 doi:10.20944/preprints202208.0104.v1

Review

Reinforcement Learning: Theory and Applications in HEMS

Omar Al-Ani[1] and Sanjoy Das[2,*]

[1] Kansas State University; oalani@ksu.edu
[2] Kansas State University; sdas@ksu.edu
* Correspondence: sdas@ksu.edu

Abstract: The twin capabilities of learning from experience and learning at higher levels of abstraction, set reinforcement learning apart from other areas of machine learning and (within the broader context) all of artificial intelligence. It allows algorithmic agents to replace human beings in the real world, including in homes and buildings, in application domains that had hitherto been considered to be beyond today's capabilities. This goal, specifically aimed at home energy automation that forms the backdrop of this article, which surveys the use of deep reinforcement learning in various HEMS applications. The article provides an overview of generic reinforcement learning. This is followed with discussions on the state-of-the-art methods for value based, policy gradient, and actor-critic methods in deep reinforcement learning. In order to make published literature in reinforcement learning more accessible to HEMS researchers, verbal descriptions are accompanied with explanatory figures as well as mathematical expressions using the same terminology as the machine learning community. Next, a detailed survey of how reinforcement learning is used in different HEMS domains is described. The survey also considers what kind of reinforcement learning algorithms are used in each HEMS application. The survey suggests that this research is still in its infancy.

Keywords: HEMS; Reinforcement Learning; Deep Neural Network; Q-Value; Policy Gradient; Natural Gradient; Actor-Critic; Residential, Commercial, Academic.

1. Introduction

The largest group of consumers of electricity in the US are residential units. In the year 2020, this sector alone accounted for approximately 40% of all electricity usage [1]. The average daily residential consumption of electricity is 12 kWh per person [2]. Therefore, effectively managing the usage of electricity in homes, while maintaining acceptable comfort levels, is vital to address global challenges of dwindling natural resources and climate change. Rapid technological advances have now made *home energy management systems* (HEMS) an attainable goal that is worth pursuing. HEMS consist of automation technologies that can respond to continuously or periodically changing home environmental as well as relevant external conditions, without human intervention [3], [4]. In this review, the term 'home' is taken in a broad context to also include all residential units, classrooms, apartments, offices complexes, and other buildings in the smart grid [5], [6], [7], [8].

Artificial Intelligence (AI), more specifically machine learning, is one of the key contributing factors that have helped realize HEMS today [9], [10], [11]. *Reinforcement learning* (RL) is the class of machine learning algorithms that is making deep inroads in various applications in HEMS. RL allows an algorithmic entity from experience to make sequences of decisions and implement actions in the same manner as a human being [12], [13], [14], [15], [16], [17]. DNNs has proven to be a powerful tool in RL, for it endows the RL agent with the capability to adapt to a wide variety of highly complex domains [18], [19]. In [20] it has been proposed that RL can attain the ultimate goal of artificial general intelligence [21].

 doi:10.20944/preprints202208.0104.v1

Consequently, RL is making deep inroads into many application domains today. It has been applied extensively to robotics [22]. Specific applications in this area include manipulation of soft bio-inspired robots with many degrees of freedom [23], navigation and path planning applications of mobile robots and UAVs [24], [25], [26]. RL finds widespread applications in communications and networking [27]. It has been used in 5G enables UAVs with cognitive capabilities [28], cybersecurity [29], and edge computing [30]. Other domains where RL has been used include hospital decision-making [31], precision agriculture [32], and fluid mechanics [33]. It is of little surprise that RL has been extensively used to solve various problems arising in energy systems [34], [35], [36], [37], [38]. Another review article on the use of RL [38] considers three application areas frequency and voltage control as well as energy management.

RL is increasingly being used in HEMS applications and several review papers have already been published. The review article in [39] focuses on RL for HVAC and water heaters. The paper in [40] is based on research published between 1997 and 2019. The survey observes that only 11% of published research reports deployment of RL in actual HEMS. The article in [41] specifically focuses on occupant comfort in residences and offices. A more recent review on building energy management [42] focuses on deep neural networks based RL. A recent article [43] considers RL along with model predictive control in smart building applications. The article in [44] is a survey of RL in demand response.

In contrast to the previous reviews, the scope of our review is broad enough to cover all areas of HEMS, including HEMS interfacing with the energy grid. More importantly, it provides a comprehensive overview of all major RL methods, providing a sufficient level of explanation for readers' understanding. Therefore, this article would be of benefit for researchers in other areas of the energy systems, and beyond, to acquire a theoretical level understanding of basic RL techniques.

The rest of this article is organized in the following manner. Section 2. addresses the various elements of HEMS in greater details. Section 3. introduces basic ideas on reinforcement learning. Further details of value-based RL and associated deep architectures are discussed in Section 4., while policy-based and actor-critic architectures – the other class of RL algorithms are described in Section 5. Sections 6. And 7. Discuss the results of the present research survey. While Section 6. focuses on the domains to which RL was applied, Section 7. is a study on the classes of algorithms that were used. The article concludes in Section 8.

2. Home Energy Management Systems

HEMS refers to a slew of automation techniques that can respond to continuously or periodically changing the home/ building's internal as well as relevant external conditions, and without the need for human intervention. This section addresses the enabling technologies that make this an attainable goal.

2.1. Networking and Communication

All HEMS devices must have the ability to send/receive data with each other using the same communication protocol. HEMS provides the occupants with the tools that allow them to monitor, manage and control all the activities within the system. The advancements in technologies and more specifically in IoT-enabled devices and wireless communications protocols such as ZigBee, Wi-Fi, and Z-Wave made HEMS feasible [45], [46]. These smart devices are connected through a home area network (HAN) and/or to the internet, i.e. a wide area network (WAN).

ZigBee and Z-Wave are the two dominant communication protocols in today's home automation market. Z-Wave is better than ZigBee with the range of transmission (120m with three devices as repeaters vs 60m with two devices working as repeaters). When it comes to inter-brand operability, Z-Wave still has the advantage over ZigBee. However,

 doi:10.20944/preprints202208.0104.v1

ZigBee win the competition when it comes to the data rate of transmission and the maximum number of connected devices. Both protocols share many common features like they both use RF communication mode, and both are two-way communication protocols. Z-Wave is specially founded for home automation applications, while ZigBee is used in a wider range of places such as industrial, research, health care, and home automation [47]. Both protocols have established various connections with different companies and tens of hundreds of smart devices are using one of these protocols. A study done by [48] foresees that ZigBee is most likely to be the standard communication protocol for HEMS. However, due to the presence of numerous factors it is still difficult to tell with high certainty if this forecast would take place in future. It is also possible for an alternative communication protocol to emerge.

HEMS needs this level of connectivity to be able to get the electricity price from the smart grid through the smart meter and control all the system's elements accordingly (e.g., turn on/off the TV, set the thermostat settings, charge/discharge the battery., etc.). In some scenarios, HEMS uses the forecasting electricity prices to schedule the shiftable loads (e.g., washing machine, dryer, electric vehicle charging) [45].

2.2. Sensors and Controller Platforms

HEMS consists of smart appliances with sensors, these IoT-enabled devices communicate with the controller by sending and receiving data. They collect information from the environment and/or about their electricity usage using built-in sensors. The smart meter gathers information regarding the total consumers' consumption from the appliances, the peak load period, and electricity price from the smart grid.

The controller can be in the form of a physical computer located within the premises, that is equipped with the ability to run complex algorithms. An alternate approach is to leverage any of the cloud services that are available to the consumers through cloud computing firms.

The controller gathers information from the following sources: (*i*) the energy grid through the smart meter, which includes the power supply status and electricity price, (*ii*) the status of renewable energy and the energy storage systems, (*iii*) the electricity usage of each smart device at home, and (*iv*) the outside environment. Then process all the data through a computational algorithm to take specific action for each device in the whole system separately [5].

2.3. Control Algorithms

AI and machine learning methods are making deep inroads into HEMS [49], [10]. HEMS algorithms incorporated into the controller might be in the form of simple knowledge-based systems. These approaches embody a set of if-then-else rules, that may be crisp or fuzzy. However, due to their reliance on a fixed set of rules, such methods may not be of much practical use with real-time controllers. Moreover, they cannot effectively leverage the big amount of data available today [5]. Although it is possible to impart a certain degree of trainability to fuzzy systems, the structural bottleneck of consolidating all inputs using only conjunctions (and), and disjunctions (or) still persists.

Numerical optimization comprises of another class of computational methods for the smart home controller. These methods entail an objective function that is to be either minimized (e.g. cost) or maximized (e.g. occupant comfort), as well as a set of constraints imposed by the underlying physical HEMS appliances and limitations. Due to its simplicity, linear programing is a popular choice for this class of algorithms. More recently, game theoretic approaches have emerged as an alternative approach for various HEMS optimization problems [5].

In recent years, artificial intelligence and machine learning, more specifically deep learning techniques, have become popular for HEMS applications. Deep learning takes advantage of all the available data for training the neural network to predict the output

 doi:10.20944/preprints202208.0104.v1

and control the connected devices. It is very helpful to forecast the weather, load, and electricity price. Furthermore, it deal with non-linearities without using reverting to explicit mathematical models. Since 2013, there have been significant efforts directed at using deep neural networks within an RL framework [50], [51], that has met with great success.

3. Overview of Reinforcement Learning

3.1. Deep Neural Networks

A deep neural network (DNN) is a trainable highly nonlinear function approximator of the form $\mathbf{y}(\boldsymbol{\theta}): \mathcal{R}^M \to \mathcal{R}^N$ where M and N are the dimensionalities of the input and output spaces and $\boldsymbol{\theta}$ is its set of parameters. Structurally, the DNN consists of an input layer, and output layer, and at least one hidden layer. The input layer receives the DNN input vector $\mathbf{x}$. The neurons in any other layer receive as their inputs, the weighted outputs of neurons in the preceding layer. The weights of the DNN make up its weight parameter $\boldsymbol{\theta}$. For simplicity, we consider DNNs with scalar outputs so that $y(\boldsymbol{\theta}): \mathcal{R}^M \to \mathcal{R}$. The (true) output of the DNN is denoted as $y(\mathbf{x}|\boldsymbol{\theta})$, which is that of the sole neuron in the output layer.

In a typical regression application, the DNN's training set $\mathcal{S}$ consists of pairs $\big(\mathbf{x}(n), t(n)\big) \in \mathcal{S}$ where $n = 1, \ldots, |\mathcal{S}|$ is the sample index (for the sake of conciseness, this relationship is often denoted as $n \in \mathcal{S}$ in this article). The quantity $t(n)$ is the *target*, or desired output. During training, $\boldsymbol{\theta}$ is updated in steps so that for each input $\mathbf{x}(n)$, the DNN's output $y(n)$ is as close as possible to $t(n)$. Supervised learning algorithms aim to minimize the DNN's *loss function* $J_{\mathcal{S}}(\boldsymbol{\theta})$. The subscript $\mathcal{S}$ indicates that the loss is empirically estimated over the sample set $\mathcal{S}$. A popular choice of the latter is the sum of squared L_2 norms of the difference between the target and output for all samples in $\mathcal{S}$. In this case,

$$J_{\mathcal{S}}(\boldsymbol{\theta}) = \frac{1}{2}\frac{1}{|\mathcal{S}|}\sum_{n\in\mathcal{S}}\big(y(\mathbf{x}(n)|\boldsymbol{\theta}) - t(n)\big)^2. \tag{1}$$

Training the DNN comprises of multiple passes called epochs; each epoch comprising of one pass through all samples in $\mathcal{S}$. In *stochastic gradient descent*, with $\eta \ll 1$ being the *learning rate* the parameter $\boldsymbol{\theta}$ is incremented once for every sample $n \in \mathcal{S}$ as,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\big(y(\mathbf{x}(n)|\boldsymbol{\theta}) - t(n)\big)\nabla_{\boldsymbol{\theta}} y(\mathbf{x}(n)|\boldsymbol{\theta}). \tag{2}$$

This increment is equivalent to a single gradient step with $J(\boldsymbol{\theta}) = \frac{1}{2}\big(y(\mathbf{x}(n)|\boldsymbol{\theta}) - t(n)\big)^2$.

While SGD is useful in many online applications, *minibatch gradient descent* is the most common training method. In each epoch $\mathcal{S}$ are divided into non-overlapping minibatches $\mathcal{B}_k \subset \mathcal{S}$ (i.e. $\cup_k \mathcal{B}_k = \mathcal{S}$, and $k \neq l \Rightarrow \mathcal{B}_k \cap \mathcal{B}_l = \phi$). The parameter $\boldsymbol{\theta}$ is updated once for every minibatch $\mathcal{B}$ as,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\frac{1}{|\mathcal{B}|}\sum_{n\in\mathcal{B}}\big(y(\mathbf{x}(n)|\boldsymbol{\theta}) - t(n)\big)\nabla_{\boldsymbol{\theta}} y(\mathbf{x}(n)|\boldsymbol{\theta}). \tag{3}$$

One of the advantages of training in minibatches is that the trajectory taken by the training algorithm is straightened out, thereby speeding up convergence. It can be seen that the loss function is $J_{\mathcal{B}}(\boldsymbol{\theta})$, which is identical to that in Eqn. (1), with sample set $\mathcal{B}$.

Typically, the loss function includes an additional regularization term designed to keep the weights in $\boldsymbol{\theta}$ low in order to prevent overfitting; overfitting results in poorer performance after the DNN is deployed into the real world. Nowadays, faster training is accomplished by using extensions of gradient descent such as ADAM. These as well as many other important aspects of DNNs and training algorithms have not be addressed here; the above discussion minimally suffices to understand how DNNs are used in reinforcement learning (RL). A brief exposition to DNNs is available at [52]. For a rigorous treatment of DNNs, the interested reader is referred to [53].

3.2. Reinforcement Learning

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

An *agent* in RL is a learning entity, such as a deep neural network (DNN) that exerts control over a stochastic, external *environment* by means of a sequence of *actions* over time. The agent learns to improve the performance of its environment using *reward* signals that are derived from the latter. Rewards are quantitative metrics that indicate the immediate performance of the environment (e.g. average instantaneous user comfort). The sets $\mathbb{S}$ and $\mathbb{A}$ are the state and action spaces and can be discrete or continuous. Although RL can be extended to continuous domains, everywhere in this article it is assumed that all temporal signals are sampled at discrete, regularly spaced intervals. At each discrete time instance t, the current state $s_t \in \mathbb{S}$ of the environment is known to the agent, which then implements an action $a_t \in \mathbb{A}$. The environment transitions to the next state s_{t+1} with a probability $p(s_{t+1}|s_t, a_t)$ while returning an immediate reward signal $r_t \equiv r(s_t, a_t, s_{t+1})$; with $r: \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to \mathcal{R}$ denoting the environment's reward function that it unknown to the agent. The transition can be denoted concisely as $s_t \xrightarrow{a_t, r_t} s_{t+1}$. The overall schematic is depicted in Figure 1 (cf. [52]).

Instead of greedily aiming to improve the immediate reward r_t at every time instance t, the agent may be iteratively trained to maximize the sum of the immediate and the weighted future rewards, which is called the *return*,

$$R_t = r_t + \sum_{t'=1}^{T-t} \gamma^{t'} r_{t+t'}. \tag{4}$$

The quantity $\gamma \in [0,1]$ is called the *discount* factor. This lookahead feature prevents the agent to learn to implement greedy actions that fetch large instantaneous rewards, while adversely affect the environment later on. The process begins at time $t = 0$ and terminates at time $t = T$, the time *horizon*. The environment's initial state at $t = 0$ is $s_0 \in \mathbb{S}$ which may be probabilistic, with an initial state distribution p_0. It should be noted that if $T = \infty$, then the discount must necessarily be less than unity ($\gamma < 1$) so that the return R_t is bounded at all times t. The 5-tuple $(\mathbb{S}, \mathbb{A}, p, r, \gamma)$ defines a Markov decision process (MDP). The initial distribution p_0 is assumed to be subsumed by the transition probabilities p. The MDP can be viewed as an extension of a discrete Markov model.

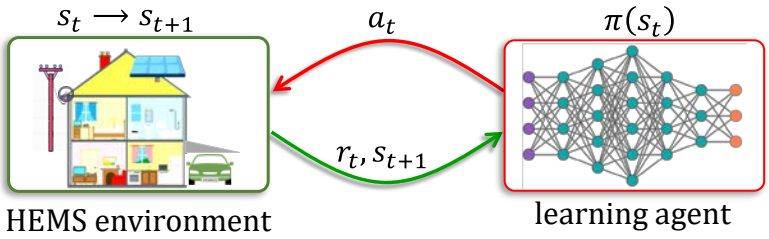

Figure 1. The quantities shown are associated with the transition $s_t \xrightarrow{a_t, r_t} s_{t+1}$.

During this episode, the action taken by the agent is in accordance with a *policy* $\pi \in \Pi$, where Π is the policy space. From the Markovian (memoryless) property of the MDP, it follows that the optimal action of an agent at each state in terms of its stated goal of maximizing the total reinforcement, is independent of all previous states of the environment, i.e. its history. Therefore, the action a_t taken by the agent at time t under policy π is based solely on the state s_t and the prior history of states and actions need not be taken into account. The policy can be deterministic or stochastic. A deterministic policy can be treated as a function $\pi: \mathbb{S} \to \mathbb{A}$ (see Figure 1) so that $a_t = \pi(s_t)$, whereas a stochastic policy π represents a probability distribution over $\mathbb{A}$ such that $a_t \sim \pi(s_t)$. In several domains, the probability distribution is determined from the nature of the application itself.

The entire sequence of states, actions, and rewards is an *episode*, denoted $\mathcal{E}$, so that,

$$\mathcal{E} \equiv s_0 \xrightarrow{a_0, r_0} s_1 \xrightarrow{a_1, r_1} s_2 \cdots \xrightarrow{a_{T-1}, r_{T-1}} s_T. \tag{5}$$

The overall objective of reinforcement learning is to maximize an objective function $J(\cdot)$. Let $J(\mathcal{E})$ denote the total return R_0 of a given episode $\mathcal{E}$. If the MDP is initialized to

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

any state $s \in \mathbb{S}$ at $t = 0$ (such that $s_0 = s$), the expected value of this return which is dependent on policy π may be expressed as,

$$J^{\pi}(s) = \mathbb{E}_{\pi}[R_0|s_0 = s]. \quad (6)$$

The operator $\mathbb{E}_{\pi}[\cdot]$ is the expectation when all episodes are generated by the MDP under policy π. When follows the MDP's initial state distribution, i.e. $s_0 \sim p$, the expectation may be denoted simply as J^{π} without any argument. This informal, function overloaded convention is adopted throughout this manuscript as there are other ways to define the objective function.

3.2. Taxonomy of RL Algorithms

RL methods can be classified in several ways. In *on-policy* RL, a referential policy π^*, such as that of a human, which is considered to be the *optimal policy* and is known a priori, and the goal is to learn a policy $\pi \cong \pi^*$. In *off-policy* approaches, the goal is to obtain the optimal policy π^* itself.

In *model-free* training, RL takes place with the agent connected to the real world environment, whereas in *model-based* RL, the agent is trained using a simulation platform to represent the environment. The classification of various approaches used in HEMS applications is shown in Figure 2.

In model based RL, as the transition probabilities and the reward function are available, the algorithm can be implemented in an *offline* manner. Of more practical interest is *online* RL where the agent can be trained in real time by interacting with the real physical environment as shown in Figure 1. Although online RL is considered to be model-free, practically all research papers report the use of HEMS models for training [40].

Another fundamental trichotomy of the plethora of RL approaches used today includes *value-based* RL, *policy-based* RL and *actor-critic* RL, the latter having emerged more recently. Actor-critic methods are hybrid approaches that borrow features from value-based as well as policy-based RL.

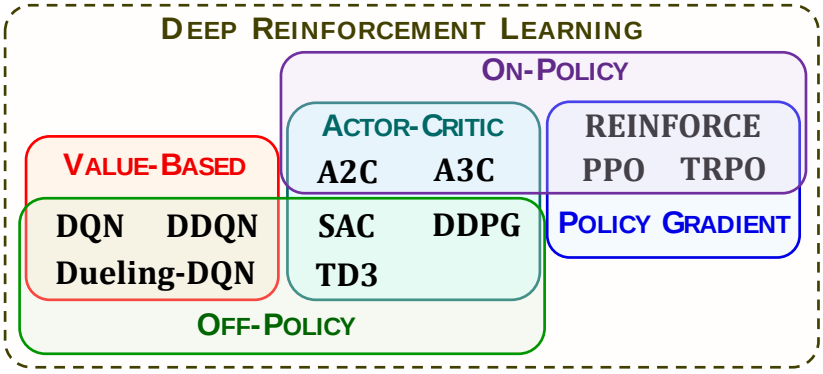

Figure 2. Taxonomy of Deep Reinforcement Learning. Classification of all deep reinforcement learning methods that are described in this article are shown. Also see [54].

4. Value Based Reinforcement Learning

Value based approaches are based on dynamic programming. Historically value-based RL, first proposed in [55], heralds the advent of the broad area of reinforcement learning as a distinct branch of AI. The formal definition of an MDP was introduced shortly thereafter [56], [57]. As noted earlier, an MDP is memoryless. An implication of this feature is that when the environment is in any given state $s_t = s$ at any instant t, the prior history $s_0 \xrightarrow{a_0, r_0} s_1 \xrightarrow{a_1, r_1} \cdots \xrightarrow{a_{t-1}, r_{t-1}} s_t$ is not of any consequence in deciding the future course of actions. Accordingly, one can define the *state-action value*, or Q-value of the state s and for every available action a as the expected return from state s when implementing action a,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

$$q(s,a) \triangleq \mathbb{E}_\pi[R_t|s_t = s, a_t = a]. \quad (7)$$

Referring to a specific policy π may be achieved by using a superscript in the above equation, so that the left hand ide of Eqn. (7) is written as $q^\pi(s,a)$. Even under a deterministic policy, $q^\pi(s,a)$ can still be defined for any action $a \neq \pi(s)$ merely by treating a as an *evaluative* action and following the policy at all future times. The Q-value function can be expressed recursively in the following manner,

$$q^\pi(s,a) = \sum_{s' \in \mathbb{S}} p(s'|s,a)\left(r(s,a,s') + \gamma \sum_{a' \in \mathbb{A}} \pi(a'|s')\, q^\pi(s',a')\right). \quad (8)$$

The Q-value function $q^\pi: \mathbb{S} \times \mathbb{A} \to \mathcal{R}$ can be defined irrespective of whether the policy is deterministic or stochastic.

A stochastic policy is intrinsic to many real world applications. For instance, in order to decrease the ambient temperature by manually lowering the thermostatic setting, the final setting involves a degree of randomness arising from human imprecision. In multi-agent environments, the best course may often be to adopt a stochastic policy. As an example, in a repeated game of rock-paper-scissors, randomly selecting each action ('rock', 'paper', or 'scissors') with equal probabilities of ⅓ is the only policy that would ensure that the probability of losing does not exceed that of winning.

From a machine learning standpoint, stochastic policies help explore and assess the effects of the entire repertoire of actions available in $\mathbb{A}$. Such *exploration* is critical during the initial stages of the learning algorithm. The two most commonly used stochastic policies are the *ϵ-greedy* and the *softmax* policies. Under an ϵ-greedy policy π, the probability of picking an action a when the environmental state is s is given by,

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{|\mathbb{A}|} + (1-\epsilon), & a = \underset{a' \in \mathbb{A}}{\operatorname{argmax}}\, q(s,a') \\ \frac{\epsilon}{|\mathbb{A}|}, & a \neq \underset{a' \in \mathbb{A}}{\operatorname{argmax}}\, q(s,a') \end{cases}. \quad (9)$$

It is always a good idea to lower the parameter ϵ steadily so that as learning progresses, the agent is greedier - being likelier to select actions with the highest Q-values, $\operatorname{argmax}_{a'} q(s,a')$.

The softmax policy is another popular method to incorporate exploration into a policy. The expression below is the probability of action a being selected under such a policy π,

$$\pi(a|s) = \left[\sum_{a' \in \mathbb{A}} e^{\frac{q(s,a')}{\tau}}\right]^{-1} e^{\frac{q(s,a)}{\tau}}. \quad (10)$$

Initialized to a high value, the Gibbs-Boltzmann parameter τ may be steadily lowered as the learning algorithm progresses, so that the policy becomes increasingly *exploitative*, that is take the action with the highest Q-value. Unless specified otherwise, it shall be assumed hereafter that the policy space Π is stochastic so that actions follow probability distribution ($a \sim \pi(s)$).

Instead of an evaluative action a, suppose the policy π is applied from state s (so that either $a = \pi(s)$ or $a \sim \pi(s)$), then the expected return is called the state's *value*,

$$v(s) \triangleq \mathbb{E}_\pi[R_t|s_t = s]. \quad (11)$$

As with the Q-value function, the policy π becomes explicit if the value of s is written as $v^\pi(s)$. The value of s can also be expressed in terms of Q-values as,

$$v(s) \equiv \sum_a \pi(a|s) q(s,a) = \mathbb{E}_{a \sim \pi(s)}[q(s,a)]. \quad (12)$$

The difference between value of any state s and the Q-value of implementing an action a from s under policy is the advantage function, so that,

$$A(s) \equiv q^\pi(s,a) - v^\pi(s). \quad (13)$$

Although the preferred notation in this manuscript is to use lowercase letters to denote variables, the advantage function is represented using uppercase as the lowercase a is reserved to denote an action.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

The value function $v: \mathbb{S} \to \mathcal{R}$ allows the optimal policy π^* to be defined in a formal manner. If the objective function with the MDP initialized to some $s_0 \in \mathbb{S}$ is defined as in Eqn. (6), then it is evident from Eqn. (10) that $J^{\pi}(s_0) = v(s_0)$. Furthermore, if the MDP visits state $s_t = s$ at the instant t, then from Eqn. (4),

$$\mathbb{E}_{\pi}[R_0] = r_0 + \gamma r_1 + \cdots + \gamma^{t-1} r_{t-1} + \gamma^t v^{\pi}(s). \tag{14}$$

At this stage, we invoke the memoryless property of the underlying MDP. At instant t the partial sum of the terms $r_0 + \gamma r_1 + \cdots + \gamma^{t-1} r_{t-1}$ in the right hand side are part of the episode's history, while $\gamma^t v^{\pi}(s)$ is the expected future return. The optimal policy at every such state s is to adopt the one that maximizes $v^{\pi}(s)$, so that,

$$\pi^*(s) \triangleq \underset{\pi \in \Pi}{\operatorname{argmax}}\, v^{\pi}(s). \tag{15}$$

When the policy π^* is deterministic, it can also be inferred that the optimal action from state s is to select the action with the highest Q-value. From Eqn. (15) it directly follows that,

$$a^* \triangleq \underset{a \in \mathbb{A}}{\operatorname{argmax}}\, q^*(s, a). \tag{16}$$

The Q-value $q^*(s, a)$ is equal to $q^{\pi^*}(s, a)$. It can be established that the Q-values corresponding to the optimal policy are higher than those associated with other policies, i.e. $q^*(s, a) \geq q^{\pi}(s, a)$ [58].

The *Bellman's equation* for optimality follows from the above consideration,

$$q^*(s, a) = \sum_{s' \in \mathbb{S}} p(s'|s, a) \left(r(s, a, s') + \gamma \max_{a' \in \mathbb{A}} q^*(s', a') \right). \tag{17}$$

The difference between Eqn. (8) and Eqn. (17) is in the second term in each summand. The policy-based Q-value in Eqn. (8) is replaced with the maximum Q-value in Eqn. (17). A mathematically rigorous coverage of various RL methods can be found in the seminal book [59] that is available online.

4.1. Tabular Q-Learning

The simplest possible implementation of the Q-learning algorithm is *tabular Q-learning* where an $|\mathbb{S}| \times |\mathbb{A}|$ sized array is maintained to store $q(s, a)$ for every state-action pair [60]. Initialized to either zeros or small random values, the tabular entries are periodically updated. As it is an online approach, Q-learning does not use transition probabilities ($p(s'|s, a)$). For each transition $s \xrightarrow{a,r} s'$, the tabular entry for $q(s, a)$ is incremented as show below,

$$q(s, a) \leftarrow (1 - \eta) q(s, a) + \eta t. \tag{18}$$

The quantity η is the *learning rate*; usually $\eta \ll 1$. The quantity t is the target,

$$t = r + \gamma \max_{a' \in \mathbb{A}} q(s', a'). \tag{19}$$

In order to impart an exploratory component to Q-learning, the action a must be selected probabilistically as in Eqn. (9) or Eqn. (10). In many cases, increments are applied in real time at the end of each time instance. It can be shown mathematically that the tabular entries converge towards the maximum values, $q^*(s, a)$ [58], implying that Q-learning is an off-policy approach. The fully trained agent can select actions as per Eqn. (16) during actual use, with the reasonable expectation that the optimal policy is implemented.

SARSA (State-Action-Reward-State-Action) [61], [59] is the on-policy RL algorithm that can be implemented in a tabular manner. The update rule for SARSA is identical to the earlier expression in Eqn. (19). However, since SARSA is an on-policy algorithm, the target is policy specific and is given by,

$$t = r + \gamma q\big(s', \pi(s')\big). \tag{20}$$

Both Q-learning and SARSA use the tabular entries $q(s', a')$ of the environment's new state s' following the transition $s \xrightarrow{a,r} s'$. The difference is in how the entries are used. Whereas Q-learning uses the tabular entry corresponding to the action a' with the highest $q(s', a')$, SARSA applies the specified policy π, using $q(s', a')$, the Q-value of the action $a' = \pi(s')$. This difference is analogous to that between Eqn. (17) and Eqn. (8).

Tabular Q-learning and SARSA can handle continuous state as well as action spaces by discretizing them into a finite and tractable number of subdivisions. Unfortunately,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

these tabular learning methods cannot be applied in many large scale application domains. Replacing tables with DNNs helps obviate this limitation. Such an arrangement is called a *Deep Q-network* (DQN), which is discussed next.

When the state space $\mathbb{S}$ is too large (e.g. $|\mathbb{S}| \approx 7.73 \times 10^{45}$ in chess), tabular learning becomes prohibitively expensive not only in terms of storage requirements but also in terms of computational time required by the training algorithm. DQNs, which are meant for off-policy training, are well equipped to handle such large discrete as well as continuous state spaces [62]. It must be noted that $|\mathbb{A}|$, the cardinality of the action space must be tractably small. In DQNs, the mapping from every state action pair (s, a) to its Q-value is carried out by means of DNNs.

In reality, the DNN input is some feature vector $\boldsymbol{\varphi}(s)$ of the state s, where $\boldsymbol{\varphi}: \mathbb{S} \rightarrow \mathcal{R}^D$ and D is the dimensionality of the feature space. In the same manner, the action a can be represented in terms of its feature vectors. However, as $|\mathbb{A}|$ is small, it is assumed that the action a itself is the other input. In practice, a unary encoding scheme may be used to represent actions. For instance, if $|\mathbb{A}| = 4$, the four discrete actions may be encoded as 0001, 0010, 0100, and 1000. Under these circumstances, the actual input to the DNN is $(\boldsymbol{\varphi}(s), a)$, and its actual output is $y(\boldsymbol{\varphi}(s), a|\boldsymbol{\theta})$. For simplicity we will treat the DNN input as (s, a) and the output as $q(s, a|\boldsymbol{\theta})$, that is, $(s, a) \equiv (\boldsymbol{\varphi}(s), a)$, and $q(s, a|\boldsymbol{\theta}) \equiv y(\boldsymbol{\varphi}(s), a|\boldsymbol{\theta})$. The Q-value $q(s, a|\boldsymbol{\theta})$ is conditioned in terms of the weight parameter $\boldsymbol{\theta}$ in this manner so as to explicitly reflect its dependence on the latter.

4.2. Deep Q-Networks

There are two possible ways in which the mapping of a state-action pair (s, a) to its Q-value $q(s, a|\boldsymbol{\theta})$ can be accomplished, which are as follows (see Figure 3).
(*i*) A different DNN for each action is maintained, so that the total of DNNs is $|\mathbb{A}|$. The state s, encoded in a suitable manner, serves as the common input to all the DNNs.
(*ii*) A single DNN with separate inputs for state s and action a is maintained and its output is $q(s, a|\boldsymbol{\theta})$. While this manner of storing Q-values requires the use of only a single DNN, in order to obtain $\max q(s, a|\boldsymbol{\theta})$, the actions must be applied sequentially to it.

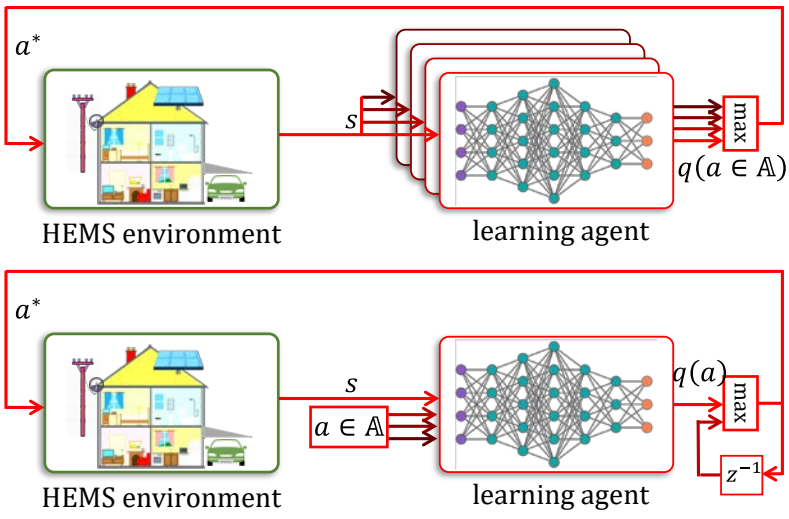

Figure 3. Deep Q-Network Layouts. One scheme uses a uses a separate DNN for each action (top). uses only one DNN that receives actions as another input (bottom).

Stochastic gradient descent can be applied in a straightforward fashion to train the weight parameter $\boldsymbol{\theta}$ as in Eqn. (2) for the squared error loss $\frac{1}{2}\big(t - q(s, a|\boldsymbol{\theta})\big)^2$, and with the DNN's output y now being $q(s, a|\boldsymbol{\theta})$,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta(q(s, a|\boldsymbol{\theta}) - t)\nabla_{\boldsymbol{\theta}} q(s, a|\boldsymbol{\theta}). \tag{21}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

This simple approach is the *neural-fitted Q-iteration* (NFQ) that was proposed in [63]. The target $t(n)$ is determined in accordance with Eqn. (19) with $q(s',a'|\boldsymbol{\theta})$ used to obtain the target t so that $t = r + \gamma \max_{a' \in \mathbb{A}} q(s',a'|\boldsymbol{\theta})$. When the learning agent interacts with the environment, the actions are generally selected using the ϵ-greedy method shown in Eqn. (9).

Temporal correlation in real-time training samples is an unfortunate drawback when directly implementing stochastic gradient descent. Unlike in tabular learning, in DQN updating $\boldsymbol{\theta}$ changes not only the output $q(s,a|\boldsymbol{\theta})$ for the relevant state-action pair (s,a) but the Q-values $q(s',a'|\boldsymbol{\theta})$ of every other pair $(s',a') \neq (s,a)$ as well. The change may be barely noticeable when s' is at a large distance from s within the feature space $\boldsymbol{\varphi}(\mathbb{S})$; unfortunately this is not usually the case in most real world domains. Consider two successive transitions $s_t \xrightarrow{a_t r_t} s_{t+1} \xrightarrow{a_{t+1} r_{t+1}}$. Due to the property of temporal correlation between successive states, it is highly reasonable to expect that $\|\boldsymbol{\varphi}(s_t) - \boldsymbol{\varphi}(s_{t+1})\|$ is very small. Therefore, applying Eqn. (21) to update $q(s_t,a_t|\boldsymbol{\theta})$ will have an undesirable yet pronounced effect on $q(s_{t+1},a_{t+1}|\boldsymbol{\theta})$. A similar argument holds for time sequences of actions as well.

To address the ill effects of temporal correlatedness, DNN training is carried out only after the completion of an entire episode or multiple episodes, during which time the DQN agent is allowed to exert control over the environment, while $\boldsymbol{\theta}$ remains unchanged. All training samples are stored in an *experience replay buffer* $\mathcal{B}$ [64], which plays the role of a minibatch described in the previous section. After a large number of training samples are gathered in $\mathcal{B}$, it is shuffled randomly before incrementing $\boldsymbol{\theta}$ which may be carried out either as in Eqn. (21), or through minibatch gradient descent as indicated earlier in Eqn. (3) with $q(s,a|\boldsymbol{\theta})$ replacing $y(\mathbf{x}|\boldsymbol{\theta})$. For convenience, such a minibatch incremental rule is provided below,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \big(q(s(n),a(n)|\boldsymbol{\theta}) - t(n)\big) \nabla_{\boldsymbol{\theta}} q(s(n),a(n)|\boldsymbol{\theta}). \tag{22}$$

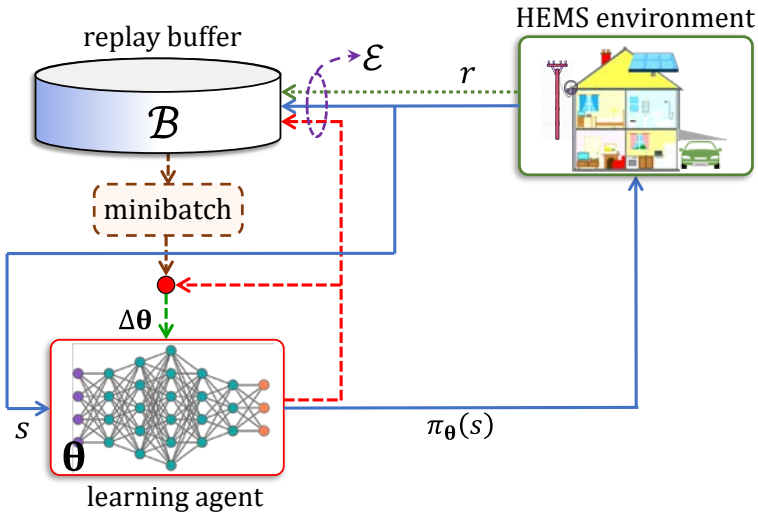

Figure 4. Replay Buffer. Shown are the replay buffer, environment and agent. The pathways are involved during the agent's interaction with the environment (solid blue) and training (dashed red).

The buffer $\mathcal{B}$ is flushed before the next cycle begins with the updated parameter $\boldsymbol{\theta}$. An improvement over this scheme is *prioritized replay* [65], where the probability of a get-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

ting selected chosen for a training step is proportional to $\left(t - q(s,a|\boldsymbol{\theta})\right)^2 + \epsilon$. The relatively small constant $\epsilon > 0$ is added to the squared loss term to ensure that all samples have non-zero probabilities. Figure 4 illustrates the use of a replay buffer.

Target non-stationarity is another closely related problem that arises in DQNs, one that is not seen in tabular Q-learning. For any given sample transition $s(n) \xrightarrow{a(n),r(n)} s'(n)$ as the DNN weight parameter $\boldsymbol{\theta}$ is incremented in accordance with Eqn. (21), an undesirable effect is that the target $t(n)$ also changes. This is because the same DNN is used to obtain $q(s'(n), a'|\boldsymbol{\theta})$. Target non-stationarity is handled by storing an older copy $\boldsymbol{\theta}^{\text{target}}$ of the *primary DNN* $\boldsymbol{\theta}$ in memory, effectively using a separate *target* DNN which is parametrized by $\boldsymbol{\theta}^{\text{target}}$. The target $t(n)$ is obtained using as,

$$t(n|\boldsymbol{\theta}^{\text{target}}) = r(n) + \gamma \max_{a' \in \mathbb{A}} q(s'(n), a'|\boldsymbol{\theta}^{\text{target}}). \quad (23)$$

The target DNN's weight parameter is updated infrequently, and only after $\boldsymbol{\theta}$ undergoes a significant amount of training. In this manner, the targets remain stationary when training the primary DNN's parameter $\boldsymbol{\theta}$ so that gradient descent steps can be implemented in a straightforward manner using terms $\frac{1}{2}\left(q(s(n), a(n)|\boldsymbol{\theta}) - t(n|\boldsymbol{\theta}^{\text{target}})\right)^2$ in the loss function $J(\boldsymbol{\theta})$. This scheme is shown in Figure. 5.

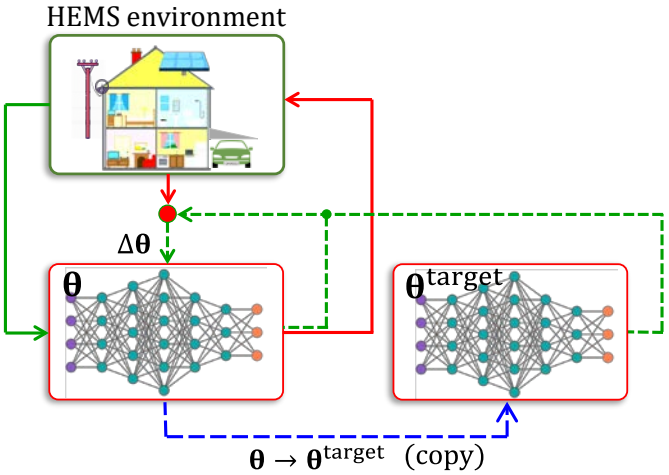

Figure 5. Use of Target Network. The scheme used to correct temporal correlatedness is shown. Pathways for control (solid red), learning (dashed green), and intermittent copying (dashed, thick blue) are shown. The replay buffer has been omitted for simplicity).

Overestimation bias [66], [67] is another problem frequently encountered in stochastic environments. This is an outcome of the maximization operation. As an example, consider an MDP with $\mathbb{S} = \{A, B, C\}$ where C is the terminal state. This is shown in Figure 6. The action space $\mathbb{A} = \{a_k | k = 1,2,\ldots,N\}$ where N is relatively large, is available to the agent. From state A, only action a_N leads to B whereas the remaining ones a_1 thru a_{N-1} lead to C. The reward received from state A is always zero, (i.e. $r(A, a_k) = 0$. From state B all actions lead to C, with the reward being either -3 or $+1$ and with equal probabilities of N^{-1}. In other words, the possible transitions are $A \xrightarrow{a_N,0} B$, $A \xrightarrow{a_{k\neq N},0} C$, $B \xrightarrow{a_k,r\in\{-3,1\}} C$. Since the rewards of -3 and 1 have the same probability when the environment transitions from B to C, the expected reward from B to C is -1 i.e. $\mathbb{E}[r(B, a_k, C)] = -1$. For simplicity, let us assume that $\eta = 1$. The Q-values for some actions would be updated to -3, whereas those of others, to $+1$. Since N is large enough, it is very likely that at least one of them, say a' has the higher of the two. Consider the Q-values of actions from state A. It is clear that for $k = 1$ through $N - 1$, $q(A, a_k) = 0$. However, when the agent selects action a_N from A, thereby reaching B, the operation $\max_{a\in\mathbb{A}} q(B, a)$ is likely to return $+1$ so that $q(A, a_N)$ would be updated to $\gamma \max_{a\in\mathbb{A}} q(B, a) = \gamma$. This makes a_N seem to be the optimal action from state A, when in fact it is the worst choice in $\mathbb{A}$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

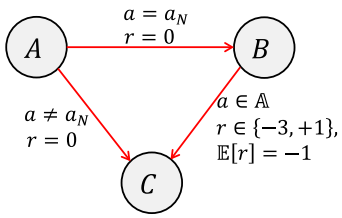

Figure 6. **Overestimation Bias.** This example is used to illustrate the effect of overestimation bias (see text for complete explanation).

Double Q-learning [68] is a popular approach to circumvent overestimation bias in off-policy RL (see Figure 7). Although first proposed in a tabular setting [66], more recent research implements double Q-learning in conjunction with DNNs, which is called the *double deep Q-network* (DDQN). It incorporates two DNNs with the parameters $\boldsymbol{\theta}^1$ and $\boldsymbol{\theta}^2$. Samples are collected by implementing actions using their mean Q-values, $\frac{1}{2}\left(q(s,a|\boldsymbol{\theta}^1) + q(s,a|\boldsymbol{\theta}^2)\right)$. For each sample transition $s(n) \xrightarrow{a(n),r(n)} s'(n)$ in $\mathcal{B}$ during training, one of the two DNNs, say DNN j ($j \in \{1,2\}$), is picked randomly and with equal probability to compute the target, and the other DNN, $\bar{\jmath}$ is trained with it. Whence,

$$\begin{cases} t(n) = r(n) + \gamma \max\limits_{a' \in \mathbb{A}} q(s'(n), a'|\boldsymbol{\theta}^j) \\ \boldsymbol{\theta}^{\bar{\jmath}} \leftarrow \boldsymbol{\theta}^{\bar{\jmath}} - \eta \sum\limits_{n \in \mathcal{B}} \left(q(s(n), a(n)|\boldsymbol{\theta}^{\bar{\jmath}}) - t(n)\right) \nabla_{\boldsymbol{\theta}^{\bar{\jmath}}} q(s(n), a(n)|\boldsymbol{\theta}^{\bar{\jmath}}) \end{cases}. \quad (24)$$

This is the manner of updating that was originally proposed [66].

An extension of DDQN is *clipped DDQN* [69], [70], [71], where the target is obtained by taking the minimum of the Q-values, $q(s'(n), a'|\boldsymbol{\theta}^1)$ and $q(s'(n), a'|\boldsymbol{\theta}^2)$ instead of at random as before,

$$t(n) = r(n) + \gamma \min_{j \in \{1,2\}} \max_{a' \in \mathbb{A}} q(s'(n), a'|\boldsymbol{\theta}^j). \quad (25)$$

Each DNN has a 0.5 probability of getting trained with the transition sample.

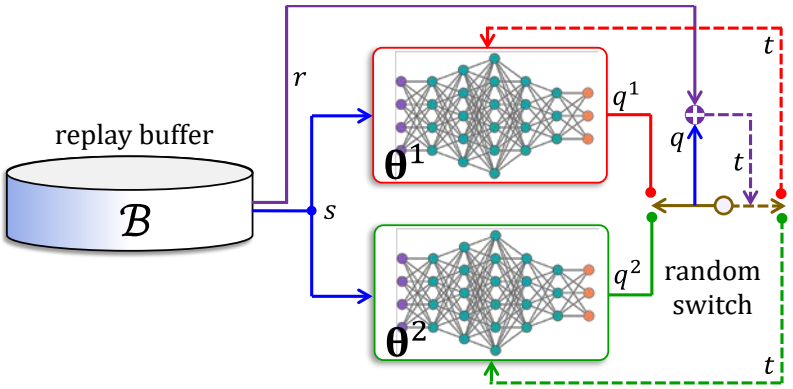

Figure 7. **Double DQN.** One DQN (θ^1 or θ^2) is picked at random and its Q value (q^1 or q^2) is used to obtain the target (t), which is used to train the other DQN. For simplicity only the pathways involved in training are shown. The target pathways are depicted with dotted lines.

Dueling DQN architectures (Dueling-DQN) [72] use a different scheme to avoid over-estimation bias (see Figure 8). It divides the state-action value $q(s,a)$ into two parts, the state value $v(s)$ and the state-action advantage $A(s,a)$. As shown in Eqn. (13), $q(s,a)$ is the difference between the two quantities. The advantage of action a in state s, $A(s,a)$ is the expected gain in the return obtained by picking action a. The DNN architecture consists of an input layer for the state s. After a few initial preprocessing layers, it splits into two separate pathways, each of which is a fully connected DNN. Letting the symbols $\boldsymbol{\theta}^V$ and $\boldsymbol{\theta}^A$ denote the weight parameters of the pathways, the scalar output of the value pathway is the state's value, $v(s|\boldsymbol{\theta}^V)$ and the output of the advantage pathway is an $|\mathbb{A}|$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

dimensional vector comprising of the advantages $A(s,a|\boldsymbol{\theta}^A)$ of all available actions in $\mathbb{A}$. The Q-value of the state-action pair (s,a) can be obtained in a straightforward manner as provided in the following equation,

$$q(s,a|\boldsymbol{\theta}) = v(s|\boldsymbol{\theta}^V) - |\mathbb{A}|^{-1}\sum_{a'} A(s,a'|\boldsymbol{\theta}^A). \tag{26}$$

The quantity $\boldsymbol{\theta}$ denotes the set of all weight parameters of the dueling-DQN, including $\boldsymbol{\theta}^V$ and $\boldsymbol{\theta}^A$ as well as those present in the earlier preprocessing layers.

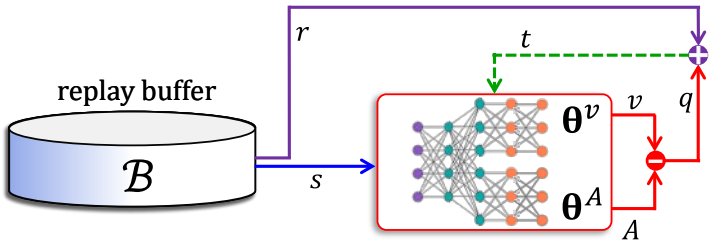

Figure 8. Dueling DQN. Shown is the dueling DQN architecture. The two outputs of the DNN are parametrized by $\boldsymbol{\theta}^v$ and $\boldsymbol{\theta}^A$. The target pathway (dotted green) is for training.

5. Policy Based and Actor-Critic Reinforcement Learning

Unlike the methods detailed in the preceding section, policy based RL does not require the estimates of the expected return from any given state. Initialized with an arbitrary policy π, tabular policy RL is an iterative process comprising of two steps [59], [73]. Policy evaluation is carried out in the first stage, where Q-values $q^\pi(s,a)$ are learned as shown in Eqn. (18) and Eqn. (20). In the second step, the policy is refined by defining the action for each state as shown in Eqn. (16). The two step process is repeated until the policy can be refined no further.

Gradient descent policy learning methods do not directly draw upon it in the same way that value based learning did. These methods are realized through parametrized agents, e.g. DNNs. An attractive feature of deep policy RL is its intrinsic ability to handle continuous states as well as continuous actions.

5.1. Deep Policy Networks

Policy gradient uses an experience replay buffer $\mathcal{B}$ is the same manner as a DQN. The buffer stores full episodes of sequences. Instead of using Eqn. (6), it is convenient to directly express the loss function in terms of episodes $\mathcal{E}$ and the DNN's weight parameter $\boldsymbol{\theta}$, in the following manner,

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]. \tag{27}$$

Policy gradient methods try to maximize this loss. The operator $\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[\cdot]$ is the expected with the DNN agent operating under the probabilistic policy $\pi_{\boldsymbol{\theta}}$. The initial state s_0 in the above expression is implicitly defined in $\mathcal{E}$. Moreover, the distribution of s_0 within $\mathbb{S}$ is in accordance with the underlying MDP. The quantity $R(\mathcal{E})$ is the total return R_0 of episode $\mathcal{E}$ starting from $t=0$.

Note that for a transition $s \xrightarrow{a} s'$, the reward $r(s,a,s')$ is a feedback signal that is determined by the environment (such as a home or residential complex) which is external to the agent. So is the discounted, aggregate return $R(\mathcal{E})$, which is also equal to that in Eqn. (4). No function $R: \mathbb{S}^T \times \mathbb{A}^T \to \mathcal{R}$ that maps a sequence of states and action of time horizon T to a return is available to the agent. Consequently, a straightforward gradient descent step in the direction of $\nabla_{\boldsymbol{\theta}} R(\mathcal{E})$ cannot be applied. In an apparent paradox, it turns out that its expected value $\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]$, can be differentiated by the agent, which is also the rationale behind expressing the loss as in Eqn. (27). This is due to a mathematical result known as the *policy gradient theorem* [74], [14], [75].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

The policy gradient theorem establishes the theoretical foundation for the majority of deep policy gradient methods. The theorem can be stated in mathematical terms as below,

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})] = \mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})\nabla_{\boldsymbol{\theta}}\log p(\mathcal{E}|\boldsymbol{\theta})]. \tag{28}$$

The significance of the theorem is that the gradient of the expected return $\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]$ does not require the use of the gradient of the return $R(\mathcal{E})$. Only the log probability of episode $\mathcal{E}$ must be differentiated. Fortunately, this gradient can readily be computed by the DNN agent. The probability of a transition $s_t \xrightarrow{a_t,r_t} s_{t+1}$ in $\mathcal{E}$ (see Eqn. (5)) is the product $\pi_{\boldsymbol{\theta}}(a_t|s_t)p(s_{t+1},r_t|s_t,a_t)$; its logarithm is $\log\pi_{\boldsymbol{\theta}}(a_t|s_t) + \log p(s_{t+1},r_t|s_t,a_t)$. The second term is intrinsic to the environment, and independent of the DNN so that differentiating it w.r.t. $\boldsymbol{\theta}$ is zero. Since $\log p(\mathcal{E}|\boldsymbol{\theta})$ is the product $p(s_0)\prod_t \pi_{\boldsymbol{\theta}}(a_t|s_t)p(s_{t+1},r_t|s_t,a_t)$, we arrive at the following interesting result,

$$\nabla_{\boldsymbol{\theta}}\log p(\mathcal{E}|\theta) = \frac{1}{T}\sum_t \nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)\,. \tag{29}$$

The left hand side of Eqn. (28) to be estimated rather easily using this expression. This is because the policy $\pi_{\boldsymbol{\theta}}$ is, in fact, is based on the DPN output. Whereas [74] uses softmax policies as in Eqn. (10), it is quite usual in later research to adopt Gaussian policies (cf. [14]). Since $\pi_{\boldsymbol{\theta}}$ is the same policy that is used to obtain transition samples, Eqn. (28) can be used for on-policy learning.

The expected gradient $\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]$ can be estimated as the average of several Monte Carlo samples of episodes (also called *rollouts*) $\mathcal{E}(n), n = 1,\ldots,N$ that are stored in $\mathcal{B}$. This provides an estimate of the gradient of the loss function defined in Eqn. (27). An early policy gradient method, *REINFORCE* [62] uses Eqn. (29) to increment $\boldsymbol{\theta}$. The REINFORCE on-policy update rule is expressed as,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta\frac{1}{|\mathcal{B}|T}\sum_{n\in\mathcal{B},t}\big(R\big(\mathcal{E}(n)\big) - b\big)\sum_t \nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}\big(a_t(n)|s_t(n)\big)\,. \tag{30}$$

In the above expression, it is assumed for simplicity that the time horizon is fixed across all N samples. The quantity b is called the *baseline* [76]. It can be set to zero in the basic implementation of policy learning.

Unfortunately, when the bias $b = 0$, the variance in the set of samples of the form, $R\big(\mathcal{E}(n)\big)\sum_{t=0}^{T-1}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}\big(a_t(n)|s_t(n)\big)$ becomes too large. This in turn requires a very large number of Monte Carlo episode samples to be collected. Including the baseline in Eqn. (30) that is close to $\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]$ helps reduce the variance to tractable limits. The theoretical optimal baseline estimate is given by,

$$b = \frac{\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})(\nabla_{\boldsymbol{\theta}}\log p(\mathcal{E}|\boldsymbol{\theta}))^2]}{\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[(\nabla_{\boldsymbol{\theta}}\log p(\mathcal{E}|\boldsymbol{\theta}))^2]}. \tag{31}$$

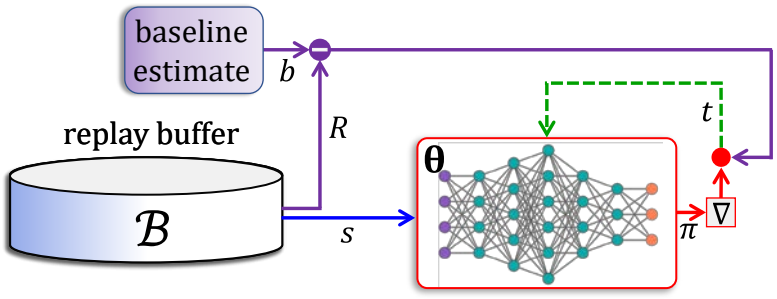

Figure 9. **Policy Gradient with Baseline.** Shown is the overall scheme used in REINFORCE with the baseline. There are different ways to implement the baseline.

There are ways to obtain reasonable baseline estimates in practice that reduce the variance without affecting the bias [76], [77]. The purpose of actor critic architectures, which will be described subsequently, are also designed to obtain reliable bias estimates.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

Before proceeding further, we will make improvements to Eqn. (30) on the basis of the following two observations.

The first observation is that in Eqn. (30) the gradient $\sum \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\big(a_t(n)|s_t(n)\big)$ linked with $\mathcal{E}(n)$ is weighted by $R\big(\mathcal{E}(n)\big) - b$ in the outer summation. In this manner, the episode $\mathcal{E}(n)$ would receive a higher weight if it fetched a higher return. However the weighting scheme is rather arbitrary. For instance, with $b = 0$ if all returns were non-negative, then all gradients would receive positive weights. On the other hand, suppose the bias b were to be replaced with the expected return, then the gradients of the episodes with lower than expected returns would receive negative weights, whereas those with better than expected returns would be assigned positive weights. Using Eqn. (11) it is observed that the bias b is also the value of the starting state $v(s_0)$. Our first improvement would be to replace the bias with a value function.

The second observation is subtler, requiring the scrutiny of the weighting scheme at each time instant t. To simplify the discussion, it will be assumed that the discount $\gamma = 1$. Consider the episode $\mathcal{E}(n)$ consisting of transitions of the form $s_t(n) \xrightarrow{a_t(n), r_t(n)} s_{t+1}(n)$. Ignoring b, the corresponding term in the inner summation, which is $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\big(a_t(n)|s_t(n)\big)$, is weighted by the return $R\big(\mathcal{E}(n)\big) = r_0(n) + \cdots + r_{t-1}(n) + r_t(n) + \cdots + r_T(n)$. At time instant t, the prior rewards $r_0(n)$ until $r_{t-1}(n)$ represent the past history of the episode $\mathcal{E}(n)$; it has no role in how good the action $a_t(n)$ was from state $s_t(n)$. Removing these terms, the weight becomes $r_t(n) + r_{t+1}(n) + \cdots + r_T(n)$, which is, in fact, $q(s_t(n), a_t(n)|\boldsymbol{\theta})$. Replacing $R\big(\mathcal{E}(n)\big)$ in Eqn. (30) with $q(s_t(n), a_t(n)|\boldsymbol{\theta})$ is the other improvement. Once the prior history of the episode is removed, the bias must be set to $v(s_t(n)|\boldsymbol{\theta})$ instead of $v(s_0)$.

From the above discussion, it is seen that each factor $R\big(\mathcal{E}(n)\big) - b$ in Eqn. (30) should be replaced with $q(s_t(n), a_t(n)|\boldsymbol{\theta}) - v(s_t(n)|\boldsymbol{\theta})$. From Eqn. (13), this is the advantage function $A(s_t(n), a_t(n)|\boldsymbol{\theta})$, so that we replace the step original increment rule with the following update rule,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \frac{1}{|\mathcal{B}|T} \sum_{n \in \mathcal{B}, t} A(s_t(n), a_t(n)|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\big(a_t(n)|s_t(n)\big). \tag{32}$$

5.2. Natural Gradient Methods

One of the problems associated with policy gradient learning approaches as in Eqn. (30) or Eqn. (32) is choosing an adequate learning rate η. Too small a value of η would necessitate a large training period, whereas a value that is too large would produce a 'jump' in $\boldsymbol{\theta}$ large enough to yield a new policy that is too different from the previous one. Although there are several reliable methods to address this effect in gradient descent for supervised learning as in Eqn. (3), it is too pronounced in RL diminishing the efficacies of such methods. In extreme cases, a seemingly small increment along the direction of the gradient may lead to irretrievable distortion of the policy itself. The underlying reason behind this limitation is that unlike in Eqn. (1), the loss function in Eqn. (25) incorporates a probability distribution.

The change in any policy whenever a perturbation is applied to the parameter should not be quantified in terms of the norm $\|\boldsymbol{\theta} - \boldsymbol{\theta}^{\text{old}}\|$, but using the Kullback-Leibler divergence between to the distributions $\pi_{\boldsymbol{\theta}}$ and $\pi_{\boldsymbol{\theta}^{\text{old}}}$ [78]. The K-L divergence is denoted as $D_{KL}\left(\pi_{\boldsymbol{\theta}} || \pi_{\boldsymbol{\theta}^{\text{old}}}\right)$ where the new and previous values of the parameter are $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^{\text{old}}$. The Hessian (2nd derivative) of $D_{KL}\left(\pi_{\boldsymbol{\theta}} || \pi_{\boldsymbol{\theta}^{\text{old}}}\right)$ is known as the Fisher information matrix $\mathbf{F}_{\boldsymbol{\theta}}$. The increment $\Delta\boldsymbol{\theta}$ applied to $\boldsymbol{\theta}$ should be in proportion to $\mathbf{F}_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathcal{E} \sim \pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]$, which is referred to as the *natural gradient* [79] of the expectation $\mathbb{E}_{\mathcal{E} \sim \pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]$. The Fisher information matrix can be estimated as [14],

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

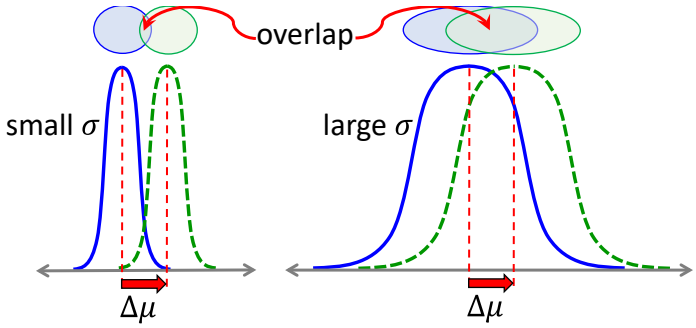

Figure 10. **K-L Divergence.** Two Gaussian distributions (solid, blue) with low variance, σ (left) and high variance, σ (right) are shown, and θ = [μ, σ]. Incrementing μ by Δμ (dashed green) results in equal change in the norm $\|\Delta\theta\|_1$ whereas $D_{KL}(\pi_\theta||\pi_{\theta+\Delta\theta})$ is higher in the distribution appearing to the left.

$$\mathbf{F}_{\boldsymbol{\theta}} \approx \nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]^{\mathrm{T}}\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]. \tag{33}$$

Recent policy gradient algorithms use concepts derived from natural gradients [14] to rectify the downside of 'vanilla' gradient descent to eliminate the use of an effective learning rate η. The use of the natural gradient greatly reduces the natural gradient algorithm's dependence on how the policy is parametrized. Unfortunately, the gains of using the natural gradient come at the cost of increased computational overheads associated with matrix inversion. The overheads may outweigh the gains when the Fisher matrix $\mathbf{F}_{\boldsymbol{\theta}}$ is very large. When the policy is represented effectively through the parameter $\boldsymbol{\theta}$ natural gradient becomes unnecessary.

Trust region policy optimization (TRPO) is a class of training algorithms that directly uses the Kullback-Leibler divergence [80]. In TRPO, a hard upper bound is imposed on this divergence produced due to the increment $\Delta\boldsymbol{\theta}$ is applied to the DNN weights $\boldsymbol{\theta}$. Denoting this bound as ε,

$$D_{KL}\left(\pi_{\boldsymbol{\theta}}||\pi_{\boldsymbol{\theta}^{\mathrm{old}}}\right) \leq \varepsilon. \tag{34}$$

Under these circumstances it can be shown that the parametric increment in TRPO is,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \left(\frac{2\varepsilon}{\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]^{\mathrm{T}}\mathbf{F}_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]}\right)^{\frac{1}{2}}\mathbf{F}_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]. \tag{35}$$

The expectation $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{E}\sim\pi_{\boldsymbol{\theta}}}[R(\mathcal{E})]$ can be estimated in the same manner as in Eqn. (30) or Eqn. (32).

Proximal policy optimization (PPO) [81] is another RL method that uses natural gradients. PPO replaces the bound in TRPO with a penalty term. An expression for the PPO's objective function for a single episode is as shown below,

$$J(\boldsymbol{\theta}) = \frac{1}{T}\sum_t \frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}^{\mathrm{old}}}(a_t|s_t)}A(s_t, a_t|\boldsymbol{\theta}) - \lambda D_{KL}\left(\pi_{\boldsymbol{\theta}}||\pi_{\boldsymbol{\theta}^{\mathrm{old}}}\right). \tag{36}$$

5.3. Off-Policy Methods

So far in this section, only on-policy algorithms have been discussed. This includes TRPO and PPO. Nevertheless, policy gradient can also be applied for off-policy learning. Since such an algorithm would be trained for the optimal policy, the samples in the replay buffer (that are collected using earlier policies) can be recycled multiple times. This aspect is a significant advantage of off-policy learning.

Policy gradient methods for off-policy learning can be implemented using the popular statistical technique of importance sampling. Suppose $f(x)$ is any function of a random variable x that follows an arbitrary distribution $q(x)$, importance sampling can be used to obtain an estimate of the expectation $\mathbb{E}_{x\sim q}[f(x)]$. Samples x are drawn from a more tractable distribution $p(x)$ and $q(x)^{-1}p(x)$ as sample weights, the weighted expec-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

tation $\mathbb{E}_{x\sim p}[p(x)^{-1}q(x)f(x)]$ is computed from several such samples. It serves as the estimated value i.e. $\mathbb{E}_{x\sim q(x)}[f(x)] \approx \mathbb{E}_{x\sim p}[p(x)^{-1}q(x)f(x)]$. This approach is adopted in off-policy RL. Suppose samples in the replay buffer are based on the policy $\pi_{\boldsymbol{\theta}}$. The gradient can be empirically estimated as,

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{a\sim p}\left[\frac{\pi_{\boldsymbol{\theta}}(a)}{p(a)} A(s,a|\boldsymbol{\theta})\right]. \tag{37}$$

5.4. Actor-Critic Networks

Actor-critic methods combine policy gradient and value-based RL methods [82]. The actor-critic architecture consists of two learning agents, the *actor* and the *critic*. From any environmental state, the actor is trained using policy gradient to respond with an action. The critic is trained with a value-based RL method to evaluate the effectiveness of the actor's output, which is then used to train the latter.

Let us denote their parameters using the symbols $\boldsymbol{\theta}^a$ and $\boldsymbol{\theta}^c$. The critic network can be modeled as a DQN, although in this explanation it is trained with the advantage function as defined in Eqn. (13). When the environmental state is $s_t(n)$, for every action $a_t(n)$ critic network provides as its output, the value of the state-action pair $q(s_t(n), a_t(n)|\boldsymbol{\theta}^c)$. The actor is incremented using gradient ascent,

$$\boldsymbol{\theta}^a \leftarrow \boldsymbol{\theta}^a + \eta^a \frac{1}{T}\sum_t q(s_t(n), a_t(n)|\boldsymbol{\theta}^c)\nabla_{\boldsymbol{\theta}^a} \log \pi_{\boldsymbol{\theta}^a}(a_t(n)|s_t(n);\boldsymbol{\theta}^a). \tag{38}$$

Eqn. (38) closely resembles the policy gradient increment as in Eqn. (30) (with $b = 0$). The only difference is that the weights applied to each gradient is determined by the critic. The update rule for the critic network, which is similar to that is Eqn. (22), is shown below,

$$\boldsymbol{\theta}^c \leftarrow \boldsymbol{\theta}^c - \eta^c \frac{1}{T}\sum_t \big(q(s_t, a_t|\boldsymbol{\theta}^c) - (r_t + q(s_{t+1}, a_{t+1}|\boldsymbol{\theta}^c)\big)\nabla_{\boldsymbol{\theta}^c} q(s_t, a_t|\boldsymbol{\theta}^c). \tag{39}$$

The *advantage actor critic* (A2C) algorithm [83] is very effective in reducing the variance in the policy gradient algorithm of the actor. The A2C architecture entails improvements over the above 'vanilla' actor critic method in two directions in the following manner.

(*i*) The actor network uses an advantage function $A(s_t, a_t)$, which is the difference between a return value R and the value of state $v(s_t|\boldsymbol{\theta}^c)$. Accordingly, the critic is trained to approximate the value function.

(*ii*) Computing the reward R uses an τ-step *lookahead* feature, where the log-gradient is weighted using the sum of the next τ rewards.

To better understand how the τ-step lookahead works, let us turn our attention to Eqn. (30). In this expression the gradient $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$ at time instant t is weighted by the factor $(R(\mathcal{E}) - b)$ where $R(\mathcal{E})$ is the return of an entire episode from $t = 0$ until $T - 1$, so that $R(\mathcal{E}) = r_0 + \gamma r_1 + \cdots \gamma^t r_t + \cdots \gamma^{t+\tau} r_{t+\tau} + \cdots \gamma^{T-1} r_{T-1}$. The baseline is b is the value of $v(s_t|\boldsymbol{\theta}^c)$. It is reasoned that the sum of the *past* rewards $r_0 + \gamma r_1 + \cdots \gamma^{t-1} r_{t-1}$ does not have any bearing on the quality of the action a_t taken at the instant t. Hence all past rewards are dropped from R. Furthermore, rewards received in the distant future, i.e. after τ instants are also dropped. In other words, R consists of the sum of the discounted rewards between the instant t and the instant $t + \tau$. Whereupon, the actor's update rule is expressed as,

$$\boldsymbol{\theta}^a \leftarrow \boldsymbol{\theta}^a + \eta^a \frac{1}{T}\sum_{t=0}^{T-\tau} \nabla_{\boldsymbol{\theta}^a} \log \pi_{\boldsymbol{\theta}^a}(a_t|s_t)\left(\sum_{t'=0}^{\tau} r_{t+t'} - v(s_t|\boldsymbol{\theta}^c)\right). \tag{40}$$

The critic is updated using the same return R, so that,

$$\boldsymbol{\theta}^c \leftarrow \boldsymbol{\theta}^c - \eta^c \frac{1}{T|\mathcal{B}|}\sum_{t=0}^{T-\tau}\left(v(s_t|\boldsymbol{\theta}^c) - \sum_{t'=0}^{\tau} r_{t+t'}\right)\nabla_{\boldsymbol{\theta}^c} v(s_t|\boldsymbol{\theta}^c). \tag{41}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

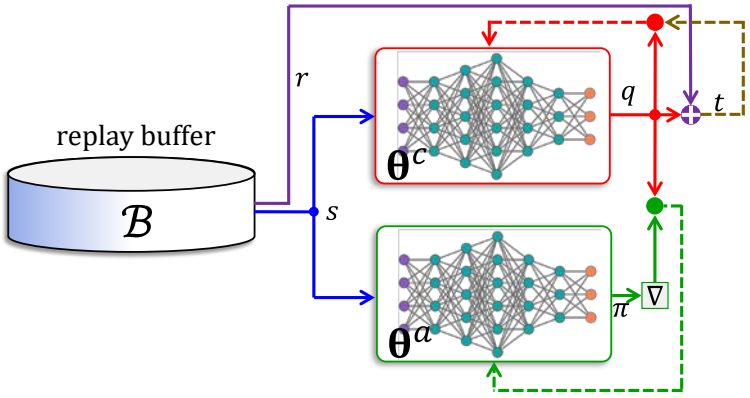

Figure 11. **Actor-Critic Network.** Shown is the overall schematic used in actor-critic learning, comprising of an actor DNN and a critic DNN.

The *asynchronous advantage actor critic* (A3C) method [83] is an extension of A2C that can be applied in parallel processing environments. A global network and a set of 'workers' are maintained in A3C. Each worker receives the actor and critic parameters that it implements on its own independent environment and collecting reward signals. The rewards are then used to determine increments $\Delta\boldsymbol{\theta}^A$ and $\Delta\boldsymbol{\theta}^C$, which are then used to asynchronously update the parameters in the global network. An advantage of A3C is that due to the parallel action of multiple workers, an experience replay buffer does not have to be incorporated.

The *deterministic policy gradient* (DPG) algorithm was described in [74], and more recently in SLH+14]. It was later extended to a deep framework in [84], known as the *deep deterministic policy gradient* (DDPG). DDPG is an off-policy actor critic method that concurrently learns the optimal Q-function q^*, as well as the optimal policy π^*.

In any off-policy actor critic model the critic must be trained to output the optimal policy π^*. Hence, the term $q(s_{t+1}, a_{t+1})$ in Eqn. (39) should be replaced with the maximum over all actions $q(s_{t+1}, a^*)$, where the optimal action $a^* = \underset{a\in\mathbb{A}}{\operatorname{argmax}}\, q(s_{t+1}, a)$ as in Eqn. (19). Unfortunately, when the action space $\mathbb{A}$ is continuous ($|\mathbb{A}| = \infty$), neither an exhaustive search to find a^* nor maximizing numerical, feasible from a computational perspective.

In order to circumvent these difficulties in identifying the optimal action that maximizes the Q-value, there three options available for use. These are outlined below.

(*i*) $q(s_{t+1}, a)$ can be sampled for several different actions and a^* be assigned the action corresponding to the sample maximum [85].

(*ii*) A convex approximation of $q(s, a)$ around s_{t+1} can be devised and a^* obtained over the approximate function [86].

(*iii*) A separate off-policy policy network can be used to learn the optimal policy π^* [87].

Out of the above three available options discussed above, the third and last has been adopted in DDPG. The critic parameter $\boldsymbol{\theta}^c$ is updated in accordance with the expression shown below,

$$\boldsymbol{\theta}^c \leftarrow \boldsymbol{\theta}^c - \eta^c \frac{1}{T|\mathcal{B}|} \sum_{n\in\mathcal{B},t} \Big(r_t + q(s_{t+1}(n), a_{t+1}(n)|\boldsymbol{\theta}^c) - q\big(s_t(n), \pi(s_t(n)|\boldsymbol{\theta}^a)\big)\Big) \nabla_{\boldsymbol{\theta}^c} q(s_t(n), a_t(n)|\boldsymbol{\theta}^c)\,. \tag{42}$$

In Eqn. (42), $\pi(s_t(n)|\boldsymbol{\theta}^a)$ is the output of DDPG's actor. DDPG uses a replay buffer $\mathcal{B}$ that includes samples from older policies. The actor's parameter $\boldsymbol{\theta}^a$ is trained using any off-policy policy gradient as in Eqn. (37).

One of the drawbacks of DDPG is the problem of overestimation [88]. If the function $q(s, a|\boldsymbol{\theta}^c)$ acquires a sharp local peak, further training converges towards the latter, leading to undesirable results. This issue has been tackled by *twin delayed deterministic policy gradient* (TD3) in [69]. TD3 maintains a pair of critics whose parameters we shall denote as $\boldsymbol{\theta}^{c_1}$ and $\boldsymbol{\theta}^{c_2}$, or more concisely as $\boldsymbol{\theta}^{c_i}$, where $i \in \{1,2\}$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

In Eqn. (42), it can be seen that DDPG has a target $r_t + q(s_{t+1}(n), a_{t+1}(n)|\boldsymbol{\theta}^c)$ where $q(s_{t+1}(n), a_{t+1}(n)|\boldsymbol{\theta}^c)$ is obtained from the critic. TD3 has two targets, $q(s_{t+1}(n), a_{t+1}(n)|\boldsymbol{\theta}^{c_i})$, $i \in \{1,2\}$. The actions $a_{t+1}(n)$ in TD3 are clipped to lie within the interval $[a_{\min}, a_{\max}]$. In order to increase exploration, Gaussian noise is added to this action. Finally the target is obtained as $r_t + \min_{i\in\{1,2\}} q(s_{t+1}(n), a_{t+1}(n)|\boldsymbol{\theta}^{c_i})$, which is used for training.

The *soft actor critic* (SAC) RL proposed recently in [70], [89] is an off-policy RL approach. The striking feature of SAC is the presence of an entropy term in the objective function,

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[r_t + \alpha H(\pi_{\boldsymbol{\theta}}|s_t)]. \quad (44)$$

Incorporating the entropy $H(\pi_{\boldsymbol{\theta}}|s_t)$ increases the degree of randomness in the policy which helps in exploration. As in TD3, SAC uses two critic networks.

6. Use of Reinforcement Learning in Home Energy Management Systems

This section aspects of the survey on the use of RL approaches for various HEMS applications. All articles in this survey are published in established journals in the past five years.

6.1. Application Classes

In this study, all applications were divided into five classes as in Figure 12 below.

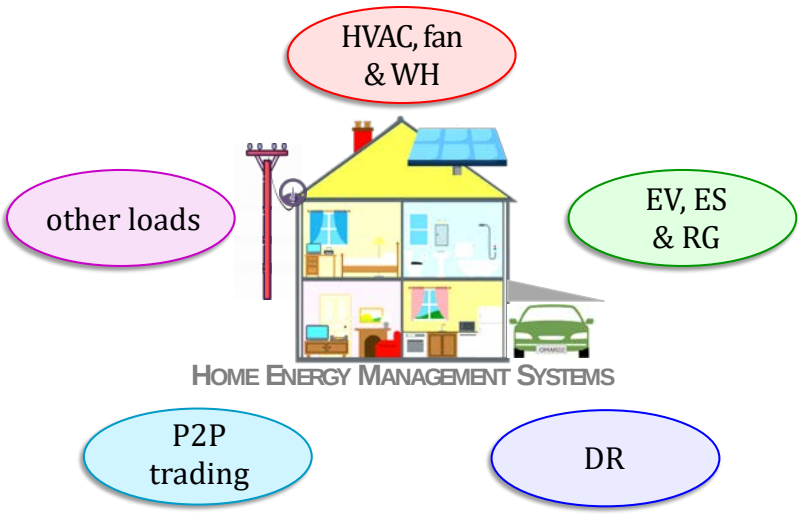

Figure 12. **HEMS Applications.** All applications of reinforcement learning in home energy management systems are classified into the five categories shown.

(*i*) *Heating, Ventilation & Air Conditioning, Fans and Water Heaters*: Heating, ventilation, and air conditioning (HVAC) systems alone are responsible for about half of the total electricity consumption [39], [90], [91], [92], [93]. In this survey, HVAC, fans and water heaters (WH) have been placed under a single category. Effective control of these loads is a major research topic in HEMS.

(*ii*) *Electric Vehicles, Energy Storage, & Renewable Generation*: The charging of electric vehicles (EVs) and energy storage (ES) devices, i.e. batteries are studied in the literature as in [94], [95]. Wherever applicable, EV and ES must be charged in coordination with renewable generation (RG) such as solar panels and wind turbines. The aim is to make decisions in order to save energy costs, while addressing comfort and other consumer requirements. Thus, EV, ES, and RG have been placed under a single class for the purpose of this survey.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

(*iii*) *Other Loads*: Suitable scheduling of several home appliances such as dishwasher, washing machine, etc. can be achieved through HEMS to save energy usage or cost. Lighting schedules are important in buildings with large occupancy. These loads have been lumped into a single class.
(*iv*) *Demand Response*: With the rapid proliferation of green energies into homes and buildings, and these sources merged into the grid, demand response (DR) has acquired much research significance in HEMS. DR programs help in load balancing, by scheduling and/or controlling shiftable loads and in incentivizing participants [96], [97] to do so through HEMS. RL for DR is one of the classes in this survey.
(*v*) *Peer-to-Peer Trading*: Home energy management has been used to maximize the profit for the prosumers by trading the electricity with each other directly in peer-to-peer (P2P) trading or indirectly through a third party as in [98]. Currently, theoretical research on automated trading is receiving significant attention. P2P trading is the last class that is considered here.

Figure 13 shows the number of research articles that applied RL to each class. Note that a significant proportion of these papers addressed more than one class. More than third of the papers we reviewed focused only on HVAC, fans and water heaters. Just above 10% of the papers studied RL control for the energy storage (ES) systems. Only 7% of the papers focused on the energy trading. However, most of the papers (46%) are targeting more than one object. These results are shown in Figure 13.

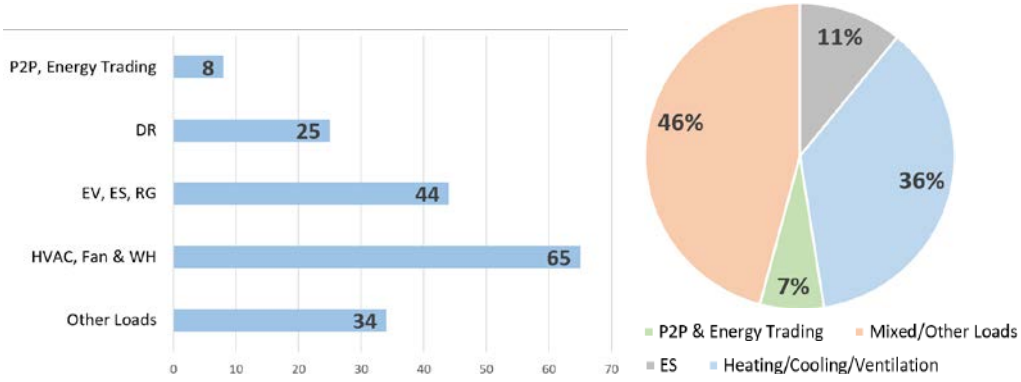

Figure 13. **Application Classes.** The total number of articles in each application class (left), as well as their corresponding proportions (right).

6.2. Objectives and Building Types

Within these HEMS applications, RL has been applied in several ways. It has been used to reduce energy consumption within residential units and buildings [99]. It has also been used to achieve a higher comfort level of occupants [100]. In operations at the interface between the residential units and the energy grid, RL has been applied to maximize prosumers profit in energy trading as well as for load balancing. For this purpose, we break down the objectives into three different types as listed below.
(*i*) *Energy Cost*: The cost of using any electrical device by the consumer and in most of the cases it is proportionally related to energy consumption. In this paper we use cost and consumption interchangeably.
(*ii*) *Occupant Comfort*: The main factor that can affect the occupant's comfort is the thermal comfort, which depends mainly on the room temperature and humidity.
(*iii*) *Load Balance*: Power supply companies try to achieve load balance by reducing the power consumption of consumers at peak periods to match the station power supply. The consumers are motivated to participate in such programs by price incentives.

All buildings were categorized into the following three types.
(*i*) *Residential*: For the purpose of this survey, individual homes, residential communities, as well as apartment complexes fall under this type of buildings.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

(*ii*) *Commercial*: These buildings include offices, office complexes, shops, malls, hotels, as well as industrial buildings.
(*iii*) *Academic*: Academic buildings range from schools, university classrooms, buildings, research laboratories, up to entire campuses.

In residential buildings, the research literature revealed that RL was applied in all five application classes. However, in case of commercial and academic buildings, RL was typically applied to the first three categories, i.e. to HVAC, fans & WH, to EVs, ESs & RGs, as well as to other loads.

Figure 14 illustrates the outcome of this survey. It may be noted that in the largest proportion of articles (44%) the RL algorithm took into account both cost and comfort. About a quarter of the remaining addressed cost alone forming the second largest proportion.

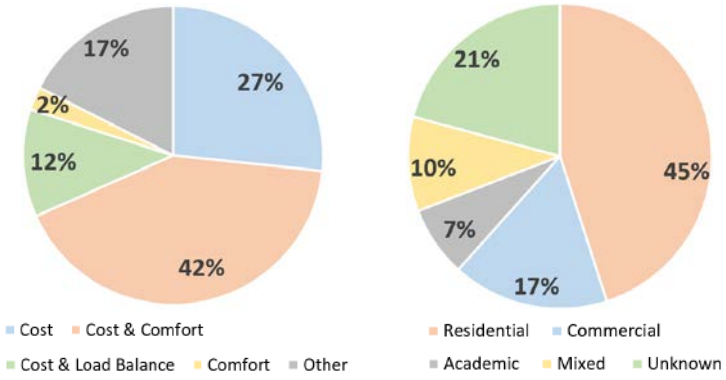

Figure 14. **Objectives and Building Type.** Proportion of articles in each objective (left) and building type (right).

6.3. Deployment, Multi-Agents, and States/Actions

The proportion of research articles where RL was actually deployed in the real world was studied. It was found that only 12% of research articles report results where RL was used with real HEMS. The results are consistent with an earlier survey [40] where this proportion was 11%. The results are shown in Fig. 15.

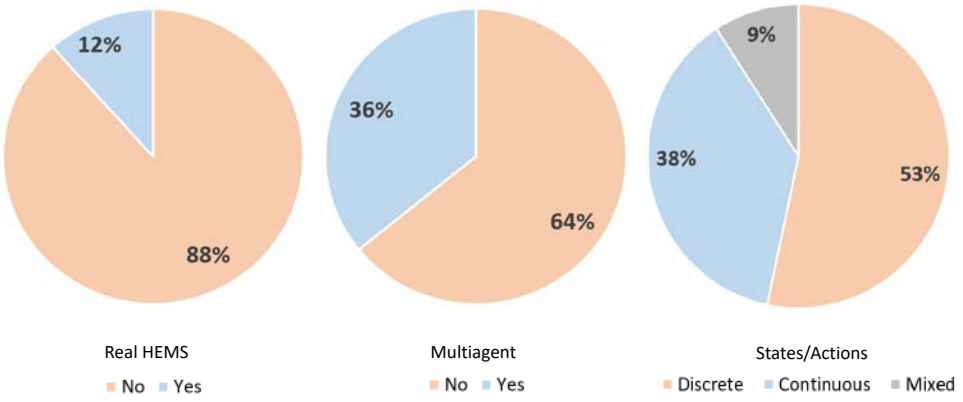

Figure 15. **Real-World, Multi-Agents, and Discretization.** Proportion of articles deployed in real world HEMS (left), using multi-agents (middle), and whether states/actions are discrete or continuous (right).

7. Reinforcement Learning Algorithms in Home Energy Management Systems

This section focuses on how the RL and DRL algorithms described in earlier sections were used in HEMS applications. Table 1 provides a list of references that used tabular

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

RL. The references have been categorized in terms of the application class, objective function, and building type that were described in the immediately preceding section.

Table 1. References using Tabular Reinforcement Learning.

Reference	Application	Objective	Building Type	Algorithm
[101]	HVAC, Fans, WH	Cost	Residential	Q-Learning
[102]		Cost & Comfort		
[103], [104]		Other	Academic	
[105]		Comfort	Mixed/NA	
[106]		Other		
[107], [98]	P2P Trading	Cost		
[108], [109]			Residential	
[110]	EV, ES, and RG			
[111], [112]			Mixed/NA	
[113]		Other	Residential	
[114], [115]	Other/Mixed	Cost & Comfort	Commercial	
[116]			Academic	
[117], [96], [118], [119], [120], [121]			Residential	
[122]		Other		
[123], [124]		Cost		
[125]			Mixed/NA	
[126]		Cost & Comfort		
[127], [128]		Cost & Load Balance		
[129]		Other		
[130]	P2P Trading	Cost		Distributed RL
[131], [132], [133]	HVAC, Fans, WH	Cost & Comfort	Residential	Other (Fitted Q-Iter.)
[134]		Comfort	Commercial	Q-Learn. & SARSA
[135]		Cost & Comfort	Residential	SARSA
[136]	Other/Mixed	Cost & Load Balance		Policy Learning
[137]		Other		
[138]		Cost & Comfort	Commercial	Model Based RL
[139]	HVAC, Fans, WH	Cost	Residential	Other (CARLA)
[140]		Cost & Comfort	Commercial	Other (Context. RL)

In a similar manner, Table 3 considers those using deep policy learning. Most algorithms used DQN. However, DDQN was also popular among HEMS researchers. The survey found only one article using the Dueling DQN.

Table 4 provides a list of references that applied actor-critic learning. It shows that PPO is more popular than TRPO. We believe that this is due to the more recency of the latter algorithm.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

References that used either a combination of two or more approaches, or any other approach not common in RL literature, are shown in Table 5.

Table 2. References using Deep Q Networks.

Reference	Application	Objective	Building Type	Algorithm
[141], [142]	Other/Mixed	Cost	Residential	DQN
[143]		Cost & Load Balance		
[94]	EV, ES, and RG	Cost		
[144]		Other		
[145]		Cost & Comfort		
[146]	HVAC, Fans, WH	Cost		
[147]	Other/Mixed		Commercial	
[148]		Cost & Comfort		
[149], [150]	HVAC, Fans, WH		Mixed/NA	
[151], [152]	Other/Mixed	Cost		
[153], [154], [155]	HVAC, Fans, WH	Cost & Comfort	Residential	DDQN
[156]			Academic	
[157]		Comfort	Commercial	
[158]	Other/Mixed	Cost & Load Balance	Residential	
[95]		Cost & Comfort		Dueling DQN
[159]	HVAC, Fans, WH	Cost		Other (FQI-LSTM, FQI-CNN)

Table 3. References using Deep Policy Networks.

Reference	Application	Objective	Building Type	Algorithm
[160]	HVAC, Fans, WH	Cost & Comfort	Academic	PPO
[161]			Commercial	
[162]	P2P Trading	Other	Mixed/NA	
[163]	EV, ES, and RG			
[164]	Other/Mixed	Cost		
[165]		Cost & Comfort	Residential	TRPO

Table 4. References using Actor-Critic Networks

Reference	Application	Objective	Building Type	Algorithm
[166], [167]	HVAC, Fans, WH	Cost & Comfort	Residential	DDPG
[51], [168], [169], [170]	Other/Mixed			
[171], [172]		Cost & Load Balance		
[173]		Cost		
[174]	EV, ES, and RG			
[175]	Other/Mixed	Cost & Comfort	Academic	
[176]		Other		
[177], [178]	EV, ES, and RG		Commercial	

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 August 2022 doi:10.20944/preprints202208.0104.v1

[179], [180], [181]	HVAC, Fans, WH	Cost & Comfort	Mixed/NA	
[182], [183], [184]	EV, ES, and RG	Other		
[185], [186]	Other/Mixed	Cost & Load Balance	Residential	SAC
[187], [188]	HVAC, Fans, WH	Cost	Commercial	
[189], [190], [191], [92]		Cost & Comfort		
[192]	Other/Mixed			
[193]			Academic	
[194], [195], [196]	HVAC, Fans, WH	Cost & Load Balance	Mixed/NA	
[197], [198], [199]		Cost & Comfort		
[200]	Other/Mixed		Residential	A2C
[201]	HVAC, Fans, WH	Cost	Commercial	A3C
[202]	P2P Trading		Mixed/NA	TD3
[203]	HVAC, Fans, WH			
[204]		Cost & Comfort		
[205]	Other/Mixed		Residential	

Table 5. References using a combination of RL or miscellaneous methods.

Reference	Application	Objective	Building Type	Algorithm
[50]	Other/Mixed	Cost & Comfort	Residential	DQN, DDPG
[206]				DQN, DDQN
[207]		Cost & Load Balance		DQN, DPG
[208]	P2P Trading			Other (Model-Based DRL)
[209]	HVAC, Fans, WH	Cost & Comfort	Academic	SAC, TD3, TRPO, PPO
[210]			Mixed/NA	Other (Clustering DRL)
[211]	EV, ES, and RG			PPO, TD3
[212]		Cost & Load Balance	Commercial	DDPG, DDQN, DQN

References

1. U.S. Energy Information Administration, “Electricity explained: use of electricity,” May 14, 2021. Available: www.eia.gov/energyexplained/electricity/use-of-electricity.php [Accessed: April 10, 2022].
2. Center for Sustainable Systems, “U.S. Energy System Factsheet,” Pub. No. CSS03-11, Center for Sustainable Systems, University of Michigan, September 2021. Available: https://css.umich.edu/factsheets/us-energy-system-factsheet [Accessed: April 10, 2022].
3. Mohammad Shakeri, Mohsen Shayestegan, Hamza Abunima, S.M. Salim Reza, M. Akhtaruzzaman, A.R.M. Alamoud, Kamaruzzaman Sopian, Nowshad Amin, "An intelligent system architecture in home energy management systems (HEMS) for efficient demand response in smart grid," *Energy and Buildings*, Volume 138, 2017, pp. 154-164, ISSN 0378-7788, doi.org/10.1016/j.enbuild.2016.12.026.
4. J. Leitão, P. Gil, B. Ribeiro, and A. Cardoso, “A survey on home energy management,” *IEEE Access*, vol. 8, pp. 5699–5722, 2020.

5. H. Shareef, M. S. Ahmed, A. Mohamed and E. Al Hassan, "Review on Home Energy Management System Considering Demand Responses, Smart Technologies, and Intelligent Controllers," in *IEEE Access*, vol. 6, pp. 24498-24509, 2018, doi: 10.1109/ACCESS.2018.2831917.
6. Bandana Mahapatra, and Anand Nayyar, "Home energy management system (HEMS): Concept, architecture, infrastructure, challenges and energy management schemes," *Energy Systems*, pp. 1-27, 2019. doi.org/10.1007/s12667-019-00364-w.
7. G. Dileep, "A survey on smart grid technologies and applications," *Renewable energy*, vol. 146, pp. 2589-2625, 2020.
8. U. Zafar, S. Bayhan and A. Sanfilippo, "Home Energy Management System Concepts, Configurations, and Technologies for the Smart Grid," in *IEEE Access*, vol. 8, pp. 119271-119286, 2020, doi: 10.1109/ACCESS.2020.3005244.
9. Kari Alanne, Seppo Sierla, "An overview of machine learning applications for smart buildings," *Sustainable Cities and Society*, Volume 76, 2022, 103445, ISSN 2210-6707, https://doi.org/10.1016/j.scs.2021.103445.
10. J. Aguilar, A. Garces-Jimenez, M.D. R-Moreno, Rodrigo García, "A systematic literature review on the use of artificial intelligence in energy self-management in smart buildings," *Renewable and Sustainable Energy Reviews*, Volume 151, 2021, 111530, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2021.111530.
11. Yassine Himeur, Khalida Ghanem, Abdullah Alsalemi, Faycal Bensaali, Abbes Amira, "Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives," *Applied Energy*, vol. 287, 2021, 116601, doi.org/10.1016/j.apenergy.2021.116601.
12. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol.13, 1983, pp. 835-846.
13. G. Tesauro, "TD-Gammon, a self-teaching backgammon program, achieves master-level play," *Neural Computation*, vol. 6, no. 2, 1994, 215–219.
14. Jan Peters, and Stefan Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682-697, 2008.
15. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.
16. D. Silver *et al*., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
17. D. Silver *et al*., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
18. Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, Anil Anthony Bharath, "A Brief Survey of Deep Reinforcement Learning," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
19. Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare and Joelle Pineau, "An Introduction to Deep Reinforcement Learning", *Foundations and Trends in Machine Learning*, vol. 11, no. 3-4, 2018. DOI: 10.1561/2200000071.
20. David Silver, Satinder Singh, Doina Precup, Richard S. Sutton, "Reward Is Enough," *Artificial Intelligence*, vol. 299, 2021: 103535.
21. Ben Goertzel, *Artificial general intelligence* (Eds: Cassio Pennachin), vol. 2, Springer, New York, 2007.
22. Tengteng Zhang, and Hongwei Mo, "Reinforcement learning for robot research: A comprehensive review and open issues," *International Journal of Advanced Robotic Systems*, vol. 18, no. 3, 2021.
23. Sarthak Bhagat, Hritwick Banerjee, Zion Tsz Ho Tse, and Hongliang Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges," *Robotics*, vol, 8, no. 1, 2019.
24. H. Shakhatreh et al., "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," in *IEEE Access*, vol. 7, pp. 48572-48634, 2019, doi: 10.1109/ACCESS.2019.2909530.
25. F. Zeng, C. Wang and S. S. Ge, "A Survey on Visual Navigation for Artificial Agents With Deep Reinforcement Learning," in *IEEE Access*, vol. 8, pp. 135426-135442, 2020, doi: 10.1109/ACCESS.2020.3011438.
26. H. Sun, W. Zhang, R. Yu and Y. Zhang, "Motion Planning for Mobile Robots-Focusing on Deep Reinforcement Learning: A Systematic Review," in *IEEE Access*, vol. 9, pp. 69061-69081, 2021, doi: 10.1109/ACCESS.2021.3076530.
27. N. C. Luong et al., "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133-3174, 4th quarter 2019, doi: 10.1109/COMST.2019.2916583.
28. Z. Ullah, F. Al-Turjman and L. Mostarda, "Cognition in UAV-Aided 5G and Beyond Communications: A Survey," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 3, pp. 872-891, Sept. 2020, doi: 10.1109/TCCN.2020.2968311.
29. T. T. Nguyen, and V. J. Reddi, "Deep reinforcement learning for cyber security," *arXiv preprint*, arXiv:1906.05799, 2019.
30. H. Zhu, Y. Cao, W. Wang, T. Jiang and S. Jin, "Deep Reinforcement Learning for Mobile Edge Caching: Review, New Features, and Open Issues," in *IEEE Network*, vol. 32, no. 6, pp. 50-57, November/December 2018, doi: 10.1109/MNET.2018.1800109.
31. Siqi Liu, Kay Choong See, Kee Yuan Ngiam, Leo Anthony Celi, Xingzhi Sun, and Mengling Feng, "Reinforcement learning for clinical decision support in critical care: comprehensive review," *Journal of Medical Internet Research*, vol. 22, no. 7, 2020.
32. D. Elavarasan, and P. M. D. Vincent, "Crop Yield Prediction Using Deep Reinforcement Learning Model for Sustainable Agrarian Applications," in *IEEE Access*, vol. 8, pp. 2020, 86886-86901, doi: 10.1109/ACCESS.2020.2992480.
33. Paul Garnier, Jonathan Viquerat, Jean Rabault, Aurélien Larcher, Alexander Kuhnle, and Elie Hachem, "A review on deep reinforcement learning for fluid mechanics," *Computers & Fluids*, vol. 225, 2021.

34. D. Zhang, X. Han and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," in *CSEE Journal of Power and Energy Systems*, vol. 4, no. 3, pp. 362-370, September 2018, doi: 10.17775/CSEE-JPES.2018.00520.
35. Z. Zhang, D. Zhang and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," in *CSEE Journal of Power and Energy Systems*, vol. 6, no. 1, pp. 213-225, March 2020, doi: 10.17775/CSEEJPES.2019.00920.
36. O. Jogunola et al., "Consensus Algorithms and Deep Reinforcement Learning in Energy Market: A Review," in *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4211-4227, 15 March15, 2021, doi: 10.1109/JIOT.2020.3032162.
37. A.T.D. Perera, Parameswaran Kamalaruban, "Applications of reinforcement learning in energy systems," *Renewable and Sustainable Energy Reviews*, Volume 137, 2021, 110618, ISSN 1364-0321, doi.org/10.1016/j.rser.2020.110618.
38. Xin Chen, Guannan Qu, Yujie Tang, Steven Low, and Na Li, "Reinforcement learning for selective key applications in power systems: Recent advances and future challenges," *IEEE Transactions on Smart Grid*, 2022, doi: 10.1109/TSG.2022.3154718.
39. Karl Mason, and Santiago Grijalva, "A review of reinforcement learning for autonomous building energy management," *Computers & Electrical Engineering*, vol. 78, pp. 300-312, 2019.
40. Zhe Wang, Tianzhen Hong, "Reinforcement learning for building controls: The opportunities and challenges," *Applied Energy*, Vol. 269, 2020, pp. 115036, , doi.org/10.1016/j.apenergy.2020.115036. 1997-2019.
41. Mengjie Han, Ross May, Xingxing Zhang, Xinru Wang, Song Pan, Da Yan, Yuan Jin, and Liguo Xu, "A review of reinforcement learning methodologies for controlling occupant comfort in buildings," *Sustainable Cities and Society*, vol. 51, pp. 101748–101762, Nov. 2019. doi.org/10.1016/j.scs.2019.101748.
42. L. Yu, S. Qin, M. Zhang, C. Shen, T. Jiang and X. Guan, "A Review of Deep Reinforcement Learning for Smart Building Energy Management," in *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12046-12063, 1 Aug.1, 2021, doi: 10.1109/JIOT.2021.3078462.
43. H. Zhang, S. Seal, D. Wu, F. Bouffard and B. Boulet, "Building Energy Management With Reinforcement Learning and Model Predictive Control: A Survey," *IEEE Access*, vol. 10, pp. 27853-27862, 2022, doi: 10.1109/ACCESS.2022.3156581.
44. José R.Vázquez-Canteli, and Zoltán Nagy, "Reinforcement learning for demand response: A review of algorithms and modeling techniques," *Applied Energy*, vol. 235, pp. 1072–1089, Feb. 2019. https://doi.org/10.1016/j.apenergy.2018.11.002.
45. Hammou Ou Ali, M. Ouassaid, Mohamed Maaroufi, "Chapter 24: Optimal appliance management system with renewable energy integration for smart homes," *Renewable Energy Systems*, Academic Press, 2021, pp. 533-552, https://doi.org/10.1016/B978-0-12-820004-9.00025-5.
46. Swati Sharda, Mukhtiar Singh, Kapil Sharma, "Demand side management through load shifting in IoT based HEMS: Overview, challenges and opportunities," *Sustainable Cities and Society*, Volume 65, 2021, 102517, ISSN 2210-6707, https://doi.org/10.1016/j.scs.2020.102517.
47. C. Withanage, R. Ashok, C. Yuen and K. Otto, "A comparison of the popular home automation technologies," *IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA)*, 2014, pp. 600-605, doi: 10.1109/ISGT-Asia.2014.6873860.
48. G. Van de Kaa, S. Stoccuto, C. Valladolid Calderón, "A battle over smart standards: Compatibility, governance, and innovation in home energy management systems and smart meters in the Netherlands," *Energy Research & Social Science*, vol. 82, 2021, 102302, ISSN 2214-6296, https://doi.org/10.1016/j.erss.2021.102302.
49. Batchu Rajasekhar, Wayes Tushar, Clement Lork, Yuren Zhou, Chau Yuen, Naran M. Pindoriya, and Kristin L. Wood, "A survey of computational intelligence techniques for air-conditioners energy management," *IEEE Transactions on Emerging Topics Computational Intelligence,* vol. 4, no. 4, pp. 555–570, Aug. 2020.
50. C. Huang, H. Zhang, L. Wang, X. Luo and Y. Song, "Mixed Deep Reinforcement Learning Considering Discrete-Continuous Hybrid Action Space for Smart Home Energy Management," *Journal of Modern Power Systems and Clean Energy*, doi: 10.35833/MPCE.2021.000394.
51. L. Yu et al., "Deep Reinforcement Learning for Smart Home Energy Management," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2751-2762, April 2020, doi: 10.1109/JIOT.2019.2957289.
52. S. Das, "Deep Neural Networks," YouTube, Jan. 31, 2022 [Video file]. Available: https://www.youtube.com/playlist?list=PL_4Jjqx0pZY-SIO8jElzW0lNpzjcunOx4 [Accessed: April 1, 2022].
53. I. Goodfellow, Y. Bengio and A. Courville, "Deep Learning," MIT Press, 2016. Available: www.deeplearningbook.org.
54. Open AI, "Part 2: Kinds of RL Algorithms" 2018. Available: spinningup.openai.com/en/latest/spinningup/rl_intro2.html.
55. R. Bellman, *Dynamic Programming*, Princeton, 1957.
56. R. Bellman, "A Markovian Decision Process", *Journal of Mathematics and Mechanics*, vol. 6, no. 5, 1957, pp. 679–84, ISSN: 00959057.
57. R. Howard, *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.
58. Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang, "A theoretical analysis of deep Q-learning," In *Learning for Dynamics and Control*, PMLR, 2020, pp. 486-489.
59. R. S. Sutton, and A. G. Barto, *Reinforcement Learning: An Introduction*, Bradford Books, MIT Press, Cambridge, MA, 1998, revised 2018. Available: web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf
60. C. J. C. H. Watkins, *Learning from Delayed Rewards*. PhD Thesis, University of Cambridge, UK, 1989.
61. Gavin A. Rummery, and Mahesan Niranjan, "On-line Q-learning using connectionist systems," *Technical Report*: *Department of Engineering*, *University of Cambridge*, vol. 37, 1994.

62. R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3–4), pp. 229-256, 1992.
63. M. Riedmiller, "Neural fitted Q iteration-first experiences with a data efficient neural reinforcement learning method," in *European Conference on Machine Learning*, Springer, pp. 317–328, 2005.
64. L. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3, pp. 293-321, 1992.
65. Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver, "Prioritized experience replay," *arXiv preprint*, arXiv:1511.05952, 2015.
66. H. Hasselt, "Double Q-learning," *Advances in Neural Information Processing Systems*, vol. 23, pp. 2613-2621, 2010.
67. Andreas Pentaliotis, "Investigating Overestimation Bias in Reinforcement Learning," M.S. thesis, University of Groningen, Univ., Melbourne, 2020 [Online]. Available: https://www.ai.rug.nl/~mwiering/Thesis-Andreas-Pentaliotis.pdf [Accessed: April 1, 2022].
68. Hado van Hasselt, Arthur Guez, and David Silver, "Deep reinforcement learning with double Q learning," In *Proceedings of the* 30th *AAAI Conference on Artificial Intelligence*, vol. 30, no. 1. 2016.
69. Scott Fujimoto, Herke Hoof, and David Meger, "Addressing function approximation error in actor-critic methods," In *International Conference on Machine Learning*, PMLR, pp. 1587-1596, 2018.
70. Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International Conference on Machine Learning*, PMLR, pp. 1861-1870, 2018.
71. Haobo Jiang, Jin Xie, and Jian Yang, "Action Candidate Driven Clipped Double Q-learning for Discrete and Continuous Action Tasks," *arXiv preprint*, arXiv:2203.11526, 2022.
72. Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," in 33rd *International Conference on Machine Learning*, *PMLR*, vol 48, pp. 1995-2003, 2016.
73. Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, pp. 1057–1063, 2000.
74. Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, pp. 1057-1063, 1999.
75. Kamil Ciosek, and Shimon Whiteson, "Expected policy gradients for reinforcement learning," *arXiv preprint*, arXiv:1801.03326, 2018.
76. Philip S. Thomas, and Emma Brunskill, "Policy gradient methods for reinforcement learning with function approximation and action-dependent baselines," *arXiv preprint*, arXiv:1706.06643, 2017.
77. Lex Weaver, and Nigel Tao, "The optimal reward baseline for gradient-based reinforcement learning," in *Proceedings*, 17th *Conference on Uncertainty in Artificial Intelligence*, pp. 538-545, 2001.
78. Sueli I. R. Costa, Sandra A. Santos, and João E. Strapasson, "Fisher information distance: A geometrical reading," *Discrete Applied Mathematics*, vol. 197, pp. 59-69, 2015.
79. S. Kakade, "A Natural Policy Gradient," in *Advances in Neural Information Processing Systems*, vol. 14, 2001.
80. John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz, "Trust region policy optimization," *Proceedings of the* 32nd *International Conference on Machine Learning*, *PMLR*, vol. 37, pp.1889-1897, 2015.
81. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal policy optimization algorithms," *arXiv preprint* arXiv:1707.06347, 2017.
82. Vijay R. Konda, and John N. Tsitsiklis, "On actor-critic algorithms," *SIAM Journal on Control and Optimization*, Vol. 42, no. 4, 2003, pp. 1143-1166.
83. Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," In *International Conference on Machine Learning*, PMLR, pp. 1928-1937, 2016.
84. Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint*, arXiv:1509.02971v6, 2017.
85. D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," In *Conference on Robot Learning*, *PMLR*, pp. 651-673, October 2018.
86. Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas, "Sample efficient actor-critic with experience replay." *arXiv preprint*, arXiv:1611.01224, 2016.
87. David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller, "Deterministic policy gradient algorithms," In *International Conference on Machine Learning*, PMLR, pp. 387-395, 2014.
88. Lingheng Meng, Rob Gorbet, and Dana Kulić, "The effect of multi-step methods on overestimation in deep reinforcement learning," In 25th *International Conference on Pattern Recognition* (ICPR), pp. 347-353, 2021.

89. Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine, "Soft actor-critic algorithms and applications," *arXiv preprint*, arXiv:1812.05905, 2018.
90. Mohammad Esrafilian-Najafabadi, Fariborz Haghighat, "Occupancy-based HVAC control systems in buildings: A state-of-the-art review," *Building and Environment*, Vol. 197, 2021, 107810, ISSN 0360-1323, doi.org/10.1016/j.buildenv.2021.107810.
91. Lizhi Jia, Shen Wei, Junjie Liu, "A review of optimization approaches for controlling water-cooled central cooling systems," *Building and Environment*, Volume 203, 2021, 108100, ISSN 0360-1323, https://doi.org/10.1016/j.buildenv.2021.108100.
92. L. Yu et al., "Multi-Agent Deep Reinforcement Learning for HVAC Control in Commercial Buildings," *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 407-419, Jan. 2021, doi: 10.1109/TSG.2020.3011739.
93. Sarah Noye, Rubén Mulero Martinez, Laura Carnieletto, Michele De Carli, Amaia Castelruiz Aguirre, "A review of advanced ground source heat pump control: Artificial intelligence for autonomous and adaptive control," *Renewable and Sustainable Energy Reviews*, Volume 153, 2022, 111685, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2021.111685.
94. A. Paraskevas, D. Aletras, A. Chrysopoulos, A. Marinopoulos, and D. I. Doukas, "Optimal Management for EV Charging Stations: A Win–Win Strategy for Different Stakeholders Using Constrained Deep Q-Learning," *Energies*, vol. 15, no. 7, p. 2323, Mar. 2022, doi: 10.3390/en15072323.
95. Mifeng Ren, Xiangfei Liu, Zhile Yang, Jianhua Zhang, Yuanjun Guo, Yanbing Jia, "A novel forecasting based scheduling method for household energy management system based on deep reinforcement learning," *Sustainable Cities and Society*, Volume 76, 2022, 103207, ISSN 2210-6707, https://doi.org/10.1016/j.scs.2021.103207.
96. F. Alfaverh, M. Denaï and Y. Sun, "Demand Response Strategy Based on Reinforcement Learning and Fuzzy Reasoning for Home Energy Management," *IEEE Access*, vol. 8, pp. 39310-39321, 2020, doi: 10.1109/ACCESS.2020.2974286.
97. Ioannis Antonopoulos, Valentin Robu, Benoit Couraud, Desen Kirli, Sonam Norbu, Aristides Kiprakis, David Flynn, Sergio Elizondo-Gonzalez, Steve Wattam, "Artificial intelligence and machine learning approaches to energy demand-side response: A systematic review," *Renewable and Sustainable Energy Reviews*, Volume 130, 2020, 109899, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2020.109899.
98. T. Chen and W. Su, "Indirect Customer-to-Customer Energy Trading With Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 4338-4348, July 2019, doi: 10.1109/TSG.2018.2857449.
99. Mathieu Bourdeau, Xiao qiang Zhai, Elyes Nefzaoui, Xiaofeng Guo, Patrice Chatellier, "Modeling and forecasting building energy consumption: A review of data-driven techniques," *Sustainable Cities and Society*, Volume 48, 2019, 101533, ISSN 2210-6707, https://doi.org/10.1016/j.scs.2019.101533.
100. Nan Ma, Dorit Aviv, Hongshan Guo, William W. Braham, "Measuring the right factors: A review of variables and models for thermal comfort and indoor air quality," *Renewable and Sustainable Energy Reviews*, Volume 135, 2021, 110436, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2020.110436.
101. J. Xu, H. Mahmood, H. Xiao, E. Anderlini and M. Abusara, "Electric Water Heaters Management via Reinforcement Learning With Time-Delay in Isolated Microgrids," *IEEE Access*, vol. 9, pp. 132569-132579, 2021, doi: 10.1109/ACCESS.2021.3112817.
102. Clement Lork, Wen-Tai Li, Yan Qin, Yuren Zhou, Chau Yuen, Wayes Tushar, Tapan K. Saha, "An uncertainty-aware deep reinforcement learning framework for residential air conditioning energy management," *Applied Energy*, Volume 276, 2020, 115426, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.115426.
103. Camila Correa-Jullian, Enrique López Droguett, José Miguel Cardemil, "Operation scheduling in a solar thermal system: A reinforcement learning-based framework," *Applied Energy*, Volume 268, 2020, 114943, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.114943.
104. J. Hao, D. W. Gao and J. J. Zhang, "Reinforcement Learning for Building Energy Optimization Through Controlling of Central HVAC System," *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 320-328, 2020, doi: 10.1109/OAJPE.2020.3023916.
105. Siliang Lu, Weilong Wang, Chaochao Lin, Erica Cochran Hameen, "Data-driven simulation of a thermal comfort-based temperature set-point control with ASHRAE RP884," *Building and Environment*, Volume 156, 2019, Pages 137-146, ISSN 0360-1323, https://doi.org/10.1016/j.buildenv.2019.03.010.
106. M. Liu, S. Peeters, D. S. Callaway and B. J. Claessens, "Trajectory Tracking With an Aggregation of Domestic Hot Water Heaters: Combining Model-Based and Model-Free Control in a Commercial Deployment," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5686-5695, Sept. 2019, doi: 10.1109/TSG.2018.2890275.
107. M. R. Bin Mohamad Saifuddin, T. Logenthiran, R. T. Naayagi and W. L. Woo, "A Nano-Biased Energy Management Using Reinforced Learning Multi-Agent on Layered Coalition Model: Consumer Sovereignty," *IEEE Access*, vol. 7, pp. 52542-52564, 2019, doi: 10.1109/ACCESS.2019.2911543.
108. S. Zhou, Z. Hu, W. Gu, M. Jiang and X. Zhang, "Artificial intelligence based smart energy community management: A reinforcement learning approach," *CSEE Journal of Power and Energy Systems*, vol. 5, no. 1, pp. 1-10, March 2019, doi: 10.17775/CSEEJPES.2018.00840.
109. K. Ojand and H. Dagdougui, "Q-Learning-Based Model Predictive Control for Energy Management in Residential Aggregator," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 70-81, Jan. 2022, doi: 10.1109/TASE.2021.3091334.

110. Y. Wang, X. Lin and M. Pedram, "A Near-Optimal Model-Based Control Algorithm for Households Equipped With Residential Photovoltaic Power Generation and Energy Storage Systems," *IEEE Transactions on Sustainable Energy*, vol. 7, no. 1, pp. 77-86, Jan. 2016, doi: 10.1109/TSTE.2015.2467190.
111. S. Kim and H. Lim, "Reinforcement Learning Based Energy Management Algorithm for Smart Energy Buildings," *Energies*, vol. 11, no. 8, p. 2010, Aug. 2018, doi: 10.3390/en11082010.
112. Yuwei Shang, Wenchuan Wu, Jianbo Guo, Zhao Ma, Wanxing Sheng, Zhe Lv, Chenran Fu, "Stochastic dispatch of energy storage in microgrids: An augmented reinforcement learning approach," *Applied Energy*, Volume 261, 2020, 114423, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2019.114423.
113. P. Kofinas, A.I. Dounis, G.A. Vouros, "Fuzzy Q-Learning for multi-agent decentralized energy management in microgrids," *Applied Energy*, Volume 219, 2018, pp. 53-67, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2018.03.017.
114. June Young Park, Thomas Dougherty, Hagen Fritz, Zoltan Nagy, "LightLearn: An adaptive and occupant centered controller for lighting based on reinforcement learning," *Building and Environment*, Volume 147, 2019, Pages 397-414, ISSN 0360-1323, https://doi.org/10.1016/j.buildenv.2018.10.028.
115. P. Korkidis, A. Dounis, and P. Kofinas, "Computational Intelligence Technologies for Occupancy Estimation and Comfort Control in Buildings," *Energies*, vol. 14, no. 16, pp. 4971, Aug. 2021, doi: 10.3390/en14164971.
116. Xiongfeng Zhang, Renzhi Lu, Junhui Jiang, Seung Ho Hong, Won Seok Song, "Testbed implementation of reinforcement learning-based demand response energy management system," *Applied Energy*, Volume 297, 2021, 117131, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.117131.
117. R. Lu, S. H. Hong and M. Yu, "Demand Response for Home Energy Management Using Reinforcement Learning and Artificial Neural Network," *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6629-6639, Nov. 2019, doi: 10.1109/TSG.2019.2909266.
118. T. Remani, E. A. Jasmin and T. P. I. Ahamed, "Residential Load Scheduling With Renewable Generation in the Smart Grid: A Reinforcement Learning Approach," *IEEE Systems Journal*, vol. 13, no. 3, pp. 3283-3294, Sept. 2019, doi: 10.1109/JSYST.2018.2855689.
119. M. Khan, J. Seo and D. Kim, "Real-Time Scheduling of Operational Time for Smart Home Appliances Based on Reinforcement Learning," *IEEE Access*, vol. 8, pp. 116520-116534, 2020, doi: 10.1109/ACCESS.2020.3004151.
120. M. Ahrarinouri, M. Rastegar and A. R. Seifi, "Multiagent Reinforcement Learning for Energy Management in Residential Buildings," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 659-666, Jan. 2021, doi: 10.1109/TII.2020.2977104.
121. S. -J. Chen, W. -Y. Chiu and W. -J. Liu, "User Preference-Based Demand Response for Smart Home Energy Management Using Multiobjective Reinforcement Learning," *IEEE Access*, vol. 9, pp. 161627-161637, 2021, doi: 10.1109/ACCESS.2021.3132962.
122. X. Xu, Y. Jia, Y. Xu, Z. Xu, S. Chai and C. S. Lai, "A Multi-Agent Reinforcement Learning-Based Data-Driven Method for Home Energy Management," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3201-3211, July 2020, doi: 10.1109/TSG.2020.2971427.
123. X. Fang, J. Wang, G. Song, Y. Han, Q. Zhao, and Z. Cao, "Multi-Agent Reinforcement Learning Approach for Residential Microgrid Energy Scheduling," *Energies*, vol. 13, no. 1, pp. 123, Dec. 2019, doi: 10.3390/en13010123.
124. Y. Wan, J. Qin, X. Yu, T. Yang and Y. Kang, "Price-Based Residential Demand Response Management in Smart Grids: A Reinforcement Learning-Based Approach," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 1, pp. 123-134, January 2022, doi: 10.1109/JAS.2021.1004287.
125. Renzhi Lu, Seung Ho Hong, Xiongfeng Zhang, "A Dynamic pricing demand response algorithm for smart grid: Reinforcement learning approach," *Applied Energy*, Volume 220, 2018, pp. 220-230, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2018.03.072.
126. Z. Wen, D. O'Neill and H. Maei, "Optimal Demand Response Using Device-Based Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2312-2324, Sept. 2015, doi: 10.1109/TSG.2015.2396993.
127. Renzhi Lu, Seung Ho Hong, "Incentive-based demand response for smart grid with reinforcement learning and deep neural network," *Applied Energy*, Volume 236, 2019, pp. 937-949, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2018.12.061.
128. Xiangyu Kong, Deqian Kong, Jingtao Yao, Linquan Bai, Jie Xiao, "Online pricing of demand response based on long short-term memory and reinforcement learning," *Applied Energy*, vol. 271, 2020, 114945, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.114945.
129. L. A. Hurtado, E. Mocanu, P. H. Nguyen, M. Gibescu and R. I. G. Kamphuis, "Enabling Cooperative Behavior for Building Demand Response Based on Extended Joint Action Learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 127-136, Jan. 2018, doi: 10.1109/TII.2017.2753408.
130. D. Barth, B. Cohen-Boulakia, and W. Ehounou, "Distributed Reinforcement Learning for the Management of a Smart Grid Interconnecting Independent Prosumers," *Energies*, vol. 15, no. 4, p. 1440, Feb. 2022, doi: 10.3390/en15041440.
131. F. Ruelens, S. Iacovella, B. Claessens, and R. Belmans, "Learning Agent for a Heat-Pump Thermostat with a Set-Back Strategy Using Model-Free Reinforcement Learning," *Energies*, vol. 8, no. 8, pp. 8300–8318, Aug. 2015, doi: 10.3390/en8088300.
132. F. Ruelens, B. J. Claessens, S. Vandael, B. De Schutter, R. Babuška and R. Belmans, "Residential Demand Response of Thermostatically Controlled Loads Using Batch Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2149-2159, Sept. 2017, doi: 10.1109/TSG.2016.2517211.

133. F. Ruelens, B. J. Claessens, S. Quaiyum, B. De Schutter, R. Babuška and R. Belmans, "Reinforcement Learning Applied to an Electric Water Heater: From Theory to Practice," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3792-3800, July 2018, doi: 10.1109/TSG.2016.2640184.
134. Mengjie Han, Ross May, Xingxing Zhang, Xinru Wang, Song Pan, Yan Da, Yuan Jin, "A novel reinforcement learning method for improving occupant comfort via window opening and closing," *Sustainable Cities and Society*, vol. 61, 2020, 102247, ISSN 2210-6707, https://doi.org/10.1016/j.scs.2020.102247.
135. Hussain Kazmi, Johan Suykens, Attila Balint, Johan Driesen, "Multi-agent reinforcement learning for modeling and control of thermostatically controlled loads," *Applied Energy*, Volume 238, 2019, pp. 1022-1035, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2019.01.140.
136. S. Xu et al., "Agent-based modeling and simulation for the electricity market with residential demand response," *CSEE Journal of Power and Energy Systems*, vol. 7, no. 2, pp. 368-380, March 2021, doi: 10.17775/CSEEJPES.2019.01750.
137. S. S. Reka, P. Venugopal, H. H. Alhelou, P. Siano and M. E. H. Golshan, "Real Time Demand Response Modeling for Residential Consumers in Smart Grid Considering Renewable Energy With Deep Learning Approach," *IEEE Access*, vol. 9, pp. 56551-56562, 2021, doi: 10.1109/ACCESS.2021.3071993.
138. G. Kontes et al., "Simulation-Based Evaluation and Optimization of Control Strategies in Buildings," *Energies*, vol. 11, no. 12, pp. 3376, Dec. 2018, doi: 10.3390/en11123376.
139. Q. Jia, S. Chen, Z. Yan and Y. Li, "Optimal Incentive Strategy in Cloud-Edge Integrated Demand Response Framework for Residential Air Conditioning Loads," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 31-42, 1 Jan.-March 2022, doi: 10.1109/TCC.2021.3118597.
140. P. Macieira, L. Gomes, and Z. Vale, "Energy Management Model for HVAC Control Supported by Reinforcement Learning," *Energies*, vol. 14, no. 24, pp. 8210, Dec. 2021, doi: 10.3390/en14248210.
141. José R. Vázquez-Canteli, Stepan Ulyanin, Jérôme Kämpf, Zoltán Nagy, "Fusing TensorFlow with building energy simulation for intelligent energy management in smart cities," *Sustainable Cities and Society*, Volume 45, 2019, pp. 243-257, ISSN 2210-6707,https://doi.org/10.1016/j.scs.2018.11.021.
142. T. Zhou and M. Lin, "Deadline-Aware Deep-Recurrent-Q-Network Governor for Smart Energy Saving," *IEEE Transactions on Network Science and Engineering*, doi: 10.1109/TNSE.2021.3123280.
143. B. J. Claessens, P. Vrancx and F. Ruelens, "Convolutional Neural Networks for Automatic State-Time Feature Extraction in Reinforcement Learning Applied to Residential Load Control," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3259-3269, July 2018, doi: 10.1109/TSG.2016.2629450.
144. Felix Tuchnitz, Niklas Ebell, Jonas Schlund, Marco Pruckner, "Development and Evaluation of a Smart Charging Strategy for an Electric Vehicle Fleet Based on Reinforcement Learning," *Applied Energy*, vol. 285, 2021, 116382, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.116382.
145. A. Tittaferrante and A. Yassine, "Multi-Advisor Reinforcement Learning for Multi-Agent Multi-Objective Smart Home Energy Control," *IEEE Transactions on Artificial Intelligence*, doi: 10.1109/TAI.2021.3125918.
146. Shengyuan Zhong, Xiaoyuan Wang, Jun Zhao, Wenjia Li, Hao Li, Yongzhen Wang, Shuai Deng, Jiebei Zhu, "Deep reinforcement learning framework for dynamic pricing demand response of regenerative electric heating," *Applied Energy*, vol. 288, 2021, 116623,ISSN 0306-2619,https://doi.org/10.1016/j.apenergy.2021.116623.
147. P. Wei, S. Xia, R. Chen, J. Qian, C. Li and X. Jiang, "A Deep-Reinforcement-Learning-Based Recommender System for Occupant-Driven Energy Optimization in Commercial Buildings," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6402-6413, July 2020, doi: 10.1109/JIOT.2020.2974848.
148. Z. Liang et al., "Safe Reinforcement Learning-Based Resilient Proactive Scheduling for a Commercial Building Considering Correlated Demand Response," *IEEE Open Access Journal of Power and Energy*, vol. 8, pp. 85-96, 2021, doi: 10.1109/OAJPE.2021.3064319.
149. Xiangtian Deng, Yi Zhang, Yi Zhang, He Qi, "Towards optimal HVAC control in non-stationary building environments combining active change detection and deep reinforcement learning," *Building and Environment*, vol. 211, 2022, 108680, ISSN 0360-1323, https://doi.org/10.1016/j.buildenv.2021.108680.
150. T. Wei, S. Ren and Q. Zhu, "Deep Reinforcement Learning for Joint Datacenter and HVAC Load Control in Distributed Mixed-Use Buildings," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 3, pp. 370-384, 1 July-Sept. 2021, doi: 10.1109/TSUSC.2019.2910533.
151. T. Chen and W. Su, "Local Energy Trading Behavior Modeling With Deep Reinforcement Learning," *IEEE Access*, vol. 6, pp. 62806-62814, 2018, doi: 10.1109/ACCESS.2018.2876652.
152. P. Suanpang, P. Jamjuntr, K. Jermsittiparsert, P. Kaewyong, "Autonomous Energy Management by Applying Deep Q-Learning to Enhance Sustainability in Smart Tourism Cities," *Energies*, vol. 15, p. 1906, 2022, doi: 10.3390/en15051906.
153. C. Blad, S. Bøgh, and C. Kallesøe, "A Multi-Agent Reinforcement Learning Approach to Price and Comfort Optimization in HVAC-Systems," *Energies*, vol. 14, no. 22, pp. 7491, Nov. 2021, doi: 10.3390/en14227491.
154. Ting Yang, Liyuan Zhao, Wei Li, Jianzhong Wu, Albert Y. Zomaya,"Towards healthy and cost-effective indoor environment management in smart homes: A deep reinforcement learning approach," *Applied Energy*, vol. 300, 2021, 117335, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.117335.

155. Amirreza Heidari, François Maréchal, Dolaana Khovalyg, "An occupant-centric control framework for balancing comfort, energy use and hygiene in hot water systems: A model-free reinforcement learning approach, *Applied Energy*," vol. 312, 2022, 118833, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2022.118833.
156. William Valladares, Marco Galindo, Jorge Gutiérrez, Wu-Chieh Wu, Kuo-Kai Liao, Jen-Chung Liao, Kuang-Chin Lu, Chi-Chuan Wang, "Energy optimization associated with thermal comfort and indoor air control via a deep reinforcement learning algorithm," *Building and Environment*, vol. 155, 2019, pp. 105-117, ISSN 0360-1323, https://doi.org/10.1016/j.buildenv.2019.03.038.
157. Alex Dmitrewski, Miguel Molina-Solana, Rossella Arcucci, "CntrlDA: A building energy management control system with real-time adjustments. Application to indoor temperature," *Building and Environment*, vol. 215, 2022, 108938, ISSN 0360-1323, https://doi.org/10.1016/j.buildenv.2022.108938.
158. Alwyn Mathew, Milan Jeetendra Jolly, Jimson Mathew, "Improved residential energy management system using priority double deep Q-learning," *Sustainable Cities and Society*, vol. 69, 2021, 102812, ISSN 2210-6707, https://doi.org/10.1016/j.scs.2021.102812.
159. F. Ruelens, B. J. Claessens, P. Vrancx, F. Spiessens and G. Deconinck, "Direct load control of thermostatically controlled loads based on sparse observations using deep reinforcement learning," *CSEE Journal of Power and Energy Systems*, vol. 5, no. 4, pp. 423-432, Dec. 2019, doi: 10.17775/CSEEJPES.2019.00590.
160. Y. Chemingui, A. Gastli, and O. Ellabban, "Reinforcement Learning-Based School Energy Management System," *Energies*, vol. 13, no. 23, pp. 6354, Dec. 2020, doi: 10.3390/en13236354.
161. X. Zhang et al., "Two-Stage Reinforcement Learning Policy Search for Grid-Interactive Building Control," *IEEE Transactions on Smart Grid*, doi: 10.1109/TSG.2022.3141625.
162. L. Yang, Q. Sun, N. Zhang and Y. Li, "Indirect Multi-energy Transactions of Energy Internet with Deep Reinforcement Learning Approach," *IEEE Transactions on Power Systems*, doi: 10.1109/TPWRS.2022.3142969.
163. Chenyu Guo, Xin Wang, Yihui Zheng, Feng Zhang, "Real-time optimal energy management of microgrid with uncertainties based on deep reinforcement learning," *Energy*, vol. 238, Part C, 2022, 121873, ISSN 0360-5442, https://doi.org/10.1016/j.energy.2021.121873.
164. Seunghoon Jung, Jaewon Jeoung, Hyuna Kang, Taehoon Hong, "Optimal planning of a rooftop PV system using GIS-based reinforcement learning," *Applied Energy*, vol. 298, 2021, 117239, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.117239.
165. H. Li, Z. Wan and H. He, "Real-Time Residential Demand Response," *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4144-4154, Sept. 2020, doi: 10.1109/TSG.2020.2978061.
166. G. Gao, J. Li and Y. Wen, "DeepComfort: Energy-Efficient Thermal Comfort Control in Buildings Via Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8472-8484, Sept. 2020, doi: 10.1109/JIOT.2020.2992117.
167. Yan Du, Helia Zandi, Olivera Kotevska, Kuldeep Kurte, Jeffery Munk, Kadir Amasyali, Evan Mckee, Fangxing Li, "Intelligent multi-zone residential HVAC control strategy based on deep reinforcement learning," *Applied Energy*, vol. 281, 2021, 116117, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.116117.
168. N. Kodama, T. Harada and K. Miyazaki, "Home Energy Management Algorithm Based on Deep Reinforcement Learning Using Multistep Prediction," *IEEE Access*, vol. 9, pp. 153108-153115, 2021, doi: 10.1109/ACCESS.2021.3126365.
169. B. Svetozarevic, C. Baumann, S. Muntwiler, L. Di Natale, M.N. Zeilinger, P. Heer, "Data-driven control of room temperature and bidirectional EV charging using deep reinforcement learning: Simulations and experiments," *Applied Energy*, vol. 307, 2022, 118127, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.118127.
170. I. Zenginis, J. Vardakas, N. E. Koltsaklis and C. Verikoukis, "Smart Home's Energy Management through a Clustering-based Reinforcement Learning Approach," *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2022.3152586.
171. H. -M. Chung, S. Maharjan, Y. Zhang and F. Eliassen, "Distributed Deep Reinforcement Learning for Intelligent Load Scheduling in Residential Smart Grids," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2752-2763, April 2021, doi: 10.1109/TII.2020.3007167.
172. Dawei Qiu, Yujian Ye, Dimitrios Papadaskalopoulos, Goran Strbac, "Scalable coordinated management of peer-to-peer energy trading: A multi-cluster deep reinforcement learning approach," *Applied Energy*, vol. 292, 2021, 116940, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.116940.
173. Y. Ye, D. Qiu, X. Wu, G. Strbac and J. Ward, "Model-Free Real-Time Autonomous Control for a Residential Multi-Energy System Using Deep Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3068-3082, July 2020, doi: 10.1109/TSG.2020.2976771.
174. W. Li, M. Tang, X. Zhang, D. Gao, and J. Wang, "Operation of Distributed Battery Considering Demand Response Using Deep Reinforcement Learning in Grid Edge Control," *Energies*, vol. 14, no. 22, p. 7749, Nov. 2021, doi: 10.3390/en14227749.
175. Samir Touzani, Anand Krishnan Prakash, Zhe Wang, Shreya Agarwal, Marco Pritoni, Mariam Kiran, Richard Brown, Jessica Granderson, "Controlling distributed energy resources via deep reinforcement learning for load flexibility and energy efficiency," *Applied Energy*, vol. 304, 2021, 117733, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.117733.
176. Xinlei Zhou, Wenye Lin, Ritunesh Kumar, Ping Cui, Zhenjun Ma, "A data-driven strategy using long short term memory models and reinforcement learning to predict building electricity consumption," *Applied Energy*, Volume 306, Part B, 2022, 118078, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.118078.

177. Renzhi Lu, Yi-Chang Li, Yuting Li, Junhui Jiang, Yuemin Ding, "Multi-agent deep reinforcement learning based demand response for discrete manufacturing systems energy management," *Applied Energy*, vol. 276, 2020, 115473, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.115473.
178. L. Desportes, I. Fijalkow, and P. Andry, "Deep Reinforcement Learning for Hybrid Energy Storage Systems: Balancing Lead and Hydrogen Storage," *Energies*, vol. 14, no. 15, p. 4706, Aug. 2021, doi: 10.3390/en14154706.
179. Zhengbo Zou, Xinran Yu, Semiha Ergan, "Towards optimal control of air handling units using deep reinforcement learning and recurrent neural network," *Building and Environment*, vol. 168, 2020, 106535, ISSN 0360-1323, https://doi.org/10.1016/j.buildenv.2019.106535.
180. B. Liu, M. Akcakaya and T. E. Mcdermott, "Automated Control of Transactive HVACs in Energy Distribution Systems," *IEEE Transactions on Smart Grid*, vol. 12, no. 3, pp. 2462-2471, May 2021, doi: 10.1109/TSG.2020.3042498.
181. J. Li, W. Zhang, G. Gao, Y. Wen, G. Jin and G. Christopoulos, "Toward Intelligent Multizone Thermal Control With Multiagent Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11150-11162, 15 July15, 2021, doi: 10.1109/JIOT.2021.3051400.
182. Y. Miao, T. Chen, S. Bu, H. Liang, and Z. Han, "Co-Optimizing Battery Storage for Energy Arbitrage and Frequency Regulation in Real-Time Markets Using Deep Reinforcement Learning," *Energies*, vol. 14, no. 24, p. 8365, Dec. 2021, doi: 10.3390/en14248365.
183. Y. Du and D. Wu, "Deep Reinforcement Learning from Demonstrations to Assist Service Restoration in Islanded Microgrids," *IEEE Transactions on Sustainable Energy*, doi: 10.1109/TSTE.2022.3148236.
184. Dawei Qiu, Zihang Dong, Xi Zhang, Yi Wang, Goran Strbac, "Safe reinforcement learning for real-time automatic control in a smart energy-hub," *Applied Energy*, vol. 309, 2022, 118403, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.118403.
185. S. Bahrami, Y. C. Chen and V. W. S. Wong, "Deep Reinforcement Learning for Demand Response in Distribution Networks," *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1496-1506, March 2021, doi: 10.1109/TSG.2020.3037066.
186. Y. Ye, Y. Tang, H. Wang, X. -P. Zhang and G. Strbac, "A Scalable Privacy-Preserving Multi-Agent Deep Reinforcement Learning Approach for Large-Scale Peer-to-Peer Transactive Energy Trading," *IEEE Transactions on Smart Grid*, vol. 12, no. 6, pp. 5185-5200, Nov. 2021, doi: 10.1109/TSG.2021.3103917.
187. D. Deltetto, D. Coraci, G. Pinto, M. S. Piscitelli, and A. Capozzoli, "Exploring the Potentialities of Deep Reinforcement Learning for Incentive-Based Demand Response in a Cluster of Small Commercial Buildings," *Energies*, vol. 14, no. 10, p. 2933, May 2021, doi: 10.3390/en14102933.
188. Silvio Brandi, Massimo Fiorentini, Alfonso Capozzoli, "Comparison of online and offline deep reinforcement learning with model predictive control for thermal energy management," *Automation in Construction*, vol. 135, 2022, 104128, ISSN 0926-5805, https://doi.org/10.1016/j.autcon.2022.104128.
189. W. Hu, Y. Wen, K. Guan, G. Jin and K. J. Tseng, "iTCM: Toward Learning-Based Thermal Comfort Modeling via Pervasive Sensing for Smart Buildings," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4164-4177, Oct. 2018, doi: 10.1109/JIOT.2018.2861831.
190. D. Coraci, S. Brandi, M. S. Piscitelli, and A. Capozzoli, "Online Implementation of a Soft Actor-Critic Agent to Enhance Indoor Temperature Control and Energy Efficiency in Buildings," *Energies*, vol. 14, no. 4, p. 997, Feb. 2021, doi: 10.3390/en14040997.
191. H. Zhao, B. Wang, H. Liu, H. Sun, Z. Pan and Q. Guo, "Exploiting the Flexibility Inside Park-Level Commercial Buildings Considering Heat Transfer Time Delay: A Memory-Augmented Deep Reinforcement Learning Approach," *IEEE Transactions on Sustainable Energy*, vol. 13, no. 1, pp. 207-219, Jan. 2022, doi: 10.1109/TSTE.2021.3107439.
192. Dafeng Zhu, Bo Yang, Yuxiang Liu, Zhaojian Wang, Kai Ma, Xinping Guan, "Energy management based on multi-agent deep reinforcement learning for a multi-energy industrial park," *Applied Energy*, vol. 311, 2022, 118636, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2022.118636.
193. Yude Qin, Ji Ke, Biao Wang, Gennady Fedorovich Filaretov, "Energy optimization for regional buildings based on distributed reinforcement learning," *Sustainable Cities and Society*, Volume 78, 2022, 103625, ISSN 2210-6707, https://doi.org/10.1016/j.scs.2021.103625.
194. Giuseppe Pinto, Davide Deltetto, Alfonso Capozzoli, "Data-driven district energy management with surrogate models and deep reinforcement learning," *Applied Energy*, vol. 304, 2021, 117642, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.117642.
195. Giuseppe Pinto, Marco Savino Piscitelli, José Ramón Vázquez-Canteli, Zoltán Nagy, Alfonso Capozzoli, "Coordinated energy management for a cluster of buildings through deep reinforcement learning," *Energy*, vol. 229, 2021, 120725, ISSN 0360-5442, https://doi.org/10.1016/j.energy.2021.120725.
196. Giuseppe Pinto, Anjukan Kathirgamanathan, Eleni Mangina, Donal P. Finn, Alfonso Capozzoli, "Enhancing energy management in grid-interactive buildings: A comparison among cooperative and coordinated architectures," *Applied Energy*, vol. 310, 2022, 118497, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.118497.
197. Z. Zhang, C. Ma, and R. Zhu, "Thermal and Energy Management Based on Bimodal Airflow-Temperature Sensing and Reinforcement Learning," *Energies*, vol. 11, no. 10, pp. 2575, Sep. 2018, doi: 10.3390/en11102575.
198. Ashkan Haji Hosseinloo, Alexander Ryzhov, Aldo Bischi, Henni Ouerdane, Konstantin Turitsyn, Munther A. Dahleh, "Data-driven control of micro-climate in buildings: An event-triggered reinforcement learning approach," *Applied Energy*, vol. 277, 2020, 115451, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.115451.

199. V. Taboga, A. Bellahsen and H. Dagdougui, "An Enhanced Adaptivity of Reinforcement Learning-Based Temperature Control in Buildings Using Generalized Training," *IEEE Transactions on Emerging Topics in Computational Intelligence*, doi: 10.1109/TETCI.2021.3066999.
200. S. Lee and D. -H. Choi, "Federated Reinforcement Learning for Energy Management of Multiple Smart Homes With Distributed Energy Resources," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 488-497, Jan. 2022, doi: 10.1109/TII.2020.3035451.
201. X. Zhang, D. Biagioni, M. Cai, P. Graf and S. Rahman, "An Edge-Cloud Integrated Solution for Buildings Demand Response Using Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 420-431, Jan. 2021, doi: 10.1109/TSG.2020.3014055.
202. T. Chen, S. Bu, X. Liu, J. Kang, F. R. Yu and Z. Han, "Peer-to-Peer Energy Trading and Energy Conversion in Interconnected Multi-Energy Microgrids Using Multi-Agent Deep Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 13, no. 1, pp. 715-727, Jan. 2022, doi: 10.1109/TSG.2021.3124465.
203. J. H. Woo, L. Wu, J. -B. Park and J. H. Roh, "Real-Time Optimal Power Flow Using Twin Delayed Deep Deterministic Policy Gradient Algorithm," *IEEE Access*, vol. 8, pp. 213611-213618, 2020, doi: 10.1109/ACCESS.2020.3041007.
204. C. Fu and Y. Zhang, "Research and Application of Predictive Control Method Based on Deep Reinforcement Learning for HVAC Systems," *IEEE Access*, vol. 9, pp. 130845-130852, 2021, doi: 10.1109/ACCESS.2021.3114161.
205. Y. Ye, D. Qiu, H. Wang, Y. Tang, and G. Strbac, "Real-Time Autonomous Residential Demand Response Management Based on Twin Delayed Deep Deterministic Policy Gradient Learning," *Energies*, vol. 14, no. 3, p. 531, Jan. 2021, doi: 10.3390/en14030531.
206. Y. Liu, D. Zhang and H. B. Gooi, "Optimization strategy based on deep reinforcement learning for home energy management," *CSEE Journal of Power and Energy Systems*, vol. 6, no. 3, pp. 572-582, Sept. 2020, doi: 10.17775/CSEEJPES.2019.02890.
207. E. Mocanu et al., "On-Line Building Energy Optimization Using Deep Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3698-3708, July 2019, doi: 10.1109/TSG.2018.2834219.
208. H. Shuai and H. He, "Online Scheduling of a Residential Microgrid via Monte-Carlo Tree Search and a Learned Model," *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1073-1087, March 2021, doi: 10.1109/TSG.2020.3035127.
209. Marco Biemann, Fabian Scheller, Xiufeng Liu, Lizhen Huang, "Experimental evaluation of model-free reinforcement learning algorithms for continuous HVAC control," *Applied Energy*, vol. 298, 2021, 117164, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.117164.
210. Raad Z. Homod, Hussein Togun, Ahmed Kadhim Hussein, Fadhel Noraldeen Al-Mousawi, Zaher Mundher Yaseen, Wael Al-Kouz, Haider J. Abd, Omer A. Alawi, Marjan Goodarzi, Omar A. Hussein, "Dynamics analysis of a novel hybrid deep clustering for unsupervised learning by reinforcement of multi-agent to energy saving in intelligent buildings," *Applied Energy*, vol. 313, 2022, 118863, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2022.118863.
211. Glenn Ceusters, Román Cantú Rodríguez, Alberte Bouso García, Rüdiger Franke, Geert Deconinck, Lieve Helsen, Ann Nowé, Maarten Messagie, Luis Ramirez Camargo, "Model-predictive control and reinforcement learning in multi-energy system case studies," *Applied Energy*, vol. 303, 2021, 117634, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.117634.
212. Marina Dorokhova, Yann Martinson, Christophe Ballif, Nicolas Wyrsch, "Deep reinforcement learning control of electric vehicle charging in the presence of photovoltaic generation," *Applied Energy*, vol. 301, 2021, 117504, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2021.117504.