INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2026, VOL. 72, NO. 1, PP. 1–10
Manuscript received July 30, 2025; revised February, 2026. doi: 10.24425/ijet.2026.157871

Performance analysis of tabular and Fuzzy Q-learning under varying state and action space resolution

Roman Zajdel

***Abstract*—Reinforcement learning (RL) algorithms, such as Q-learning, are widely applied to control tasks involving continuous state spaces that require discretization or function approximation. However, the effect of state and action space resolution on learning efficiency and convergence stability remains a significant challenge, particularly when comparing classical tabular approaches with fuzzy function approximations. This study presents an in-depth experimental analysis of Q(0)-learning and trace-based Q(λ)-learning, applied to three benchmark control problems: Cart–Pole, Ball–Beam, and Mountain Car. The experiments systematically investigate how increasing the granularity of state discretization (number of bins), the number of fuzzy sets, and the size of the action space influence convergence speed and result variance. The results clearly demonstrate that Q(λ)-learning consistently outperforms Q(0)-learning in both tabular and fuzzy settings, providing faster convergence and greater stability at higher discretization resolutions. Furthermore, fuzzy Q(λ)-learning exhibits superior scalability and generalization capabilities, particularly for complex underactuated systems such as Ball–Beam. These findings highlight the practical advantages of combining eligibility traces with fuzzy state representation in reinforcement learning. This approach supports the design of more robust controllers for real-world dynamic systems.**

***Keywords*—Reinforcement Learning, Q-learning, Fuzzy Systems, State Discretization**

I. Introduction

REINFORCEMENT Learning (RL) provides a general framework for sequential decision-making in dynamic and uncertain environments [1]. Among various RL algorithms, Q-learning remains one of the most widely studied methods due to its simplicity and proven convergence under tabular representations [2]. However, practical applications involving continuous or high-dimensional state–action spaces require discretization or function approximation, both of which significantly affect learning performance and scalability [3], [4].

Tabular Q-learning assumes a finite number of states and actions, which makes it highly sensitive to the resolution of state discretization. Increasing this resolution often leads to the "curse of dimensionality," slower convergence, and poor generalization. To address these limitations, various extensions have been proposed, such as eligibility traces (resulting in Q(λ)-learning) [1], [5], and function approximation techniques including fuzzy logic systems [6], [7]. These hybrid approaches aim to improve generalization and stability when learning in continuous spaces.

The discretization of the action space plays a crucial role in reinforcement learning (RL), especially in environments with continuous or high-dimensional state and action spaces. Several studies have examined how the number of available discrete actions affects the learning process. For instance, action representation choices significantly impact both the convergence speed and the final policy quality [8]. A finer discretization may increase policy expressiveness but often leads to slower learning and increased exploration complexity [9]. Adaptive discretization methods have been proposed to balance these trade-offs by dynamically adjusting action resolution during training [10]. In fuzzy reinforcement learning, the number of fuzzy sets—and hence the effective number of action rules—also influences convergence and generalization [11]. More recently, parameterized action spaces have been proposed to blend discrete and continuous representations, further emphasizing the importance of action granularity in modern RL algorithms [12].

Motivated by this gap, the present study provides a comprehensive experimental comparison of classical tabular and fuzzy Q-learning, with and without eligibility traces, under different discretization and action space configurations. The goal is to analyze the impact of increasing state–action space resolution on learning performance, stability, and scalability, thereby offering practical insights for the development of more robust RL controllers for real-world dynamic systems.

The remainder of this article is structured as follows. Section II reviews prior research on fuzzy reinforcement learning and the use of eligibility traces. Section III-A describes the considered control environments, followed by Section III-B and Section III-C, which present the core RL algorithms and the discretization and fuzzification strategies applied to state representations. Section III-D outlines the fuzzy Q-learning framework based on zero-order Takagi–Sugeno models. The experimental setup and results are detailed in Section IV, with key observations summarized in Section V. Section VI discusses the implications of the findings, and Section VII concludes the paper with final remarks and directions for future work.

R. Zajdel is with Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, Rzeszow, Poland (e-mail: rzajdel@prz.edu.pl).

II. Related Work

Recent advancements in reinforcement learning (RL) have continued to explore fuzzy and hybrid extensions of the classic Q-learning algorithm to enhance learning efficiency, robustness, and interpretability in complex control scenarios. Among these, fuzzy systems—particularly of the Takagi–Sugeno (T–S) type—have been widely adopted to approximate the action-value function $Q(s, a)$. A common modeling strategy involves fixing the antecedent (membership) functions and adapting only the consequent (typically linear or affine) functions, resulting in compact and interpretable function approximators.

For instance, Jouffe [13] introduced fuzzy Q-learning based on fixed fuzzy partitions and linear consequents trained via temporal-difference (TD) methods. This principle has since been extended in various domains: Glorennec and Jouffe [14] applied it to robot navigation, Wen et al. [15] used a T–S fuzzy controller with parallel distributed compensation (PDC) for trajectory planning in humanoid manipulators, and Zander et al. [16] designed T–S-based actor–critic architectures, demonstrating performance on par with deep Q-networks (DQN) while preserving interpretability.

More recently, Zahmatkesh et al. [17] proposed a robust fuzzy Q-learning algorithm for nonlinear flight control, showing resilience against wind disturbances during autonomous landings. Parallel work by Hostetter et al. [18] introduced self-organizing fuzzy Q-networks that automatically construct rule bases while maintaining competitiveness with offline RL baselines. Zander et al. [19] also confirmed that fuzzy DQN can outperform standard DQN in interpretability-focused benchmarks. Additionally, Kozuno et al. [5] revisited Peng's Q(λ) and demonstrated that eligibility traces can improve convergence and temporal credit assignment even when integrated with modern function approximators.

Building on these developments, Tang et al. [20] proposed a fuzzy actor–critic framework tailored for dynamic load balancing in smart grids, demonstrating improved energy efficiency and stability. Similarly, Wang et al. [21] designed a fuzzy deep reinforcement learning controller for autonomous vehicle navigation, emphasizing rule interpretability. In 2023, Lei and Lin [22] proposed a hierarchical fuzzy reinforcement learning architecture for multi-task control, highlighting scalability and cross-task generalization.

Collectively, these studies underscore the versatility of fuzzy Q-learning—particularly with fixed antecedents and learned consequents—as a robust, scalable, and interpretable alternative to conventional RL.

While these studies confirm the practical advantages of fuzzy and trace-based extensions, they primarily focus on the overall feasibility or hybrid design. Systematic analyses that examine how the resolution of fuzzy partitions and the number of discrete actions jointly affect convergence speed, result variance, and scalability remain scarce. Such aspects are crucial when designing controllers for benchmark environments with varying state dimensionality, underactuation, and sparse rewards, such as Cart–Pole, Ball–Beam, or Mountain Car.

III. Methods

A. System Dynamics

Three benchmark systems were considered: the ball-beam system, the cart-pole swing-up system, and the mountain car. These classic nonlinear tasks are standard benchmarks for reinforcement learning algorithms [1], [2], [4].

a) Ball-Beam.: The state variables of the ball–beam system (Fig. 1) are the position of the ball, x, and its velocity, $\dot{x}$ [23]. The discrete-time dynamics of the system are given by:

$$\begin{aligned} x_t &= x_{t-1} + \tau \dot{x}_{t-1}, \\ \dot{x}_t &= \dot{x}_{t-1} + \tau g \sin(\phi_t), \end{aligned} \tag{1}$$

where $\tau = 0.02\,\mathrm{s}$ is the discretization step and $g = 9.81\,\mathrm{m/s^2}$ is the gravitational constant. Control over the ball's position is achieved by varying the tilt angle ϕ of the beam relative to the horizontal plane. The objective is to balance the ball at the center of the beam; nevertheless, in reinforcement learning scenarios, a satisfactory outcome is often defined as maintaining the ball on the beam for a given period of time. In the discrete-action implementation of the ball–beam environment, a three-action set is typically used—tilt left, keep the beam level, or tilt right—which in practical simulation code corresponds to actions such as $\{-\pi/4,\ 0,\ \pi/4\}$ (e.g. in `ballbeam-gym`, with `action_mode=discrete`) [24].

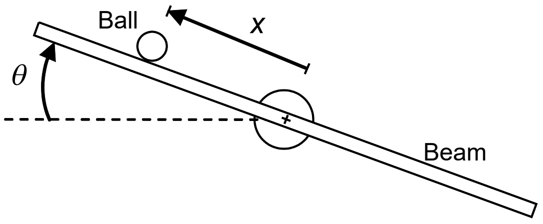

Fig. 1. Schematic of the Ball-Beam system.

b) Cart-Pole.: The inverted pendulum consists of a rigid rod of length 1 m mounted on a horizontally moving cart. The pendulum is constrained to move only within the plane of the figure, while the cart can travel up to 2.4 m to the left or right from its central position (see Fig.2). The cart-pole system is governed by non-linear equations, neglecting friction coefficients for both the cart and the pendulum [25]:

$$\begin{aligned} x_t &= x_{t-1} + \tau \dot{x}_{t-1}, \\ \dot{x}_t &= \dot{x}_{t-1} + \tau \frac{F_t + ml\left(\dot{\theta}_{t-1}^2 \sin(\theta_{t-1}) - \ddot{\theta}_{t-1}\cos(\theta_{t-1})\right)}{m_c + m}, \\ \theta_t &= \theta_{t-1} + \tau \dot{\theta}_{t-1}, \\ \dot{\theta}_t &= \dot{\theta}_{t-1} + \tau \frac{g\sin(\theta_{t-1}) - \cos(\theta_{t-1}) \cdot \frac{F_t + ml\dot{\theta}_{t-1}^2 \sin(\theta_{t-1})}{m_c + m}}{l\left(\frac{4}{3} - \frac{m\cos^2(\theta_{t-1})}{m_c + m}\right)} \end{aligned} \tag{2}$$

where the state vector consists of the following variables: x (cart position), $\dot{x}$ (cart velocity), θ (pendulum angle from the vertical), and $\dot{\theta}$ (angular velocity of the pendulum). The gravitational constant is $g = 9.81\,\mathrm{m/s^2}$, the cart mass is $m_c = 1.0\,\mathrm{kg}$, and the pendulum mass is $m = 0.1\,\mathrm{kg}$. The pendulum length is $l = 0.5\,\mathrm{m}$ (i.e., half the total length), and the simulation uses a discretization step of $\tau = 0.02\,\mathrm{s}$. The system is controlled by applying a horizontal force $F \in \{-10, +10\}\,\mathrm{N}$ to the cart, resulting in a discrete two-element action space defined as $\mathcal{A} = \{-10, +10\}\,\mathrm{N}$. The pole must be balanced within an angular deviation of $|\theta| < 12°$ while the cart position is constrained to the track limits $-2.4\,\mathrm{m} < x < 2.4\,\mathrm{m}$. The experiment terminates when the controller is able to balance the pole continuously for 100,000 time steps.

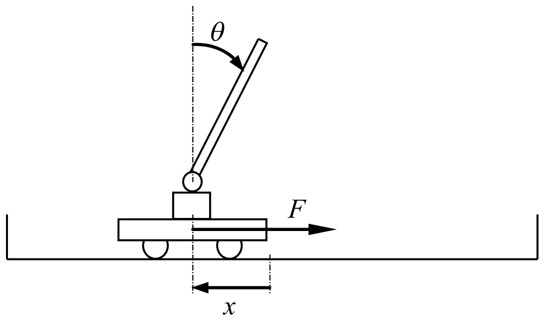

Fig. 2. Schematic of the Cart-Pole system with position and angle constraints.

c) Mountain Car.: The mountain car problem is a classical RL benchmark for function approximation and policy learning [1], [2], [4]. The state variables for this system are the position of the cart, x, and its velocity, $\dot{x}$. The values of these variables are constrained to the intervals $[-1.2,\ 0.5]\,\mathrm{m}$ and $[-0.07,\ 0.07]\,\mathrm{m/s}$, respectively. The state is updated according to the following equations:

$$\begin{aligned} x_{t+1} &= x_t + \dot{x}_{t+1}, \\ \dot{x}_{t+1} &= \dot{x}_t + 0.001\,a_t - 0.0025\,\cos(3\,x_t), \end{aligned} \tag{3}$$

where $a \in \mathcal{A}$ denotes the action. The action set $\mathcal{A}$ typically consists of three discrete values, $\{-1, 0, 1\}$, which can be interpreted as "reverse", "neutral", and "forward", respectively. The goal of the control strategy is to reach the hilltop marked as the "Goal" in Fig.3. Due to limited engine power, the car cannot ascend directly and must instead build momentum by oscillating between slopes, converting kinetic energy into potential energy to reach the target state.

B. RL Algorithm

Reinforcement learning problems are commonly modeled as Markov Decision Processes (MDPs), characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ denotes the state transition probability, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor [26].

A key objective in RL is to discover an optimal policy π^* that maximizes the expected cumulative reward. One approach to this is to estimate the action-value function $Q^\pi(s, a)$, which defines the expected return when starting in state s, taking action a, and following policy π thereafter.

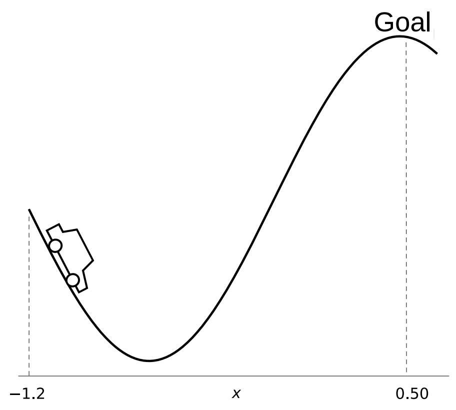

Fig. 3. Schematic of the Mountain Car system.

When the transition model $\mathcal{P}$ is unknown or difficult to compute, model-free methods such as Q-learning [2] can be employed. Q-learning estimates the optimal action-value function $Q^*(s, a)$ directly through interaction with the environment using the Bellman optimality equation as an update rule. A commonly used variant of Q-learning is *Q(0)-learning*, which corresponds to the standard form of the algorithm where only the currently visited state-action pair is updated at each time step. The update rule of Q is expressed using the *temporal-difference error* δ_t as follows:

$$\delta_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t), \tag{4}$$

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\,\delta_t, \tag{5}$$

where $\alpha \in (0, 1]$ is the learning rate and $\gamma \in [0, 1)$ is the discount factor. This method updates values incrementally based solely on immediate experience, which may lead to slow propagation of value information over longer time scales.

To accelerate learning in environments with delayed rewards, the *Q(λ)-learning* algorithm extends Q(0) by introducing eligibility traces [1], [2], [5]. These traces allow the algorithm to assign credit to multiple recently visited state-action pairs. The eligibility trace $e_t(s, a)$ keeps track of how recently and frequently each pair (s, a) has been visited, and the Q-values are updated according to:

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha\,\delta_t\,e_t(s, a), \tag{6}$$

with eligibility traces decaying over time as:

$$e_{t+1}(s, a) = \begin{cases} \gamma\lambda e_t(s, a) + 1, & \text{if } (s, a) = (s_t, a_t), \\ \gamma\lambda e_t(s, a), & \text{otherwise,} \end{cases} \tag{7}$$

where $\lambda \in [0, 1]$ controls the decay rate of the traces. The use of eligibility traces results in faster convergence and more effective credit assignment across temporally extended sequences of actions.

C. State Discretization and Fuzzification

This study employs two alternative representations of the action-value function Q.

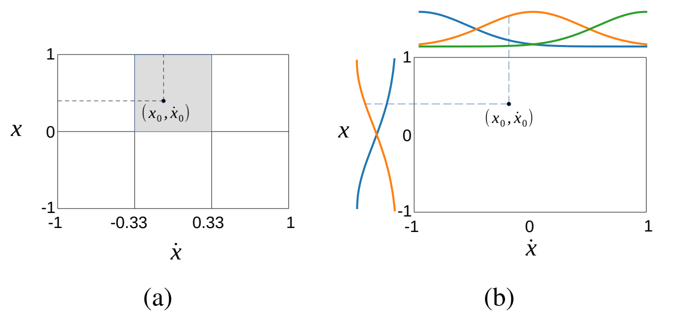

Fig. 4. Comparison of two state-space representations of the Ball–Beam system: (a) crisp discretization of the state variables x and $\dot{x}$ into 2 and 3 bins, respectively, and (b) fuzzy partitioning of the same variables using overlapping membership functions — 2 fuzzy sets for x and 3 fuzzy sets for $\dot{x}$.

A common and straightforward approach for representing $Q(s,a)$ in reinforcement learning is *crisp discretization* of the continuous state space. Each continuous state variable is partitioned into a finite number of bins, typically using uniform intervals or decision-based splits. The resulting combination of discrete bin indices uniquely identifies a state, which serves as a key in a tabular $Q(s,a)$ structure.

Although tabular representations are simple to implement and enable direct learning and analysis, they suffer from several well-known limitations [1]. Chief among these is the *curse of dimensionality*: the Q-table size grows exponentially with the number of state variables and actions, leading to substantial memory and computational demands. Moreover, tabular methods inherently lack generalization, as updates affect only visited states, leaving large portions of the state space unrepresented and unexplored.

Fig. 4 illustrates a local comparison between crisp and fuzzy state-space representations for the Ball–Beam system, focusing on a specific region where $x \in [0,1]$ and $\dot{x} \in [-0.33, 0.33]$. In the crisp discretization (Fig. 4a), this region corresponds to a single discrete state s_i resulting from uniform partitioning of the state variables x and $\dot{x}$ into 2 and 3 bins, respectively, yielding $2 \times 3 = 6$ total discrete states. Any operating point $(x_0, \dot{x}_0)$ in this region is assigned exclusively to one state.

In contrast, the fuzzy representation (Fig. 4b) uses overlapping membership functions—2 for x and 3 for $\dot{x}$—resulting in soft partitions of the same input space. A single point $(x_0, \dot{x}_0)$ activates multiple fuzzy rules to varying degrees. Unlike the crisp case, even small variations within the region continuously modulate the degrees of membership, leading to smooth changes in rule activation. This facilitates gradual policy updates and improved generalization across the state space [27], [28].

Fuzzy Rule-Based Approximation: Fuzzy models offer an interpretable and robust alternative to standard function approximators, particularly under uncertainty or imprecise state observations [28]. The continuous state space is covered by M overlapping fuzzy regions, typically defined by Gaussian membership functions. Let n denote the dimensionality of the state space, and $j \in \{1,\dots,n\}$ index the state variables. The consequent model $Q^{(i)}(s,a;\boldsymbol{q}^{(i)})$ is parameterized by a vector $\boldsymbol{q}^{(i)} \in \mathbb{R}^K$, where K is the number of basis functions used in the approximation (e.g., for a linear model, $K = n+1$). Each fuzzy rule, indexed by $i \in \{1,\dots,M\}$, is expressed as:

$$\text{IF } s_1 \text{ is } \mu_1^{(i)} \text{ AND } \dots \text{ AND } s_n \text{ is } \mu_n^{(i)} \text{ THEN } Q^{(i)}(s,a;\boldsymbol{q}^{(i)}), \tag{8}$$

where the membership function $\mu_j^{(i)}(s_j)$ is defined as:

$$\mu_j^{(i)}(s_j) = \exp\left(-\frac{(s_j - c_j^{(i)})^2}{2(\sigma_j^{(i)})^2}\right), \tag{9}$$

with $c_j^{(i)}$ and $\sigma_j^{(i)}$ denoting the center and spread of the Gaussian function.

The (unnormalized) firing strength of the i-th rule is:

$$\tilde{w}_i(s) = \prod_{j=1}^{n} \mu_j^{(i)}(s_j), \tag{10}$$

and the normalized firing strength is given by:

$$w_i(s) = \frac{\tilde{w}_i(s)}{\sum_{m=1}^{M} \tilde{w}_m(s)}. \tag{11}$$

Each local model $Q^{(i)}$ typically has a simple parametric form:

$$Q^{(i)}(s,a;\boldsymbol{q}^{(i)}) = \sum_k q_k^{(i)} f_k(s,a), \tag{12}$$

where $f_k(s,a)$ are basis functions associated with the rule's consequent. In the Takagi–Sugeno (T–S) framework, these functions often correspond to linear components or action-dependent constants, e.g., $f_k(s,a) = \mathbb{I}_{a=a_k}$ (indicator functions). Thus, $Q^{(i)}(s,a)$ may be understood as a local lookup table of action values per rule, without requiring explicit radial basis functions or nonlinear terms [1].

The global action-value approximation is then a normalized weighted sum of the local models:

$$\hat{Q}(s,a;\boldsymbol{q}) = \sum_{i=1}^{M} w_i(s) \cdot Q^{(i)}(s,a;\boldsymbol{q}^{(i)}), \tag{13}$$

where $\boldsymbol{q}$ denotes the concatenation of all local parameter vectors $\boldsymbol{q}^{(i)}$.

Learning with Temporal-Difference Updates: To train the fuzzy Q-function, the temporal-difference (TD) error is defined as:

$$\delta_t = r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \boldsymbol{q}_t) - \hat{Q}(s_t, a_t; \boldsymbol{q}_t). \tag{14}$$

Fuzzy Q(0)-Learning Update Rule: The parameters are updated via gradient descent:

$$\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \alpha \cdot \delta_t \cdot \nabla_{\boldsymbol{q}} \hat{Q}(s_t, a_t; \boldsymbol{q}_t), \tag{15}$$

where α is the learning rate. The gradient is propagated through the fuzzy aggregation, in which the normalized firing strengths $w_i(s_t)$ determine the contribution of each local model. This form is a general case that encompasses the update rule in (19) for zero-order T–S models.

Fuzzy Q(λ)-Learning Update Rule: In the fuzzy Q(λ)-learning variant, an eligibility trace vector $\mathbf{e}_t$ is maintained:

$$\mathbf{e}_t = \gamma\lambda \cdot \mathbf{e}_{t-1} + \nabla_{\boldsymbol{q}} \hat{Q}(s_t, a_t; \boldsymbol{q}_t), \tag{16}$$

and the parameters are updated as:

$$\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \alpha \cdot \delta_t \cdot \mathbf{e}_t. \tag{17}$$

This formulation combines short-term and long-term learning effects while maintaining the smooth generalization and interpretability of the underlying fuzzy rule base [29], [30]. In the special case of zero-order T–S models, this update reduces to (21) with eligibility traces computed based solely on normalized rule activations $w_i(s_t)$.

D. Fuzzy Q-Learning with Zero-Order T–S Models

This section discusses two specific realizations of fuzzy reinforcement learning algorithms used in this work: *Fuzzy Q(0)-learning* and *Fuzzy Q(λ)-learning*. Both approaches utilize zero-order Takagi–Sugeno (T–S) fuzzy inference systems as function approximators for the action-value function $Q(s, a)$. These methods can be viewed as constrained cases of the general update rules discussed in (15) and (17), where each fuzzy rule has a scalar consequent associated with a given action.

For each action $a \in \mathcal{A}$, a separate set of scalar parameters $q_i^a \in \mathbb{R}$ is maintained. The approximate action-value function is computed as:

$$\hat{Q}_t(s_t, a) = \sum_{i=1}^{M} w_i(s_t) \cdot q_i^a, \tag{18}$$

where $w_i(s_t)$ denotes the normalized firing strength of the i-th fuzzy rule as defined in (11).

In the *Fuzzy Q(0)-learning* variant, the temporal-difference error δ_t is defined as in (14), and the parameters q_i^a for the selected action a_t are updated using:

$$q_i^{a_t} = q_i^{a_t} + \alpha \cdot w_i(s_t) \cdot \delta_t. \tag{19}$$

In the *Fuzzy Q(λ)-learning* variant, eligibility traces are maintained for each rule-action pair. The trace for rule i and action a_t is updated as:

$$e_i^{a_t} = \gamma\lambda \cdot e_i^{a_t} + w_i(s_t), \tag{20}$$

and the parameter update becomes:

$$q_i^{a_t} = q_i^{a_t} + \alpha \cdot \delta_t \cdot e_i^{a_t}. \tag{21}$$

These simplified updates preserve the generalization capability of fuzzy systems in continuous spaces while reducing computational cost. In contrast to more complex parametric models (e.g., (13)), the zero-order structure provides an interpretable and efficient approximation of $Q(s, a)$.

IV. Experiments

Four experiments were conducted using the three systems described in Section III-A. The experiments were divided into two groups: the first addressed discrete representations of state variables, while the second focused on fuzzy representations. For each type of representation of the action-value function Q, the corresponding form of the reinforcement learning algorithm was applied. In the case of discrete representations, updates followed equations (4–7), while the fuzzy approximations were updated according to equations (18–21).

Experiment 1 investigated the impact of varying the number of discretization bins for the state variables, while keeping the number of available actions fixed. For the Ball–Beam and Mountain Car systems, which involve two state variables, the number of bins was varied independently for each variable in the range from 2 to 10. For the Cart–Pole system, which includes four state variables, a simplified binning strategy was adopted. Specifically, the variables were grouped into two pairs: (x, θ) and $(\dot{x}, \dot{\theta})$. Within each pair, both variables shared the same number of bins, while the number of bins was varied independently between the pairs.

Experiment 2 also used the discrete representation, but this time both the number of state discretization bins and the number of actions were varied. In this configuration, all state variables shared the same number of bins.

Experiment 3 repeated the structure of Experiment 1, but with a fuzzy representation of the state space. Instead of bins, the number of fuzzy sets per variable was varied. For Cart–Pole, the same paired simplification was applied to the fuzzy sets.

Experiment 4 mirrored the setup of Experiment 2, extending it to the fuzzy representation, with both the number of fuzzy sets per variable and the number of actions varied uniformly across all state variables.

TABLE I
Summary of Experimental Configurations

System	Exp.	Representation	Varied Parameters	Discretization Strategy
Ball–Beam and Mountain Car	1	Discrete	Bins only	$v(x),\ v(\dot{x}),\ c(Actions)$
	2	Discrete	Bins + Actions	$v(x == \dot{x}),\ v(Actions)$
	3	Fuzzy	Fuzzy sets only	$v(x),\ v(\dot{x}),\ c(Actions)$
	4	Fuzzy	Fuzzy sets + Actions	$v(x == \dot{x}),\ v(Actions)$
Cart–Pole	1	Discrete	Bins only	$v(x == \theta),\ v(\dot{x} == \dot{\theta}),\ c(Actions)$
	2	Discrete	Bins + Actions	$v(x == \theta == \dot{x} == \dot{\theta}),\ v(Actions)$
	3	Fuzzy	Fuzzy sets only	$v(x == \theta),\ v(\dot{x} == \dot{\theta}),\ c(Actions)$
	4	Fuzzy	Fuzzy sets + Actions	$v(x == \theta == \dot{x} == \dot{\theta}),\ v(Actions)$

All experimental configurations are summarized in Table I. The columns are defined as follows: **System** denotes the control system under study; **Exp.** indicates the experiment number (1–4) corresponding to the configurations described in this section; **Representation** specifies the type of state space representation employed, either discrete binning or fuzzy sets; **Varied Parameters** identifies which parameters were varied during the experiment, such as the number of bins, fuzzy sets, and/or actions; finally, **Discretization Strategy** details the grouping and variation scheme of the state variables and actions, where the notation $v(\cdot)$ means the parameter was varied, $c(\cdot)$ indicates it was held constant, and $==$ signifies that the grouped variables share identical settings. This concise notation facilitates the description of different discretization approaches, particularly for systems with multiple state variables like the Cart–Pole, where variable grouping simplifies parameter tuning.

A. Action Space

The action set in Experiments 1 and 3 was fixed and specified in Section III-A for each control system. In Experiments 2 and 4, where the action set was variable, we adopted a strategy in which the upper and lower bounds of the action set matched the minimum and maximum values of the fixed set defined in Section III-A.

In particular, for the Ball-on-Beam system, where the standard discrete action set is typically

$$A_3 = \left\{-\frac{\pi}{4},\ 0,\ \frac{\pi}{4}\right\},$$

we considered action sets of varying cardinality. More formally, let A_n denote a discretized action set containing n uniformly spaced elements from the interval $\left[-\frac{\pi}{4},\ \frac{\pi}{4}\right]$. Then:

$$A_n = \left\{a_k = -\frac{\pi}{4} + k \cdot \frac{\pi}{2(n-1)}\ :\ k = 0, 1, \dots, n-1\right\}. \tag{22}$$

For instance, $n = 2$ yields a minimal binary action set $A_2 = \{-\frac{\pi}{4},\ \frac{\pi}{4}\}$, while $n = 10$ results in a finer discretization including 10 equidistant control signals within the same bounds. This approach allowed us to systematically evaluate the effect of action granularity on learning performance.

B. reinforcement signal

For Ball-on-Beam System failure occurs if the ball falls off the beam, i.e., when $|x| > L/2$, where L is the beam length. The reward signal is then defined as:

$$r = \begin{cases} 0, & |x| \le L/2, \\ -1, & |x| > L/2. \end{cases} \tag{23}$$

The failure states for the inverted pendulum are defined as exceeding the workspace boundaries ($|x| > 2.4$ m) or the pendulum deviating by more than 12 degrees from the upright position ($|\theta| > 12°$). The reward signal is thus defined as:

$$r = \begin{cases} 0, & \text{if } |x| \le 2.4 \text{ and } |\theta| \le 12°, \\ -1, & \text{otherwise}. \end{cases} \tag{24}$$

This type of sparse, failure-driven reward structure is commonly adopted in classic control benchmarks such as CartPole and Ball-on-Beam. It penalizes the agent only when a terminal (failure) condition is reached, encouraging policies that maintain system stability for as long as possible without assigning explicit rewards for intermediate states. Although such a binary reward formulation simplifies the task design, it increases the difficulty of credit assignment and exploration, especially in environments with long horizons or rare failures [1], [25]. This makes it a suitable testbed for evaluating the robustness and sample efficiency of reinforcement learning algorithms.

In the case of the Mountain Car problem, the reward signal was defined as:

$$r = \begin{cases} 1, & x \ge 0.5, \\ -1, & \text{otherwise}. \end{cases} \tag{25}$$

This sparse reward formulation encourages the agent to reach the goal position $x \ge 0.5$ as quickly as possible, while penalizing all other states equally. Such a binary reward structure is commonly used in simplified reinforcement learning benchmarks to evaluate the exploration efficiency of algorithms [1], [31].

C. Initial Conditions and Hyperparameters

The actions were selected according to an ϵ-greedy policy [1]. Each trial consisted of 100,000 iterations of the learning algorithm. It was assumed that if the controller avoided terminal states for this duration, the trial was considered successful [32], [33].

Table II summarizes the hyperparameters used for each environment and agent type. The values were chosen based on commonly adopted configurations in the literature [1], [33], [34] and empirical tuning. Note that the Ball-on-Beam system is less standardized than the Cart-Pole or Mountain Car tasks, so parameter selection relied partially on the characteristics of the physical model and previous control studies [35].

TABLE II
LEARNING PARAMETERS FOR EACH ENVIRONMENT

Environment	**Type**	ϵ	γ	α	λ
Ball-on-Beam	Discrete	10^{-6}	0.9	0.1	0.9
	Fuzzy	10^{-6}	0.995	0.1	0.5
Cart-Pole	Discrete	10^{-2}	0.995	0.1	0.5
	Fuzzy	10^{-2}	0.995	0.1	0.5
Mountain Car	Discrete	$5 \cdot 10^{-2}$	0.95	0.01	0.6
	Fuzzy	$5 \cdot 10^{-2}$	0.95	0.01	0.6

The initial states for each environment are listed below. These were selected to place the agent in a neutral configuration with minimal bias toward success or failure. They reflect typical starting conditions used in the literature and ensure comparability across experiments.

- **Ball-on-Beam:** $x = 0,\ \dot{x} = 0$
- **Cart-Pole:** $x = 0,\ \dot{x} = 0,\ \theta = 0,\ \dot{\theta} = 0$
- **Mountain Car:** $x = -0.5,\ \dot{x} = 0$

D. Experiments Results

The results of the conducted experiments are presented in Figures 5–7. Each row in these figures corresponds to one of the experiments (Exp. 1–Exp. 4), while the left and right columns display the outcomes for Q(0)-learning and Q(λ)-learning, respectively. For Exp. 3 and Exp. 4, these represent Fuzzy Q(0)-learning and Fuzzy Q(λ)-learning.

All experiments were performed on three environments: Ball-Beam, Cart-Pole, and Mountain Car, which differ in control complexity, reward structure, and sensitivity to delayed reinforcement. The experimental configurations are summarized in Table I.

The heatmaps presented in Figures 5–7 show the average number of episodes required to reach the learning goal under various discretization settings, enabling comparative analysis of convergence speed and parameter sensitivity across environments and algorithms. The best-performing configuration in each case, highlighted with a red rectangle in the figures, was subsequently reported in Table III, along with the corresponding discretization parameters.

In the Ball-on-Beam environment, Fuzzy Q(λ)-learning consistently delivered the best results, often converging in fewer than 2 episodes. For example, in Exp. 3 it achieved an average of 1.3 episodes, compared to 2.9 for Fuzzy Q(0)-learning and over 90 for standard Q(0)-learning in Exp. 1. These outcomes underscore the benefits of using eligibility traces and fuzzy approximation in low-dimensional environments with smooth dynamics.

For the Cart-Pole environment, all algorithms ultimately reached convergence, but the efficiency varied substantially. Increasing the resolution of the state space (e.g., from 2 to 10 bins) had a more positive effect on Q(λ)-learning, indicating its greater sensitivity to finer state representations. Fuzzy approaches in Exp. 3 and Exp. 4 again proved more effective, reaching the goal in under 5 episodes on average. In Exp. 3, Fuzzy Q(λ)-learning achieved 3.9 episodes, whereas in Exp. 1, standard Q(λ)-learning required as many as 27 episodes.

In the Mountain Car environment, convergence typically required more than 100 episodes for all algorithms, due to the fact that it is a goal-reaching task and, unlike the other two tasks, the system starts far from the control target. Additionally, the agent must learn long-term planning and energy accumulation from an initial valley state, making the problem more challenging with respect to temporal credit assignment and exploration. Nonetheless, Fuzzy Q(λ)-learning achieved slightly better results than other variants—for instance, 114.1 episodes in Exp. 3 compared to 125.9 for Fuzzy Q(0)-learning. These improvements, while modest, suggest that eligibility traces and function approximation can provide benefits even in more challenging environments.

The optimal discretization settings varied across environments and algorithms. Coarse resolutions (2–4 bins) were generally sufficient for Cart-Pole and Mountain Car, whereas Ball-on-Beam benefited from finer discretizations (e.g., 9–10 bins or fuzzy sets) when used with Q(λ)-learning. These findings emphasize that optimal parameterization is both environment- and algorithm-dependent, further justifying the systematic analysis presented in this study.

V. Summary of Experimental Results

This section consolidates the findings from all four experiments conducted on the three tested environments. Table III provides a key reference by summarizing the parameter configurations—state discretization or fuzzy set counts and number of actions—that yielded the lowest number of episodes to reach the learning threshold.

The following conclusions and recommendations can be drawn:

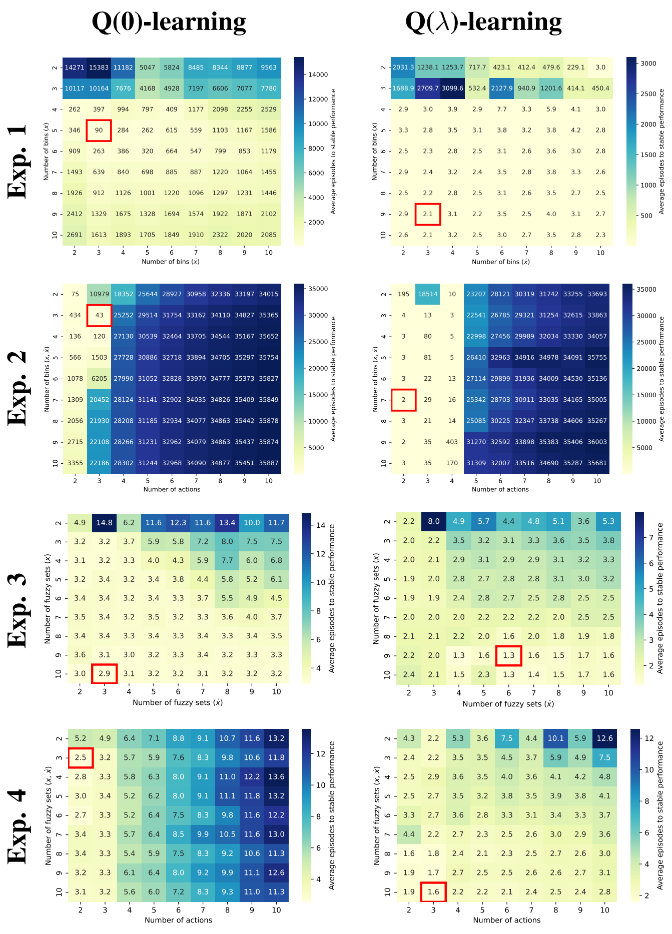

Fig. 5. Performance comparison between Q(0)-learning and Q(λ)-learning for the Ball–Beam system across four experimental conditions.

- **Preferred Configurations:** Across all environments, compact representations—typically 2–4 bins or fuzzy sets and 2–4 actions—were sufficient and often optimal. Q(λ)-learning showed particular benefit from slightly richer fuzzy representations, while Q(0)-learning favored simpler configurations.
- **Configurations to Avoid:** High-resolution discretizations (e.g., 6–10 bins or fuzzy sets) combined with large action sets tended to degrade performance, especially in Q(0)-learning. These configurations increased variance and slowed convergence. Excessively fine representations did not offer benefits and, in some cases, destabilized learning.
- **Algorithm-specific Sensitivities:** Q(λ)-learning demonstrated greater robustness to increased representation complexity and generally converged faster across most scenarios. In contrast, Q(0)-learning required well-tuned compact configurations to achieve reasonable performance and was more sensitive to overparameterization.
- **Environment-specific Behavior:** For environments with sparse or delayed rewards (e.g., Mountain Car), Q(λ)-learning offered a notable advantage due to eligibility traces. In smoother environments such as Ball-on-Beam, even basic configurations led to fast learning, with Q(λ)-

TABLE III
BEST-PERFORMING CONFIGURATIONS SELECTED BASED ON THE RESULTS HIGHLIGHTED IN RED IN FIGURES 5–7, ALONG WITH CORRESPONDING DISCRETIZATION PARAMETERS.

				Best Parameter Configuration				
System	**Exp.**	**Algorithm**	**Performance**	x	$\dot{x}$	θ	$\dot{\theta}$	Actions
Ball–Beam	1	Q(0)-learning	90.5 ± 25.8	5	3	—	—	3
	1	Q(λ)-learning	2.1 ± 1.1	9	3	—	—	3
	2	Q(0)-learning	43.2 ± 20.4	3	3	—	—	3
	2	Q(λ)-learning	2.3 ± 1.5	7	7	—	—	2
	3	Fuzzy Q(0)-learning	2.9 ± 1.1	10	3	—	—	3
	3	Fuzzy Q(λ)-learning	1.3 ± 0.6	9	6	—	—	3
	4	Fuzzy Q(0)-learning	2.55 ± 0.73	3	3	—	—	2
	4	Fuzzy Q(λ)-learning	1.59 ± 0.88	10	10	—	—	3
Cart–Pole	1	Q(0)-learning	42.4 ± 13.8	2	2	2	2	2
	1	Q(λ)-learning	27.2 ± 6.3	10	2	10	2	2
	2	Q(0)-learning	42.4 ± 13.8	2	2	2	2	2
	2	Q(λ)-learning	41.4 ± 24.4	2	2	2	2	2
	3	Fuzzy Q(0)-learning	4.5 ± 2.1	2	2	2	2	2
	3	Fuzzy Q(λ)-learning	3.9 ± 1.5	4	2	4	2	2
	4	Fuzzy Q(0)-learning	4.54 ± 2.06	2	2	2	2	2
	4	Fuzzy Q(λ)-learning	4.57 ± 1.35	2	2	2	2	2
Mountain Car	1	Q(0)-learning	121.6 ± 11.4	2	2	—	—	3
	1	Q(λ)-learning	143.5 ± 4.3	3	9	—	—	3
	2	Q(0)-learning	121.2 ± 8.6	2	2	—	—	3
	2	Q(λ)-learning	162.0 ± 32.6	6	6	—	—	2
	3	Fuzzy Q(0)-learning	125.9 ± 11.2	4	4	—	—	3
	3	Fuzzy Q(λ)-learning	114.1 ± 3.5	4	4	—	—	3
	4	Fuzzy Q(0)-learning	116.6 ± 2.9	4	4	—	—	2
	4	Fuzzy Q(λ)-learning	114.1 ± 3.5	4	4	—	—	3

Note: Columns x, $\dot{x}$, θ, $\dot{\theta}$ and Actions indicate discretization or action parameters for which the best results were obtained. — means that the state variable is not present in the model. **Exp.** refers to the experiment numbers defined in Table I.

learning and fuzzy approximations further improving results.

These insights may guide the design of reinforcement learning agents for similar control tasks, particularly when interpretability and parameter economy are desired.

VI. Discussion

The empirical findings reinforce the practical value of combining eligibility traces with fuzzy state approximators. Fuzzy Q(λ)-learning consistently yielded superior convergence speed and stability across environments, particularly in the Ball-on-Beam and Cart-Pole tasks. These results validate prior theoretical insights [5], [14], [17] and empirical outcomes from control-oriented fuzzy RL [20], [21].

Importantly, the benefits of fuzzy approximation were most pronounced when paired with low to moderate granularity of state partitions. This aligns with recent studies on scalable fuzzy architectures, such as hierarchical [22] and self-organizing systems [18], where rule base complexity is balanced against generalization needs. Our results suggest that fixed antecedent structures with learned consequents, as popularized by Takagi–Sugeno fuzzy systems, can effectively reduce the dimensionality burden without sacrificing learning quality.

In sparse-reward environments like Mountain Car, all methods struggled to converge rapidly. While Fuzzy Q(λ) maintained a relative advantage, the performance gap narrowed. This supports the notion that temporal credit assignment alone is insufficient in reward-scarce settings, and further enhancements such as intrinsic motivation or model-based components may be necessary.

Overall, our findings bridge the gap between foundational fuzzy RL theory and modern empirical needs by quantifying the interaction between state–action resolution, algorithmic extensions, and environment-specific difficulty. Future extensions should examine how adaptive fuzzy rule tuning, as in [18], can be combined with exploration-driven heuristics to improve generalization in high-dimensional RL tasks.

VII. Conclusions

This study compared the performance of standard Q(0)-learning and Q(λ)-learning across three benchmark control tasks (Ball-on-Beam, Cart-Pole, and Mountain Car) under varying state-action space representations, including both discrete and fuzzy encodings. The experiments systematically

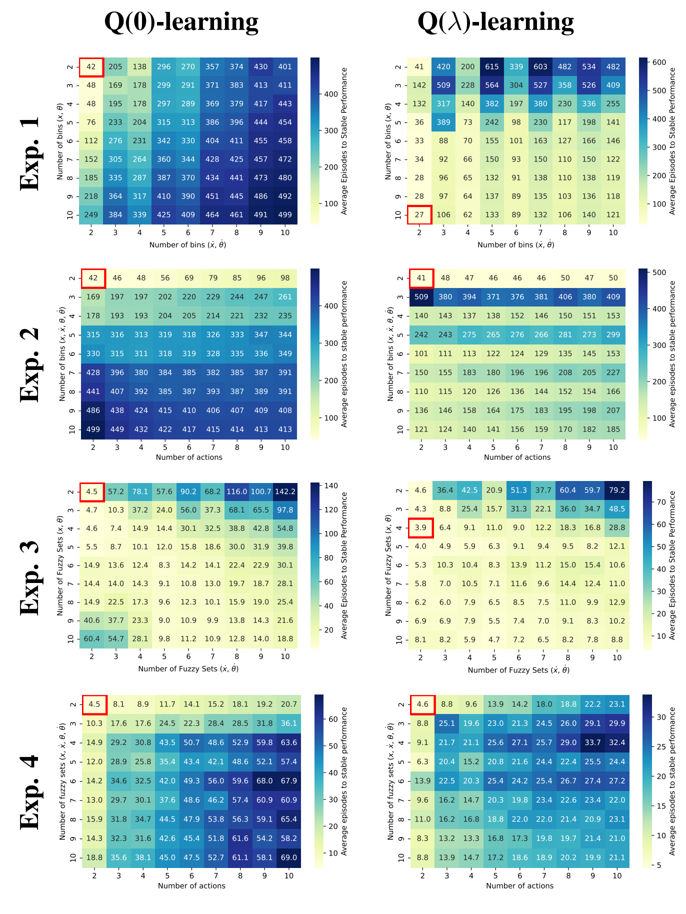

Fig. 6. Performance comparison of Q(0)-learning and Q(λ)-learning for the Cart–Pole system across four experimental conditions.

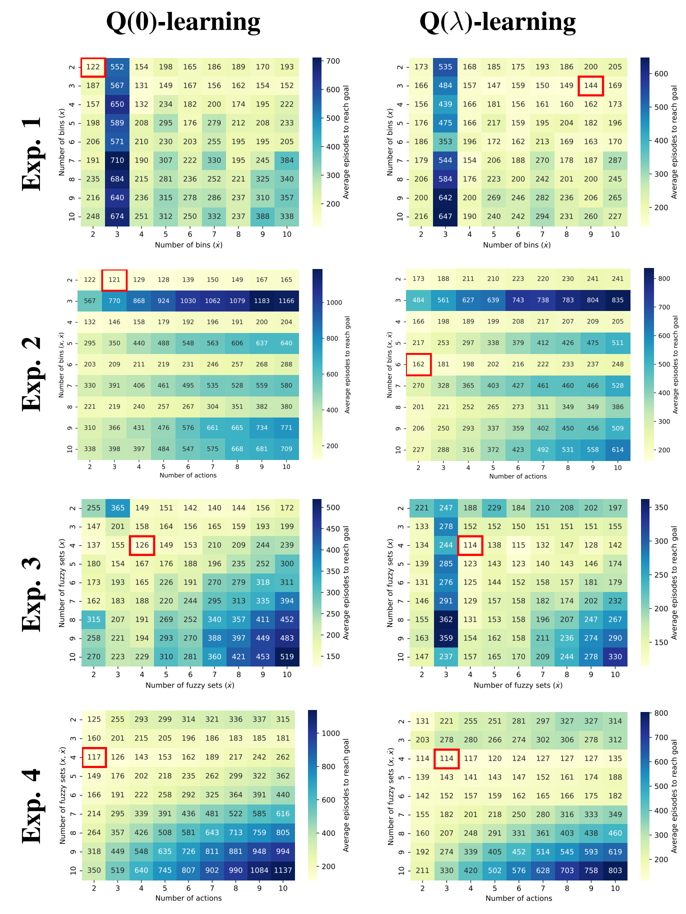

Fig. 7. Performance comparison of Q(0)-learning and Q(λ)-learning for the Mountain Car environment across four experimental conditions.

evaluated multiple configurations of state resolution and action granularity, providing a detailed view of how algorithm performance scales with representation complexity.

The results consistently favored Q(λ)-learning, which demonstrated faster convergence, greater stability, and better scalability with increased state resolution or fuzzy partitioning. In particular, the use of eligibility traces helped overcome delayed reward issues in environments such as Mountain Car. Additionally, the analysis revealed that overly fine-grained discretization or excessive fuzzy partitioning often degrades learning performance, especially for Q(0)-learning.

The empirical findings support the following conclusions:

- Q(λ)-learning outperforms Q(0)-learning in both discrete and fuzzy state representations across all tested environments.
- Compact state-action configurations (e.g., $(2, 3)$, $(3, 2)$, $(4, 2)$) offer a favorable trade-off between representation richness and convergence speed.
- Excessive discretization or fuzzy granularity may hinder learning by introducing sparsity or instability in the value updates.

Future Work

Future research will explore three main directions. First, replacing discrete or fuzzy encodings with continuous function approximators such as radial basis functions or neural networks could enhance generalization in high-dimensional environments. Second, extending the current framework to policy optimization methods—including actor–critic or policy-gradient algorithms—may enable more effective learning in continuous action spaces. Finally, robustness and transferability will be examined by assessing the sensitivity of fuzzy abstractions to noise and non-stationarity, as well as evaluating whether learned state representations can be reused across related tasks.

Ultimately, these directions aim to bridge the gap between interpretable rule-based learning and the flexibility of deep reinforcement learning, moving toward hybrid systems that are both efficient and explainable.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[2] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[3] J. N. Tsitsiklis and B. Van Roy, “Analysis of temporal-differences learning with function approximation,” *Machine Learning*, vol. 24, no. 1, pp. 25–57, 1996.

[4] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. i, 4th ed. ed. Athena Scientific, 2012.

[5] T. Kozuno, E. Imani, and M. Sugiyama, “Revisiting eligibility traces in deep reinforcement learning,” in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, vol. 139. PMLR, 2021, pp. 5865–5876.

[6] T. J. Procyk and E. H. Mamdani, “Adaptive critic designs,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, no. 10, pp. 757–763, 1990.

[7] L.-X. Wang, “Fuzzy system modeling using fuzzy rules and fuzzy reasoning,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 5, pp. 1414–1427, 1992.

[8] P. Chaudhari *et al.*, “On the importance of action representations in deep reinforcement learning,” in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018, arXiv:1811.12560.

[9] N. Kostrikov, “Discrete vs continuous action spaces in policy gradient methods,” *arXiv preprint arXiv:1801.02851*, 2018.

[10] F. Klein and W. Konen, “Adaptive discretization of continuous action spaces for on-policy optimization,” *arXiv preprint arXiv:1906.11845*, 2019.

[11] T. J. Procyk and D. P. Filev, “Fuzzy q-learning for general state–action spaces,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 30, no. 1, pp. 76–84, 2000.

[12] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, “Reinforcement learning with parameterized actions,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 7263–7270.

[13] L. Jouffe, “Fuzzy q-learning: a reinforcement learning approach for autonomous agents,” in *Proceedings of the Sixth International Fuzzy Systems Conference*, vol. 2. IEEE, 1997, pp. 1351–1356. [Online]. Available: https://doi.org/10.1109/FUZZY.1997.616345

[14] P.-Y. Glorennec and L. Jouffe, “Fuzzy q-learning with function approximation,” in *Proceedings of the Sixth International Fuzzy Systems Conference*, vol. 2. IEEE, 1997, pp. 1311–1316. [Online]. Available: https://doi.org/10.1109/FUZZY.1997.616333

[15] S. Wen, R. Ma, L. Dong, and Y. Liu, “Q-learning trajectory planning based on takagi–sugeno fuzzy parallel distributed compensation structure of humanoid manipulator,” *Soft Computing*, vol. 23, no. 18, pp. 8483–8492, 2019. [Online]. Available: https://doi.org/10.1007/s00500-018-3511-0

[16] E. Zander, B. van Oostendorp, and B. Bede, “Reinforcement learning with takagi–sugeno–kang fuzzy systems,” *Complex Engineering Systems*, vol. 3, no. 2, pp. 48–63, 2023. [Online]. Available: https://doi.org/10.1016/j.cesys.2023.04.001

[17] S. Zahmatkesh, B. Behnam, A. Vahidi, and H. N. Pishkenari, “Robust fuzzy q-learning for autonomous landing of fixed-wing uavs under wind disturbances,” *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105579, 2023. [Online]. Available: https://doi.org/10.1016/j.engappai.2022.105579

[18] R. Hostetter, G. Luger, and M. Morales, “Self-organizing fuzzy q-learning networks: Towards interpretable and competitive offline reinforcement learning,” *Neurocomputing*, vol. 524, pp. 239–253, 2023. [Online]. Available: https://doi.org/10.1016/j.neucom.2022.12.093

[19] E. Zander, B. van Oostendorp, and B. Bede, “Takagi–sugeno fuzzy dqns: Combining interpretable fuzzy inference and deep q-learning,” *Applied Soft Computing*, vol. 135, p. 110009, 2023. [Online]. Available: https://doi.org/10.1016/j.asoc.2023.110009

[20] J. Tang, J. Liu, and W. Zhang, “A fuzzy actor–critic algorithm for energy-efficient load balancing in smart grids,” *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 6, pp. 1247–1259, 2021.

[21] Y. Wang, Y. Chen, and Z. Li, “Interpretable fuzzy deep reinforcement learning for autonomous vehicle navigation,” *Engineering Applications of Artificial Intelligence*, vol. 110, p. 104716, 2022.

[22] M. Lei and X. Lin, “Hierarchical fuzzy reinforcement learning for multi-task control: A generalizable framework,” *Neurocomputing*, vol. 521, pp. 73–85, 2023.

[23] P. E. Wellstead, *Introduction to Physical System Modelling*. Chennai, India: Control Systems Principles, 2000.

[24] S. Larsson, “ballbeam-gym: Ball & beam openai gym environments,” https://github.com/simon-larsson/ballbeam-gym, 2019, accessed: 2025-07-25.

[25] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, September/October 1983.

[26] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, reprint ed. John Wiley Sons, 2014.

[27] L.-J. Lin, “Reinforcement learning for robots using neural networks,” *Technical Report CMU-CS-94-207, Carnegie Mellon University*, 1994.

[28] T. Procyk and E. H. Mamdani, “An adaptive fuzzy controller using fuzzy logic,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, no. 5, pp. 360–371, 1977.

[29] C.-T. Lin and C.-S. G. Lee, “Reinforcement learning for an adaptive fuzzy logic controller,” in *Fifth IEEE International Conference on Fuzzy Systems*, vol. 3, 1996, pp. 2052–2058.

[30] D. Nauck and R. Kruse, “Neuro-fuzzy systems: State-of-the-art modeling techniques,” *The Knowledge Engineering Review*, vol. 18, no. 3, pp. 241–257, 2003.

[31] A. W. Moore, “Efficient memory-based learning for robot control,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 1990.

[32] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pp. 278–287, 1999.

[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.

[34] S. Larsson, “ballbeam-gym: A reinforcement learning environment for ball and beam control,” https://github.com/simon-larsson/ballbeam-gym, 2022, accessed: 2025-07-25.

[35] D. Wang and D. J. Hill, “Global stabilization of the ball and beam system based on fuzzy control approach,” *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 3, pp. 418–425, 1997.