Iranian Journal of Fuzzy Systems

Volume 22, Number 4, (2025), pp. 161-174

Original paper

Optimizing Deep Q-Networks with fuzzy inference-based adaptive replay buffer management

M. B. Dowlatshahi [1] and S. Beiranvand [2]

[1] *Department of Computer Engineering, Lorestan University, Khorramabad, Iran.*
[2] *Department of Computer Engineering, Technical and Vocational University (TVU), Tehran, Iran.*

dowlatshahi.mb@lu.ac.ir, Sbeyranvand@tvu.ac.ir

Abstract

Deep reinforcement learning algorithms, such as Deep Q-Networks (DQN), require careful tuning of replay memory parameters. In standard DQN implementations, these parameters remain fixed, which conflicts with the dynamic nature of the learning process where environmental conditions and reward stability continuously change. This mismatch often results in unstable learning or slow convergence. In this paper, we present a fuzzy logic-based system for adaptively adjusting three key replay memory parameters: memory size, the ratio of recent samples, and priority weight. The proposed fuzzy system evaluates the agent's state by monitoring reward variations and average training errors, and accordingly updates these parameters to maintain optimal values during training. To assess the effectiveness of the proposed approach, we compared it with conventional DQN and PER-DQN methods across three benchmark reinforcement learning environments: *CartPole_v1*, *LunarLander_v2*, and *Taxi_v3*. Experimental and statistical analyses demonstrate that our method improves average rewards, reduces training time, and enhances learning stability.

Keywords: Reinforcement learning, deep neural networks, replay memory, fuzzy logic, adaptive parameter tuning.

1 Introduction

As an associated branch of artificial intelligence, Deep Reinforcement Learning (DRL) applies deep neural networks along with reinforcement learning, and has seen substantial progress in video games, robotics, and decision making [15]. One of the most impactful algorithms in this area is Deep Q-Network (DQN), developed by Mnih et al. in 2015, which marked the first instance of surpassing human-level performance in Atari games [9]. Using deep neural networks, DQN Q-value function approximation is done together with a replay buffer, which enhances learning and reduces instability associated with the process [9]. To improve learning, the replay buffer uses deep neural networks and stores past experiences, mitigating sample correlations and preserving training stability [8]. While replay memory has several parameters, including memory size and sampling prioritization, these are often fixed values. This rigidity prevents adaptation of the algorithm to the changes in the environment and within the learning process, leading to the potential of slow and unstable convergence [14].

To overcome this limitation, some methods proposing adaptive tuning of replay memory parameters have been introduced, which modify these parameters based on evaluating the learning conditions, thus improving the performance of the model [20]. A fuzzy logic application is one of the more effective methods of coping with this issue, as it is able to model and make decisions on the basis of uncertainty [21]. Fuzzy logic has been applied for purposes in machine learning and adaptive control because of its flexibility and robustness against noisy and dynamic data [6]. In reinforcement learning, fuzzy systems permit the intelligent tuning of critical parameters, which enhances the stability and speed of convergence during the training, thus the performance improves [22]. In particular, fuzzy logic can modify the replay

Corresponding Author: M. B. Dowlatshahi
Received: September 2025; Accepted: Invited paper.
https://doi.org/10.22111/ijfs.2025.9358

memory parameters based on the changes of some metrics like reward values and their averages, which improves the learning quality in the algorithms based on DQN.

This study aims to develop a fuzzy logic-based method for adaptive adjustment of the three replay memory parameters of the DQN algorithm: memory size, recent sample ratio, and priority weight. The method enhances convergence and learning stability by dynamically optimizing the parameters based on the input factors of the learning process. To test the effectiveness of the method, we run experiments in standard reinforcement learning testbeds: CartPole-v1, LunarLander-v2, and Taxi-v3, and we benchmark the results against those of DQN and PER-DQN algorithms. The results bolster the effectiveness of fuzzy logic-based adaptive tuning as average rewards increase, training time decreases, learning stability improves, and overall performance in reinforcement learning improves-demonstrating the power of the proposed method. The remainder of this paper is organized as follows: Section 2 reviews related studies on deep reinforcement learning, the DQN algorithm, replay memory, and the application of fuzzy logic for parameter tuning. Section 3 presents the proposed fuzzy logic-based adaptive replay memory tuning method. Section 4 details the experimental setup, evaluation environments, and results analysis. Finally, Section 5 discusses the findings, and Section 6 concludes the paper with a summary of contributions and suggestions for future research.

2 Related work

The Q-Learning algorithm has been an important backbone for many of the recent advances in Deep Reinforcement Learning: The Deep Q-Network (DQN) algorithm of Mnih et al. [9]. From the beginning, constructions have been attempted to advance the performance and training stability of network architectures. For example, overestimation is a critical error, and various extensions, including Double DQN [17] and Dueling DQN [19], which utilize known techniques to mitigate such overestimation and to accelerate convergence obtained better performance in more complex domains. In addition to improving network structure, considerable attention has been paid to properties of the replay buffer and how its parameters alter learning. The use of a replay buffer in DQN algorithms helps to break the correlations among samples and promote the stability of the training process [8]. Nevertheless, critical factors such as the memory size, the sampling strategy, and the priority weighting are often static and decided a priori in the baseline implementations, which might hamper the efficiency of learning. To deal with this, more sophisticated methods (e.g., Prioritized Experience Replay (PER) have been developed, where we assign a larger weight to important experience, leading to faster convergence [14].

Besides naïve prioritization methods, some of the existing works mainly investigate the adaptive ways of setting replay memory parameters. These approaches respond to the learning context by adapting parameters such as memory size and attention rank weighting, thus enhancing responsiveness to changes in the environment and learning state. One such example would be Wang et al. [20], who presented an automatic tuning approach to memory parameter retrieval in reinforcement learning. It showed improved performance across multiple standard test environments. There are also other reports, such as [11], where tuned parameters are set without the need for human supervision due to automated machine learning (AutoML) capabilities.

Parameter tuning in reinforcement learning seems to greatly benefit from fuzzy logic as a strategy. Zadeh [21] introduced fuzzy logic, and since then, it has become possible to model uncertainties and noisy data; thus, applying it for control over parameter changes in a system is complex. Fuzzy logic has been used in reinforcement learning as decision systems where they dynamically change learning parameters like rates, memory, and weighting of experiences, and thus allow for better and quicker stable convergence during training [2, 6, 22]. The above findings combine fuzzy logic and reinforcement learning to solve issues revolving around parameter change tuning dynamically.

Recently, the optimization of replay memory using fuzzy logic methods combined with reinforcement learning algorithms has been analyzed. One of the important developments in this area is Prioritized Experience Replay, pioneered by Schaul et al. (ICLR 2016), which has been shown to improve the learning stability and convergence speed of reinforcement learning algorithms. Prioritized Experience Replay is used as a basis for a fuzzy-reinforcement system that adjusts replay memory parameters in real time. Some works have focused on fuzzy parameter tuning for Prioritized Experience Replay and have been shown to boost the performance of DQNs. These methods rely on fuzzy logic to adjust parameters based on reward changes and error values. Intelligent parameter adjustment in DQN is not restricted to the replay memory only. Other parameters like the learning rate, discount factor, and even the network topology have a considerable impact on the learning performance. Research has shown that adaptive dynamic tuning of DQN parameters, especially in complex variable environments, improves the speed of optimization and stability in algorithms. These methods frequently make use of artificial intelligence techniques like deep learning, evolutionary algorithms, and AutoML to find the best configurations.

An open issue in reinforcement learning is how to systematically maintain a training scaffold under shifting and unpredictable surrenders. To solve this, studies [3, 10, 23] proposed algorithms based on multi-task learning and predictive models focused on improving the learning performance in changing environments. These techniques can be combined with adaptive replay memory tuning to improve the quality of learning in the overall system. In the last few years, there has been a considerable focus on the application of deep reinforcement learning algorithms in complex, authentic environments. Studies [7, 23] investigated methods in which adaptive parameter control is based on environmental data and feedback provided in real-time. These approaches, which combine artificial intelligence and fuzzy logic, strengthen learning capability in the presence of noise and under unstable conditions, providing deep learning algorithms more flexibility.

In general, improvements to the Deep Q-Network have been made with respect to tuning replay memory parameters. Static parameters within DQN are known to hinder performance, while dynamic and intelligent parameter adjustment can greatly improve learning efficiency and stability. In this case, reinforcement learning combined with fuzzy logic has shown success in adjusting parameters facing uncertainty within the environment, when those parameters need to be managed the most. Moreover, designing fuzzy-reinforcement systems to optimize replay memory and Prioritized Experience Replay has led to faster convergence and increased training stability. Also, dynamically adjusting other important algorithm parameters, such as learning rate and network structure, increases overall learning performance and is particularly beneficial in multi-task and dynamic environments. Considering the increasing use of reinforcement learning in more complex real-life scenarios, adjusting parameters within the scope of real-time feedback and fuzzy logic is an important direction for further research on this topic.

3 Proposed method

In this work, we introduce a new method to improve DQN agent learning by automatically adjusting replay memory parameters with a fuzzy inference system. The underlying principle of this approach is to adaptively control the size of replay memory and the proportion of recent experiences sampled and priority-based weighing of samples according to the agent's learning state. In traditional DQN methods, the parameters related to replay memory are set to constant values that do not change during training. This fixed configuration is likely to hinder the agent's adaptability to changes in the environment and the rewards. In contrast, the proposed approach uses a fuzzy logic system to change these parameters in a dynamic way, allowing the agent to perform optimally in different situations.

3.1 Key components of the proposed method

The proposed approach includes four key elements:

1. **Q-Network:** The Deep Q-Network that calculates the Q-value for each possible state-action pair.
2. **Replay Memory:** A structure to store and manipulate the experiences of the agent as it trains. This memory allows adaptive modification of its size and sampling optimization.
3. **Fuzzy Inference System:** This system tunes replay memory parameters, including memory size (`mem_size`), recent sample ratio (`recent_sample_ratio`), and priority weighting (`priority_weight`), according to inputs such as the rewards' standard deviation (`reward_std`) and the average training loss (`loss_avg`). Both input and output fuzzy sets are defined over numerical ranges, and fuzzy rules are derived from empirical observations.
4. **Learning Policy:** To balance exploration and exploitation, we initially use a ε-greasy learning policy with $\varepsilon = 1.0$, which decays exponentially by 0.99 for each episode.

3.1.1 Fuzzy inference system design details

The fuzzy inference system is constructed as follows.

Inputs:

- Normalized reward in $[0, 1]$.
- Average training loss in $[0, 1]$.

Outputs:

- Replay memory size in $[1000, 10000]$.
- Recent sample ratio in $[0, 1]$.
- Importance weight of experience in $[0, 1]$.

Fuzzy rules are formulated so that an increase in reward variation or training error corresponds to an increase in the memory size, priority weight, and ratio of recent samples.

3.1.2 Training process with adaptive replay memory

The training algorithm proceeds as follows:

1. Initialize a Q-network with random weights and a replay memory with a default size (e.g., 5000 samples).
2. Initialize the fuzzy inference system with the given rule base.
3. For each episode:
    - Reset the environment and define the initial state.
    - Choose an action based on the ε-greedy policy.
    - Act and receive reward, next state, and termination flag.
    - Save the experience $(\text{state}, \text{action}, \text{reward}, \text{next state}, \text{done})$ in replay memory.
    - If the number of samples in memory exceeds a minimum threshold, initiate adaptive sampling: a fraction of samples is taken from recent experiences, and the rest randomly, with fractions determined by the fuzzy system.
    - Calculate the learning error and standard deviation of recent rewards.
    - Update the replay memory parameters using the fuzzy inference system.
    - If memory size changes, resize the replay memory while retaining existing data.
    - Gradually reduce ε to encourage the exploitation of learned knowledge.

By adjusting replay memory parameters, the agent dynamically adapts to varying environmental conditions and reward deviations, accelerating convergence and improving Q-network performance. The complete training procedure with fuzzy logic-guided adaptive replay memory is presented in Algorithm 1.

Algorithm 1 Training Process of DQN Agent with Adaptive Replay Buffer Tuning Using Fuzzy Logic

```
INITIALIZE Q-network with random weights
INITIALIZE Replay Memory with default size
INITIALIZE Fuzzy Logic System
  Inputs: reward, std_loss, avg_loss
  Outputs: mem_size, recent_sample_ratio, priority_weight
for episode = 1 to N do
  Reset environment state
  done ← False
  total_reward ← 0
  while done = False do
    // Select action using ε-greedy policy
    if random() < ε THEN then
      action ← random action
    else
      action ← action with highest Q-value
    end if
    Execute action and observe next_state and reward and done
    STORE (state, action, reward, next_state, done) in Replay Memory
    state ← next_state
    total_reward ← total_reward + reward
    if Replay Memory size ≥ batch_size then
      // Sample batch from Replay Memory
      newest_samples ← recent_sample_ratio × batch_size
      remaining_samples ← randomly from older experiences
      COMPUTE target Q-values and predictions
      UPDATE Q-network
    end if
    // Update fuzzy logic based on recent performance
    reward_std, loss_avg ← statistics from recent experiences
    mem_size, recent_sample_ratio, priority_weight ← Fuzzy Logic System(reward_std, loss_avg)
    ADJUST Replay Memory size according to mem_size
  end while
  DECAY ε for exploration reduction
end for
```

As illustrated, upon the receipt of the environment state, the DQN agent chooses an action according to the ε-greedy policy. Then we carry out the selected action in the environment, and observe the reward and the next state. These experiences are then saved in the Replay Memory, and after a good number of experiences have been stored, the training of the Q-network starts.

An important characteristic of the proposed approach is the utilization of a fuzzy logic system to dynamically adjust three key parameters of the Replay Memory: its size, the percentage of recent experiences in the sample, and the priority level. NewsU is an algorithm to discover the optimal values for those parameters and also to update their values using two types of input, which are the standard deviation of rewards and the average training loss recently. It allows the agent to learn and decide more effectively under diverse environmental situations.

4 Experiments and results

The standard DQN algorithm serves as the baseline, utilizing a replay memory and a deep neural network to approximate the Q-function. However, the replay memory parameters (such as memory size and sampling strategy) remain fixed throughout training. In PER-DQN (Prioritized Experience Replay DQN), experiences are prioritized based on their importance or prediction error, meaning that experiences with higher errors have a higher probability of being sampled. Nonetheless, the memory size and structure remain constant.

The proposed FUZZY-DQN method uses a fuzzy logic system to adjust replay memory size, the sampling ratio of recent experiences, and priority weighting as training progresses. It relies on metrics such as the standard deviation of rewards and average training loss. This approach helps the agent adapt to changing environments and better use both past and recent experiences during learning.

Choosing the right test environment is essential for evaluating how well reinforcement learning algorithms perform. The environment needs to be challenging enough to show both the strengths and weaknesses of each method, and it should also reflect real-world problems in its dynamics and the way states and actions are structured. In this study, we selected three standard environments from the OpenAI Gym suite: LunarLander-v2, Taxi-v3, and CartPole-v1. Each was chosen for a specific purpose. LunarLander-v2 represents a complex control environment with sparse, noise-sensitive rewards. Taxi-v3 is a discrete navigation task featuring a large state space and sequential decisions. CartPole-v1 is a simpler control environment focused on rapid, optimal responses.

LunarLander and CartPole have continuous state spaces with discrete actions, while Taxi-v3 has fully discrete state and action spaces. The learning challenges differ: LunarLander requires long-term stability and robustness to reward fluctuations. Taxi-v3 demands balancing exploration and exploitation in a large state space. CartPole requires fast learning and immediate response to environmental changes. This selection allows evaluation of the proposed algorithm against three common RL challenges: stability under noisy rewards, handling large state spaces, and rapid convergence in simpler tasks.

All experiments were conducted on a system with an Intel(R) Core (TM) i5-3337U 1.80 GHz processor and 12 GB of RAM (11.9 GB usable), running Windows 10 64-bit. Implementations were developed in Python using libraries such as PyTorch, OpenAI Gym, and scikit-learn, with the addition of scikit-fuzzy. Conducting experiments in this environment ensures reproducibility and consistency of results. The algorithm hyperparameters are summarized in Table 1.

All algorithms were executed with identical baseline settings, including learning rate, neural network architecture, batch size, number of episodes, and discount factor, to ensure a fair comparison. The main differences lie in the control parameters directly related to managing training data and sampling strategy:

- In DQN and PER-DQN, the replay buffer size was fixed. In contrast, FUZZY-DQN dynamically adjusted the buffer size based on model performance at each step using metrics such as reward standard deviation and average loss.
- Prioritized sampling in PER-DQN was performed with fixed parameters, while FUZZY-DQN leveraged fuzzy-rule-based priority weighting, demonstrating higher adaptability and sensitivity to training conditions.

Table 1: Configured hyperparameters for all algorithms

Parameter	**DQN**	**PER-DQN**	**FUZZY-DQN (Proposed)**
Number of episodes	50	50	50
Learning rate (Adam)	0.001	0.001	0.001
Batch size	64	64	64
Discount factor (γ)	0.99	0.99	0.99
Initial ε	1.0	1.0	1.0
ε decay rate	0.99	0.995	0.99
Minimum ε	0.05	0.05	0.05
Replay buffer size	5000 (fixed)	5000 (fixed)	1000–10000 (adaptive via fuzzy logic)
Priority-based sampling	No	Yes ($\alpha = 0.6$, $\beta_{\text{start}} = 0.4$, $\beta_{\text{increment}} = 0.001$)	Yes (priority weight 0–1 via fuzzy logic)
Target network update	No	Every 10 episodes	No
Q-network architecture	2×64 ReLU	2×64 ReLU	2×64 ReLU
Loss function	MSELoss	MSELoss (reduction = 'none')	MSELoss
Additional control parameters	—	—	recent_sample_ratio (0–1), priority_weight (0–1)

The configuration of FUZZY-DQN is presented in Table 2.

Table 2: FUZZY-DQN Configuration

Component	**Defined range**	**Membership functions**	**Description**
Input 1: reward_std	[0, 1]	Poor, Average, Good (automatic)	Normalized standard deviation of recent rewards
Input 2: loss_avg	[0, 1]	Poor, Average, Good (automatic)	Average training loss over recent episodes
Output 1: mem_size	[1000, 10000]	Small, Medium, Large	Size of the replay buffer
Output 2: recent_sample_ratio	[0, 1]	Low, Medium, High	Ratio of most recent samples in each training batch
Output 3: priority_weight	[0, 1]	Low, Medium, High	Weight assigned to prioritized sampling
Fuzzy rules	9 rules	Example: (reward_std = Poor & loss_avg = Good) → mem_size = Large	Dynamically adjusts parameters based on agent performance

4.1 LunarLander-v2 environment

The goal of the LunarLander-v2 environment, which mimics a lunar lander, is to land softly on a pre-selected platform. The agent has four options for controlling its movement: main engine, left thruster, right thruster, or no action at all. The lander's position coordinates, vertical and horizontal velocities, angle, angular velocity, and leg contact with the ground are all included in the continuous state space of this environment [4, 12].

The following are the environment's primary challenges:

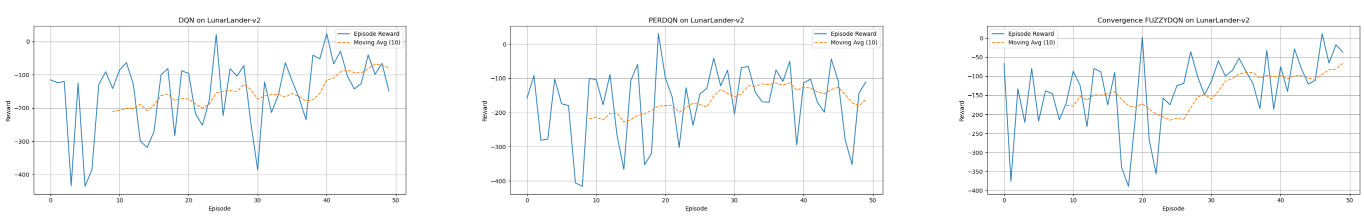
- Condition-sensitive and sparse rewards: Only successful landings receive positive rewards; unsuccessful landings receive negative rewards.
- Noisy dynamics: Minor motions can result in significant changes in trajectory; exact control and fuel economy must be balanced.

LunarLander-v2 is regarded as an appropriate benchmark to assess algorithms' capacity to manage uncertainty and adaptive tuning because of these factors. The three algorithms' average reward, standard deviation, and total training time in this setting are shown in Table 3.

Table 3: Average Reward, Standard Deviation, and Total Training Time of the Three Algorithms in LunarLander-v2

Method	Mean reward	STD	Total Training Time (s)
DQN	-156.61	117.65	45.62
PER-DQN	-154.86	93.23	117.99
FUZZY-DQN	-134.68	92.06	12.37

Figure 1 displays the results' convergence curves:



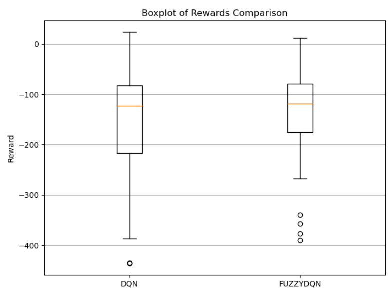
(a) (b) (c)

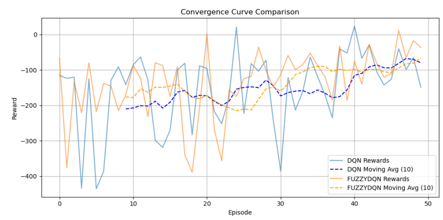
Figure 1: LunarLander-v2: Comparative convergence curves of DQN (a), PER-DQN (b), and FUZZY-DQN (c).

As can be seen, the suggested FUZZY-DQN approach considerably cuts down on training time while simultaneously increasing average reward when compared to baseline methods. The performance differences of FUZZY-DQN in comparison to the two baseline methods, however, are not statistically significant, according to the t-test and Wilcoxon statistical analyses (p-value > 0.05). Table 4 provides a summary of these findings, and Figs 2 and 3 display the comparative plots.

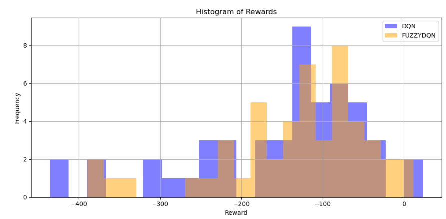
Table 4: Statistical test results in LunarLander-v2

LunarLander-v2	T-test		Wilcoxon Signed-Rank Test	
	t-statistic	p-value	w-statistic	p-value
DQN vs FUZZY-DQN	-0.8832	0.3793	523.0000	0.2737
PER-DQN vs FUZZY-DQN	1.6913	0.0939	437.0000	0.0530





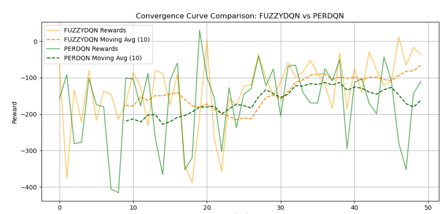
(b) Convergence Curve



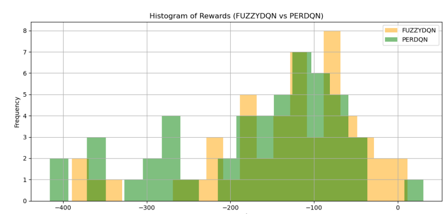
(c) Histogram

(a) Boxplot

Figure 2: LunarLander-v2: Comparison between DQN and FUZZY-DQN.



(b) Convergence Curve



(c) Histogram

(a) Boxplot

Figure 3: LunarLander-v2: Comparison between PER-DQN and FUZZY-DQN.

The results of the non-parametric Wilcoxon test and the t-test showed that there were no statistically significant differences in performance between FUZZY-DQN and the two baseline methods, DQN and PER-DQN, in the LunarLander-v2 environment (p-value > 0.05). This is mostly because even in cases where the average reward is higher, statistical significance cannot be established due to the high reward fluctuations and the noisy nature of this environment.

Boxplot analysis showed that while the data showed high variability and the interquartile range (IQR) was similar, the median reward of FUZZY-DQN was higher than that of both baseline methods. FUZZY-DQN produced more episodes with rewards above -100, according to the reward distribution histograms. Additionally, the convergence curves show that the suggested approach showed fewer fluctuations in the later phases of training and achieved stable performance more quickly than the other approaches.

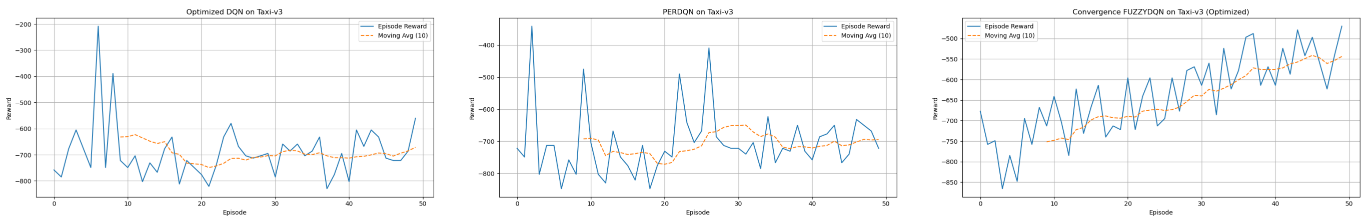
4.2 Taxi-V3 environment

A passenger must be picked up from a predetermined location and dropped off at a predetermined destination in the Taxi-v3 environment, which is a discrete navigation problem. There are 500 possible states in the state space, and there are six legitimate actions in the action space (moving in four directions, picking up, and dropping off) [18, 25]. Managing a large state space, which calls for a variety of skills and more general policies, the existence of numerous negative rewards for improper or unlawful moves, and the requirement to find the shortest path with the fewest possible penalties are some of the difficulties presented by this environment.

Table 5: Average reward, standard deviation, and total training time of three algorithms in Taxi-V3

Method	**Mean reward**	**STD**	**Total Training Time**
DQN	-690.52	104.51	345.64
PER-DQN	-705.02	100.07	3672.60
FUZZY-DQN	-640.64	94.67	356.89

Figure 4 displays the results' convergence plots.



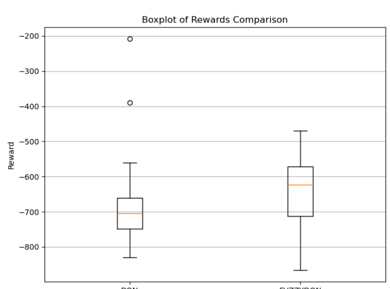
(a) DQN (b) PER-DQN (c) FUZZY-DQN

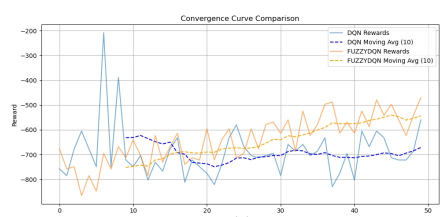
Figure 4: Taxi-v3: Comparative convergence curves of DQN (a), PER-DQN (b), and FUZZY-DQN (c).

In contrast to the LunarLander environment, the performance difference between FUZZY-DQN and the two baseline methods is statistically significant in this setting (p-value < 0.05). Furthermore, a significant optimization advantage of FUZZY-DQN is its significantly shorter training time compared to PER-DQN. Table 6 displays the findings of the statistical tests conducted in Taxi-V3, while Figures 5 and 6 display the comparative plots.

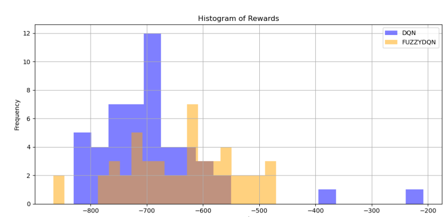
Table 6: Results of statistical tests in Taxi-V3

Comparison	**t-statistic**	**p-value**	**w-statistic**	**p-value**
DQN vs FUZZY-DQN	-2.4899	0.0145	249.0000	0.0005
PER-DQN vs FUZZY-DQN	3.2890	0.0014	241.0000	0.0004





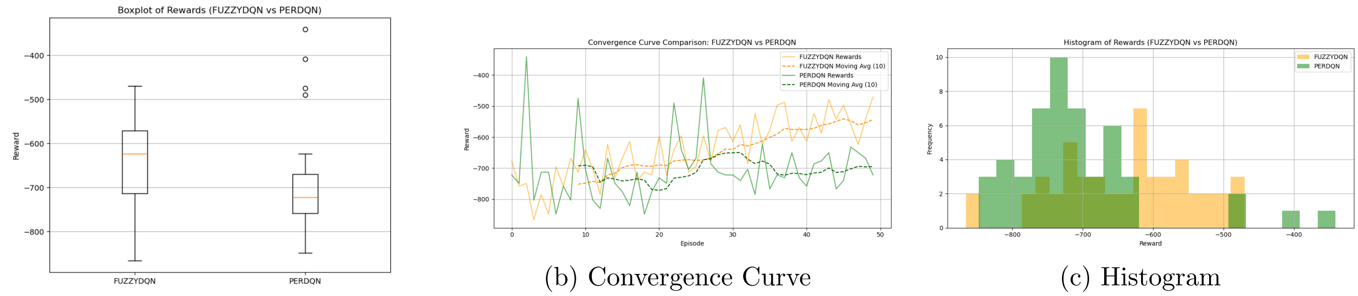
(b) Convergence Curve



(c) Histogram

(a) Boxplot

Figure 5: Taxi-v3: Rewards comparison between DQN (a), and FUZZY-DQN (b,c) shown as Boxplot (a), Convergence Curve (b), and Histogram (c).



(a) Boxplot

(b) Convergence Curve

(c) Histogram

Figure 6: Taxi-v3: Rewards comparison between PER-DQN (a), and FUZZY-DQN (b,c) shown as Boxplot (a), Convergence Curve (b), and Histogram (c).

FUZZY-DQN significantly outperforms both baseline methods in the Taxi-v3 environment, according to the results of the t-test and the Wilcoxon test (p-value significantly less than 0.05). This statistical significance suggests that adaptive replay memory control has a significant and real impact in this large-state-space environment. The boxplots demonstrate that FUZZY-DQN performs more steadily than the other two methods, with a significantly higher median reward and a smaller interquartile range (IQR). The histograms show that while the reward distributions of DQN and PER-DQN are primarily concentrated in the lower negative range, the reward distribution of FUZZY-DQN is shifted toward higher values. The convergence plot shows that compared to the other approaches, FUZZY-DQN achieves a stable reward range sooner and advances with a more positive and steadier slope.

4.3 CartPole-v1 environment

One of the traditional reinforcement learning problems is the CartPole-v1 environment, where the agent has to maintain the pole's equilibrium on a moving cart. The cart's position, velocity, pole angle, and angular velocity are all part of the state space. Force can be applied to the left or the right by the agent [24, 5]. The following are obstacles:

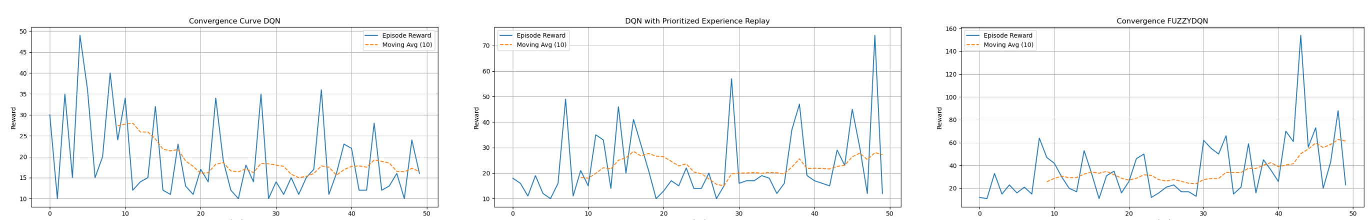
- Quick but constrained rewards (a favorable reward for every balanced step).
- The requirement for quick responses to even slight variations in the pole angle.
- Early learning sensitivity to replay memory settings.

The methods' performance in the CartPole-v1 environment is summed up in Table 7.

Table 7: Average reward, standard deviation, and total training time of three algorithms in CartPole-v1

Method	Mean reward	STD	Total Training Time
DQN	19.36	9.55	43.40
PER-DQN	22.74	13.66	75.80
FUZZY-DQN	36.50	25.84	77.09

Figure 7 displays the suggested method's convergence plot in this setting.



(a) DQN

(b) PER-DQN

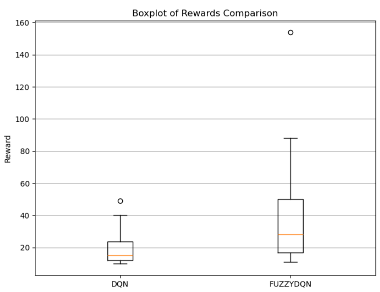
(c) FUZZY-DQN

Figure 7: CartPole-v1: Comparative convergence curves of DQN (a), PER-DQN (b), and FUZZY-DQN (c).

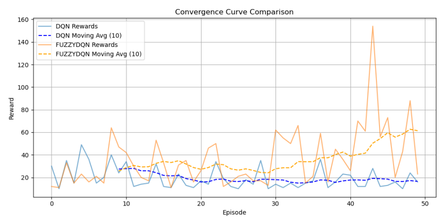
The statistical tests show that FUZZY-DQN performs significantly better than the baseline methods in this environment as well (p-value $\ll 0.05$). Figures 8 and 9 display the comparative plots, while Table 8 presents the findings of the statistical tests in CartPole-v1.

Table 8: Results of statistical tests in CartPole-v1

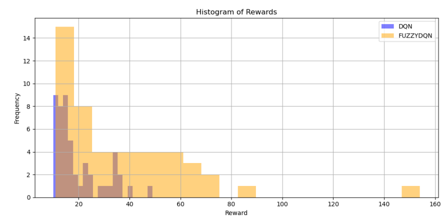
	T-test		Wilcoxon Signed-Rank Test	
	t-statistic	p-value	w-statistic	p-value
DQN vs FUZZY-DQN	-4.3545	0.0000	141.5000	0.0000
PER-DQN vs FUZZY-DQN	3.2953	0.0014	253.0000	0.0003



(a) Boxplot

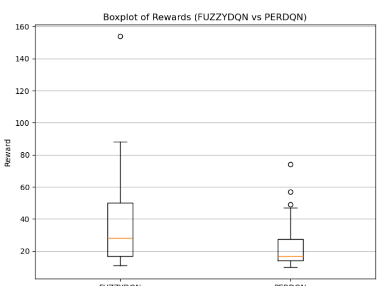


(b) Convergence Curve

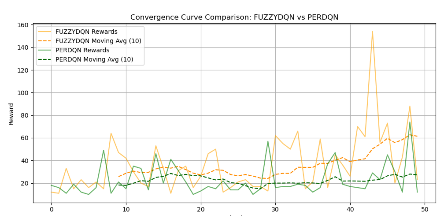


(c) Histogram

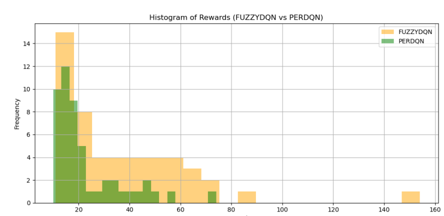
Figure 8: CartPole-v1: Rewards comparison between DQN and FUZZY-DQN (a: Boxplot, b: Convergence Curve, c: Histogram).



(a) Boxplot



(b) Convergence Curve



(c) Histogram

Figure 9: CartPole-v1: Rewards comparison between PER-DQN and FUZZY-DQN (a: Boxplot, b: Convergence Curve, c: Histogram).

Both the t-test and the Wilcoxon test indicated that the performance difference between FUZZY-DQN and the other methods was highly significant in the CartPole-v1 environment ($p < 0.001$). These findings show that the suggested approach significantly and steadily improves in a setting with simpler dynamics but high sensitivity to quick reactions. Boxplot analysis revealed that FUZZY-DQN's median reward is substantially higher and its range of variation is wider than the other methods', suggesting that the method can achieve very high performance in a variety of episodes. The histograms show that while the distributions of the two other methods are concentrated at lower values, the reward distribution of FUZZY-DQN is obviously shifted toward the maximum value (500). According to the convergence plot, FUZZY-DQN grows more quickly in the early training phases, reaching high reward values fast, and then showing less fluctuation.

The experimental findings in the three distinct settings show how the suggested FUZZY-DQN approach improves reinforcement learning performance. Specifically:

- When compared to the two baseline approaches, FUZZY-DQN produced a statistically significant increase in average reward in the Taxi-V3 and CartPole-v1 environments.

- Although there was a numerical improvement in the LunarLander-v2 environment, no statistically significant difference was discovered; nonetheless, the significant training time reduction is a noteworthy benefit.
- The suggested method's stability and enhanced learning behavior are further demonstrated by reward distribution histograms, boxplots, and convergence plots.
- One significant benefit for real-world applications is the training time reduction, particularly when compared to PER-DQN.

These findings offer compelling proof of the efficacy of fuzzy logic-based adaptive replay memory tuning in enhancing the stability and efficiency of DQN agent training.

5 Discussion and analysis

The suggested FUZZY-DQN approach performs better than DQN and PER-DQN in the majority of situations, according to the results from the three test environments. In the Taxi-v3 and CartPole-v1 environments, this superiority is especially statistically significant, demonstrating the genuine and repeatable impact of the suggested method. The adaptive tuning of the replay memory is the cause of this improvement. FUZZY-DQN uses a fuzzy logic system to adaptively adjust three key parameters (memory size, recent sample ratio, and priority weight) based on dynamic agent performance metrics, in contrast to DQN, which uses a fixed replay memory and sampling parameters, and PER-DQN, which only modifies prioritization.

This adaptive mechanism helps the agent to more reliably reproduce successful trajectories and optimally integrate recent and prior experiences in the Taxi-v3 environment, which is characterized by a large state space and low-frequency reward feedback. This feature raises the median reward in the boxplots and decreases performance variability (lower STD). Continuous updating of the recent sample ratio allows the agent to maximize new experiences, which speeds up convergence in CartPole-v1, a dynamic environment with instant rewards and the requirement for quick responses.

However, even though FUZZY-DQN outperforms other approaches in the LunarLander-v2 environment, which features complicated dynamics and noisy reward feedback, the differences are not statistically significant ($p > 0.05$). High reward fluctuations and the environment's sensitivity to slight modifications in the agent's policy could be the cause of this, which would lower the statistical tests' discriminative power. The suggested approach, however, achieves relative stability more quickly than DQN and PER-DQN, according to the convergence plot. Fuzzy logic and replay memory integration offers benefits like lowering sampling bias, broadening the variety of training data, and enhancing the harmony between exploration and exploitation, according to reinforcement learning theory. The results of this study show that adaptive modifications in memory size and composition can result in notable improvements even in situations where PER produces only slight gains, which is in line with earlier research on the efficacy of experience prioritization in PER.

The suggested algorithm can more effectively identify experiences that have the greatest impact on learning while balancing the use of both new and old samples thanks to this adaptive approach. Comparing these fuzzy settings to the two baseline algorithms, empirical results show a significant improvement in performance metrics. The fuzzy logic mechanism for data management and sampling alone improves learning quality; this improvement is noteworthy even when other important algorithm parameters stay constant. Thus, it is possible to draw the conclusion that using fuzzy logic to control and manage data parameters is essential for enhancing reinforcement learning performance. This is because it allows for dynamic and automatic adjustments, which make the training process more stable and efficient.

Nevertheless, the suggested approach has drawbacks. Designing the fuzzy system rules requires prior knowledge and trial-and-error, and in rapidly changing environments, the response may not be timely enough. Higher computational costs may also result from adding more fuzzy system input variables for more complex environments. However, training times in every environment show that this approach is still computationally competitive in real-world applications.

6 Conclusion and future work

This study proposed a novel fuzzy logic-based method for adaptively tuning replay memory parameters in Deep Q-Network agents. The suggested method enhances reinforcement learning performance and decreases training time by intelligently adjusting memory size, recent sample ratio, and priority weight, according to experimental results in three standard environments: LunarLander-v2, Taxi-v3, and CartPole-v1. In particular, statistically significant improvements over baseline methods were obtained in the Taxi-v3 and CartPole-v1 environments ($p < 0.05$), indicating the efficacy of the fuzzy logic system in optimally managing replay memory. Additionally, a key benefit for real-world applications is the notable decrease in training time when compared to PER-DQN.

Although there was a numerical improvement in the LunarLander-v2 environment, the difference was not statistically significant ($p > 0.05$), most likely because of the environment's greater complexity and wider reward fluctuations. This implies that creating more intricate fuzzy rules or combining the method with other optimization strategies may be necessary to achieve more accurate adjustments in such settings.

The following avenues for future research are recommended in light of these findings:

- Creating fuzzy rules through evolutionary optimization techniques or automated learning to improve adaptive tuning's precision and effectiveness.
- Using the suggested approach in more intricate, multi-agent settings to assess performance and generalization in more practical settings.
- Combining this strategy with more sophisticated DQN methods like Dueling DQN, Double DQN, and other algorithmic advancements.
- Examining how sensitive the parameters of a fuzzy system are and creating a framework for their automatic modification during training.

This study concludes by showing that fuzzy logic systems can be an effective tool for improving the adaptability and effectiveness of reinforcement learning algorithms, especially when it comes to replay memory management.

Acknowledgement

The authors wish to express their appreciation for several excellent suggestions for improvements in this paper made by the referees.

References

[1] F. E. Alsaadi, et al., *A new fuzzy reinforcement learning method for effective chemotherapy*, Mathematics, **11**(2) (2023), 477. https://doi.org/10.3390/math11020477

[2] M. Annabestani, et al., *A new soft computing method for integration of expert's knowledge in reinforcement learning problems*, arXiv, (2021). https://doi.org/10.48550/arXiv.2106.07088

[3] C. D'Eramo, et al., *Sharing knowledge in multi-task deep reinforcement learning*, arXiv, (2024). https://doi.org/10.48550/arXiv.2401.09561

[4] X. C. Han, et al., *Attention ensemble mixture: A novel offline reinforcement learning algorithm for autonomous vehicles*, Applied Intelligence, **55**(6) (2025), 1-14. https://doi.org/10.1007/s10489-025-06403-7

[5] U. Hwang, H. T. Lim, S. Hong, *Tackling environment heterogeneity in federated reinforcement learning*, 2025 IEEE Conference on Artificial Intelligence (CAI), IEEE, (2025). https://doi.org/10.1109/CAI64502.2025.00221

[6] J. S. R. Jang, C. T. Sun, E. Mizutani, *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*, Upper Saddle River, NJ, USA: Prentice Hall, 1997. https://doi.org/10.1109/TAC.1997.633847

[7] A. A. Khater, M. Fekry, M. El Bardini, A. M. El Nagar, *Deep reinforcement learning based adaptive fuzzy control for electro hydraulic servo system*, Neural Computing and Applications, (2025). https://doi.org/10.1007/s00521-024-10741-x

[8] L. J. Lin, *Self-improving reactive agents based on reinforcement learning, planning and teaching*, Machine Learning, **8**(3-4) (1992), 293-321. https://doi.org/10.1007/BF00992699

[9] V. Mnih, et al., *Human-level control through deep reinforcement learning*, Nature, **518**(7540) (2015), 529-533. https://doi.org/10.1038/nature14236

[10] D. E. Neves, L. Ishitani, Z. K. G. P. Junior, *Advances and challenges in learning from experience replay*, Artificial Intelligence Review, **58**(2) (2024), 1-54. https://doi.org/10.1007/s10462-024-11062-0

[11] J. Parker-Holder, et al., *Automated reinforcement learning (autorl): A survey and open problems*, Journal of Artificial Intelligence Research, **74** (2022), 517-568. https://doi.org/10.48550/arXiv.2201.03916

[12] J. F. Pettit, et al., *Disco-dso: Coupling discrete and continuous optimization for efficient generative design in hybrid spaces*, Proceedings of the AAAI Conference on Artificial Intelligence, **39**(25) (2025). https://doi.org/10.48550/arXiv.2412.11051

[13] T. Schaul, et al., *Prioritized experience replay*, arXiv, (2015). https://doi.org/10.48550/arXiv.1511.05952

[14] T. Schaul, J. Quan, I. Antonoglou, D. Silver, *Prioritized experience replay*, Proceeding International Conference on Learning Representations (ICLR), arXive, (2016). https://doi.org/10.48550/arXiv.1511.05952

[15] R. S. Sutton, A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

[16] M. E. Taylor, P. Stone, *Transfer learning for reinforcement learning domains: A survey*, Journal of Machine Learning Research, **10** (2009), 1633-1685. https://www.jmlr.org/papers/volume10/taylor09a/taylor09a.pdf

[17] H. Van Hasselt, A. Guez, D. Silver, *Deep reinforcement learning with double Q-learning*, Proceedings of the AAAI Conference on Artificial Intelligence, (2016), 2094-2100. https://doi.org/10.48550/arXiv.1509.06461

[18] L. Wang, Y. Huang, *Hierarchical reinforcement learning with curriculum learning and subpolicy transfer in navigation environments*, International Conference on Intelligent Computing, Singapore: Springer Nature Singapore, (2025). https://doi.org/10.1007/978-981-96-9894-3_14

[19] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, N. De Freitas, *Dueling network architectures for deep reinforcement learning*, in Proceedings of the 33rd International Conference on Machine Learning (ICML), New York, NY, USA, (2016), 1995-2003. https://doi.org/10.48550/arXiv.1511.06581

[20] K. Young, B. Wang, M. E. Taylor, *Metatrace actor-critic: Online step-size tuning by meta-gradient descent for reinforcement learning control*, arXiv, (2018). https://doi.org/10.48550/arXiv.1805.04514

[21] L. A. Zadeh, *Fuzzy sets*, Information and Control, **8**(3) (1965), 338-353. https://doi.org/10.1016/S0019-9958(65)90241-X

[22] J. X. Zhan, et al., *Accelerating deep reinforcement learning with fuzzy logic rules*, International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Cham: Springer Nature Switzerland, (2023). https://doi.org/10.1007/978-3-031-36822-6_23

[23] T. Zhang, et al., *Dynamics-adaptive continual reinforcement learning via progressive contextualization*, IEEE Transactions on Neural Networks and Learning Systems, **35**(10) (2023), 14588-14602. https://doi.org/10.1109/TNNLS.2023.3280085

[24] B. Zhao, Z. Z. Liu, Z. Tang, *KAFQN: Kolmogorov-Arnold fuzzy-guided Q-network in reinforcement learning*, ICASSP 2025, IEEE, (2025). https://doi.org/10.1109/ICASSP49660.2025.10890744

[25] L. L. Zhen, *Value-based reinforcement learning*, in Artificial Intelligence for Engineers: Basics and Implementations, Cham: Springer Nature Switzerland, (2025), 337-355. https://doi.org/10.1007/978-3-031-75953-6_14