/m^2$, and the surface temperature of the PV module was about 48 °C. The result is shown in Figures 26–28. Similar to the simulation result, the step sizes are selected appropriately by the RL MPPT methods during the tracking process, which leads to a better performance comparing to the P&O method.

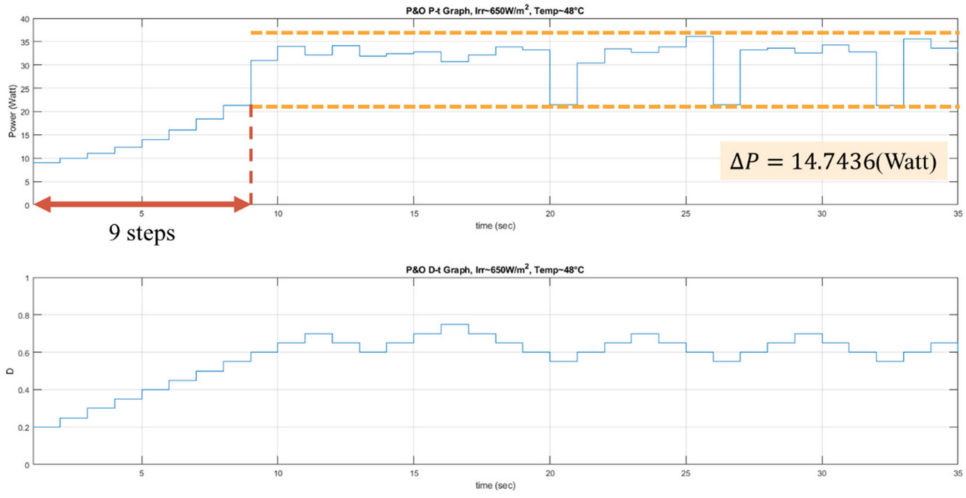



**Figure 26.** *P-t* graph and *D-t* graph of the P&O experiment result.

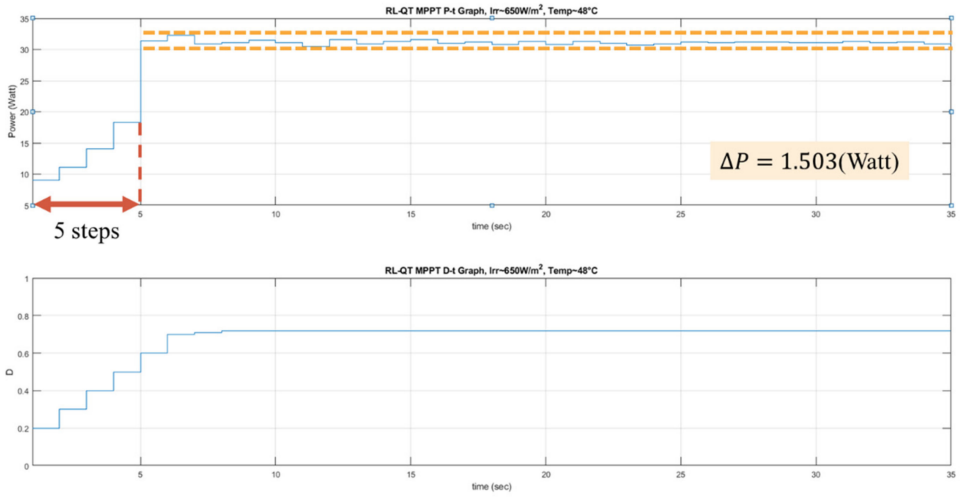



**Figure 27.** *P-t* graph and *D-t* graph of the RL-QT MPPT experiment result.

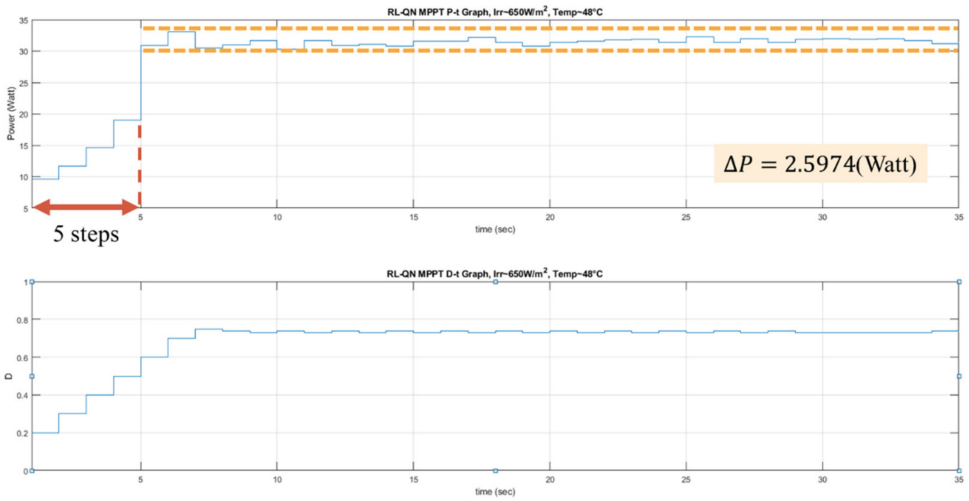



**Figure 28.** *P-t* graph and *D-t* graph of the RL-QN MPPT experiment result.

Table 6 shows the comparison of the experiment results of the three solar MPPT tracking methods. Both the RL-QT MPPT and the RL-QN MPPT are capable of tracking the MPP with fewer tracking steps than the P&O method and remain stable near the MPP. The average power around MPP of the two RL MPPT methods are near that of the P&O method, which indicate that the proposed methods are able to track the MPP.

**Table 6.** Comparison of the experimental results.

| | P&O | RL-QT MPPT | RL-QN MPPT |
|---|---|---|---|
| **Tracking steps (Sec)** | 9 | 5 | 5 |
| **Oscillation range (Watt)** | 14.7436 | 1.503 | 2.5974 |
| **Average power around MPP (Watt)** | 31.7478 | 31.144 | 31.4977 |

## 5. Conclusions

In this paper, two MPPT methods based on model-free reinforcement learning are proposed. The tracking process can be seen as a sequential decision-making problem since the MPP can be achieved through selecting an appropriate perturbation step size for every time step. Therefore, an MDP model is suitable for describing the interaction between the circuit connected to the PV module and the controller which is able to choose $\Delta D$ and change the duty ratio $D$ of the circuit. An MDP

model consists of four elements, which are state, action, transition, and reward. With the MDP model described, an RL-QT MPPT method is proposed by constructing the Q-table to perform MPPT control. However, the state representation is needed to be discretized for the tabular method, which may cause the loss of MPPT control accuracy. Therefore, a Q-network-based MPPT method is proposed. In the RL-QN MPPT method, the Q-table is approximated by a neural network, so that the discretization of the states are not needed. For both RL-QT MPPT and RL-QN MPPT, the tracking method consists of the learning phase and the tracking phase, which is able to expedite the tracking process since the Q value will not be updated in the tracking phase. A conclusion can be drawn from the simulation and experimental results that the RL MPPT methods are more effective than the traditional P&O method to track the MPP since the proposed RL MPPT methods possess smaller ripples and faster tracking speeds. Furthermore, for the proposed RL MPPT methods, the RL-QT MPPT method performs with smaller oscillations and the RL-QN MPPT method achieves a higher average power.

**Author Contributions:** Conceptualization and methodology, K.-Y.C., S.-T.Y., and Y.-P.C.; software and hardware, K.-Y.C. and S.-T.Y.; validation, K.-Y.C. and Y.-P.C.; writing—original draft preparation, K.-Y.C. and S.-T.Y.; writing—review and editing, K.-Y.C. and Y.-P.C.; funding acquisition and supervision Y.-P.C.

**Funding:** The authors would like to thank the Ministry of Science and Technology, Taiwan under Grant MOST 105-2511-S-009-016-MY3 and MOST 108-2221-E-009 -119 - for providing research funding.

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

## References

1. Bendib, B.; Belmili, H.; Krim, F. A survey of the most used MPPT methods: Conventional and advanced algorithms applied for photovoltaic systems. *Renew. Sustain. Energy Rev.* **2015**, *45*, 637–648. [CrossRef]
2. Salas, V.; Olias, E.; Barrado, A.; Lazaro, A. Review of the maximum power point tracking algorithms for stand-alone photovoltaic systems. *Sol. Energy Mat. Sol. Cells* **2006**, *90*, 1555–1578. [CrossRef]
3. Xiao, W.; Dunford, W.G. A modified adaptive hill climbing MPPT method for photovoltaic power systems. In Proceedings of the 2004 IEEE 35th Annual Power Electronics Specialists Conference (IEEE Cat. No. 04CH37551), Aachen, Germany, 20–25 June 2004; Volume 3, pp. 1957–1963.
4. Jung, Y.; So, J.; Yu, G.; Choi, J. Improved perturbation and observation method (IP&O) of MPPT control for photovoltaic power systems. In Proceedings of the Conference Record of the Thirty-first IEEE Photovoltaic Specialists Conference, Lake Buena Vista, FL, USA, 3–7 January 2005; pp. 1788–1791.
5. Ahmed, J.; Salam, Z. An improved perturb and observe (P&O) maximum power point tracking (MPPT) algorithm for higher efficiency. *Appl. Energy* **2015**, *150*, 97–108.
6. Rezoug, M.; Chenni, R.; Taibi, D. Fuzzy logic-based perturb and observe algorithm with variable step of a reference voltage for solar permanent magnet synchronous motor drive system fed by direct-connected photovoltaic array. *Energies* **2018**, *11*, 462. [CrossRef]
7. Mohd Zainuri, M.A.A.; Mohd Radzi, M.A.; Soh, A.C.; Rahim, N.A. Adaptive P&O-fuzzy control MPPT for PV boost dc-dc converter. In Proceedings of the 2012 IEEE International Conference on Power and Energy (PECon), Kota Kinabalu, Malaysia, 30 October–2 November 2012; pp. 524–529.
8. Sutton, R.; Barto, A. *Introduction to Reinforcement Learning*; MIT Press: Cambridge, MA, USA, 1998.
9. Youssef, A.; El-Telbany, M.; Zekry, A. Reinforcement Learning for Online Maximum Power Point Tracking Control. *J. Clean Energy Technol.* **2016**, *4*, 245–248. [CrossRef]
10. Hsu, R.C.; Liu, C.-T.; Chen, W.-Y.; Hsieh, H.-I.; Wang, H.-L. A Reinforcement Learning-Based Maximum Power Point Tracking Method for Photovoltaic Array. *Int. J. Photoenergy* **2015**, *2015*, 1–12. [CrossRef]
11. Kofinas, P.; Doltsinis, S.; Dounis, A.; Vouros, G. A reinforcement learning approach for MPPT control method of photovoltaic sources. *Renew. Energy* **2017**, *108*, 461–473. [CrossRef]
12. Lin, L.-J. Reinforcement Learning for Robots using Neural Networks. Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 1992.
13. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.; Veness, J.; Bellemare, M.; Graves, A.; Riedmiller, M.; Fidjeland, A.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
14. Tiwari, G.N.; Tiwari, A. *Handbook of Solar Energy—Theory, Analysis and Applications*; Sprinter: Singapore, 2017.

15. Humada, A.M.; Hojabri, M.; Mekhilef, S.; Hamada, H.M. Solar cell parameters extraction based on single and double-diode models: A review. *Renew. Sustain. Energy Rev.* **2016**, *56*, 494–509. [CrossRef]
16. Sedra, A.; Smith, K. *Microelectronic Circuits*, 6th ed.; Oxford University Press: New York, NY, USA, 2011.
17. Bellman, R. A Markovian Decision Process. *J. Math. Mech.* **1957**, *6*, 679–684. [CrossRef]
18. Howard, R.A. *Dynamic Programming and Markov Processes*; MIT Press: Cambridge, MA, USA, 1960.
19. Watkins, C.J.C.H.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 257–277. [CrossRef]
20. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
21. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
22. Raspberry Pi Foundation. Available online: http://www.raspberrypi.org (accessed on 16 November 2019).
23. Hossain, M.A.; Islam, M.S.; Chowdhury, M.M.H.; Sabuj, M.N.H.; Bari, M.S. Performance Evaluation of 1.68 kwp DC Operated Solar Pump with Auto Tracker Using Microcontroller Based Data Acquisition System. In Proceedings of the International Conference on Mechanical Engineering 2011, Dhaka, Bangladesh, 18–20 December 2011.