Fuzzy Ex-RL: Fuzzy Experience-Based Reinforcement Learning

Ali Ghandi[*] Saeed Bagheri shouraki[†]

Abstract

Reinforcement Learning has proven to be a robust approach for addressing sequential decision-making problems, finding significant applications in both academic research and industry. However, RL algorithms often struggle with adaptability when there are slight changes in environmental parameters. Transfer Learning offers a promising solution by utilizing prior knowledge to expedite and improve the RL learning process under such conditions.

To address these challenges, we introduce Fuzzy Ex-RL, an enhancement of the Ex-RL algorithm[7] that incorporates fuzzy models to improve transfer learning capabilities. Fuzzy Ex-RL is designed to handle the elasticity in patterns, thereby effectively mapping experiences with differing skewness. This method is showing significant improvements in both success rate and sample efficiency. Our results indicate that Fuzzy Ex-RL achieves approximately a 60% increase in success rate and improvement in sample efficiency compared to traditional RL methods. Moreover, in transfer learning scenarios, Fuzzy Ex-RL outperforms the original Ex-RL by about 25%.

Keywords: Reinforcement Learning, Transfer Learning, and Experience-based learning

1 Introduction

Reinforcement Learning (RL) has established itself as a robust and effective methodology for solving sequential decision-making problems. In this framework, an agent iteratively interacts with an environment, refining its performance through a process of trial and error. This ability to adapt has made RL invaluable across various academic and industrial applications, particularly where traditional approaches have fallen short.

Recent advancements have seen the integration of Deep Learning (DL) into RL tasks, forming a hybrid model that has gained considerable attention. DL addresses specific RL challenges, such as handling high-dimensional input spaces and improving scalability, by employing powerful function approximators. These approximators effectively distill high-dimensional data into more manageable low-dimensional representations, enhancing the RL agent's learning efficiency.

In practical applications of RL, environment models are usually unknown. Agents are often required to gather extensive interaction experiences before they can effectively utilize their understanding of the environment for improved performance. Issues like partial observability, sparse and delayed feedback, and high-dimensional observation and action spaces exacerbate this challenge, making sample acquisition both costly and potentially unsafe. This is particularly pertinent in high-stakes domains like autonomous driving and health informatics[11].

To address these challenges, leveraging prior knowledge to accelerate the learning process becomes crucial. This is where Transfer Learning (TL) comes into play. TL enables the use of external knowledge to speed up and enhance the learning process in RL, making it a significant area of research[17, 22]. By incorporating insights gained from previous tasks, TL can help overcome the limitations inherent in purely trial-based learning approaches, thereby improving overall efficiency and safety.

TL in RL, is more complex than supervised learning due to the various components of a Markov Decision Process (MDP). The source of knowledge and its target can differ greatly within an MDP. Additionally, expert knowledge can come in different forms and be transferred through various means, especially when using deep neural networks.

To make it more clear we represent an MDP with $\mathcal{M} = (\mu_0, S, A, \mathcal{T}, \gamma, R)$ the first three components Are the initial state, the state space, and the action space respectively. $\mathcal{T} : S \times A \times S \to \mathbb{R}$ is the transition probability, which specifies the probability of moving to state s' from state s when taking a specific action. $R : S \times A \times S \to \mathbb{R}$ is the reward distribution that follows the transition. An RL agent operates in an environment $\mathcal{M}$ by following a policy π, which indicates the probability of taking action a when in state s. The value function(V^π) measures how good it is to be in state s by estimating the expected rewards the agent can receive from state s, assuming it continues to follow policy π in $\mathcal{M}$. It is also possible to estimate the quality of an action in a specific state using the Q-function, based on the definition of the value function[18]. Thus, the goal in RL is to find an optimal policy that maximizes the discounted cumulative reward or value function within

[*]Department of electrical engineering, Sharif University of Technology

[†]Department of electrical engineering, Sharif University of Technology, `https://ee.sharif.edu/ bagheri-s/`

the state space.

$$Q^{\pi}_{\mathcal{M}}(s,a) = \mathbb{E}_{s\sim P(s,a,s')}\left(R(s,a,s') + \gamma V^{\pi}_{\mathcal{M}}(s')\right) \quad (1)$$

Using transfer learning, an agent aims to learn the optimal policy for a target domain by utilizing information from both the target domain ($\mathcal{D}_t$) and source domains ($\mathcal{D}_s$). Suppose $\mathcal{M}_s$ represents a single source domain chosen from a set of source MDPs($\boldsymbol{\mathcal{M}_s}$) and $\mathcal{M}_t$ denotes the target MDP. In the simplest case, where the set $\mathcal{M}_s$ contains only one element, Mt is the same as Ms. In a typical RL loop without TL, where no external information is available, $\mathcal{D}_s$ is null.

$$\pi^* = \arg\max_{\pi\sim\mathcal{D}_s,\mathcal{D}_t} \mathbb{E}_{s\sim\mu_0^t, a\sim\pi}\left[Q^{\pi}_{M}(s,a)\right] \quad (2)$$

2 Background

This study focuses on improving the approach to transfer learning in RL proposed by Ghandi et al.[7, 6] called Ex-RL. Before discussing any potential enhancements for Ex-RL, it is essential to review similar methods.

2.1 Reward Shaping

In TL utilizing a reward shaping approach, the sole modification to the MDP occurs within the reward function. Here, the agent learns a reward function $\mathcal{F}: S \times A \times S \to \mathbb{R}$ that incorporates prior knowledge to guide the agent toward beneficial states by assigning higher reward values.

$$\mathcal{M} = (\mu_0, S, A, \mathcal{T}, R) \to \mathcal{M}' = (\mu_0, S, A, \mathcal{T}, R+\mathcal{F}) \quad (3)$$

[14]introduced Potential-Based Reward Shaping (PBRS), which stands as one of the foundational techniques in the domain of reward shaping. In this approach, the reward shaping function $\mathcal{F}$ is defined as the disparity between two real-valued functions $\Phi : \mathcal{S} \to \mathcal{R}$.

$$F(s,s') = \gamma\Phi(s') - \Phi(s) \quad (4)$$

PBRS is both a necessary and sufficient condition to guarantee policy invariance within the target MDP. Specifically, the optimal Q-function in the $\mathcal{M}_t$, can be derived from that in the $\mathcal{M}_s$, by incorporating a subtracted $\Phi(s)$. In extended approaches[20], actions are considered similarly to the vanilla approach. The reward shaping function $\mathcal{F}$ is now defined by differentiating between potential functions that depend on both the state and action variables. This imposes a limitation on extended forms, primarily requiring on-policy learning.

$$F(s,a,s',a') = \gamma\Phi(s',a') - \Phi(s,a) \quad (5)$$

Several methodologies incorporating potential-based approaches, such as the Dynamic Potential-Based method[4], which integrates a time parameter, and the Dynamic Value-Function Advice approach[8], which are not the focus of this study.

Beyond those approaches, innovative strategies have been proposed to leverage reward shaping for transferring an expert policy from a source environment $\mathcal{M}_s$ to a target environment $\mathcal{M}_t$. In the study[1], the authors posit the existence of two mapping functions, M_A and M_S, which transform states and actions between two MDPs. By utilizing these mappings, each state and action in the $\mathcal{M}_t$, can be converted to their corresponding counterparts in the source domain. This conversion allows the authors to derive probability distributions of states and actions from the expert policy in the source domain. These derived probabilities can then be used to shape the reward structure in the $\mathcal{M}_t$.

All of the aforementioned approaches require that $\mathcal{M}_t$ and $\mathcal{M}_s$ be identical, except in the context of the reward function. However, [19] proposed a method that bypasses this prerequisite by utilizing π_s as prior knowledge. Notably, such prior knowledge is often not available.

2.2 Learning from Demonstrations

In this methodology, demonstrations are obtained from various sources and presented in the form of transition tuples:(s,a,s',r). These demonstrations may be derived from both optimal and near-optimal experts. Most research typically focuses on cases where the source and target MDPs are the same. Leveraging these demonstrations, an agent can initialize its value function or policy via offline methods. In contrast, within online settings, these demonstrations can be utilized to enhance the agent's exploration strategy.

Learning from demonstration has mostly been studied in the context of policy gradient methods. However, it can also be applied using policy iteration[3] or temporal-difference(TD) learning methods[21]. In the referenced study [3], the agent samples from self-generated data(D_π) and combines it with external demonstrations(D_E) provided by an expert policy. The agent estimates Q-values using a greedy algorithm. To ensure the policy π aligns closely with the expert's decisions, a loss function is used that increases whenever the agent's decisions differ from the expert's.

$$\mathcal{L}(\pi,\pi_E) = \frac{1}{N_E}\sum_{i=1}^{N_E}\{1_{\{\pi_E(s_i)\neq\pi(s_i)\}}\}. \quad (6)$$

[15]introduced the Deep Q-learning from Demonstration algorithm, which leverages dual replay buffers, designated as D_π and D_E, to manage the sampling ratio between agent experiences and expert demonstra-

tions. The D_E buffer employs a prioritized replay schema, wherein the sampling probability of a transition i is determined by its priority p_i. It's prioritize using $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$ which modulated by a temperature parameter α.

In another study [2], the author employed potential-based reward shaping methods in conjunction with learning from demonstrations. This approach defines a potential-based function over expert demonstrations, so that provides greater rewards for states and actions that resemble those of the expert demonstrations. This mechanism effectively induces behavior akin to that of an expert.

[13] introduced an imitation learning technique, inspired by demonstration learning, designed to clone expert behavior using a policy gradient algorithm. Their approach involves defining a loss function based on demonstration samples, which penalizes the model whenever it makes decisions that diverge from those of the expert. In its extended form, the method imposes a loss penalty exclusively when the critic assigns a lower value to the Q-function, as opposed to simply selecting actions consistent with the demonstration data.

$$\mathcal{L}_{BC} = \sum_{i=1}^{|D_E|} \left\| \pi(s_i|\theta_\pi) - a_i \right\|^2 \mathbb{1}\left[Q(s_i, a_i) > Q(s_i, \pi(s_i))\right] \tag{7}$$

In another method, Generative Adversarial Imitation Learning (GAIL)[10] employs a discriminator trained to differentiate between interactions sampled from the current policy and those from the expert policy. Each policy represents a distribution over state-action pairs. Minimizing the divergence between these distributions attempts to confuse the discriminator, making it indistinguishable whether an interaction originates from the agent or the expert. Consequently, the agent's policy, π, becomes more akin to the expert policy π_E. Given that the π_E is unknown, learning from demonstrations aids the agent in estimating the state-action distribution. Therefore, the agent endeavors to perform distribution matching in the following manner:

$$\begin{aligned} \max_{\pi} \min_{D \to (0,1)} J(\pi, D) :=& \\ &\mathbb{E}_{d_\pi}\left[\log\left(1 - D(s,a)\right)\right] + \mathbb{E}_{d_E}\left[\log\left(D(s,a)\right)\right] \\ &:= -D_{JS}[d_\pi \| d_E] \end{aligned} \tag{8}$$

Despite the wide range of methods available for learning from demonstrations, several challenges remain. Firstly, increasing the size of the demonstration dataset requires storing all expert data, which can be memory-intensive. Additionally, most approaches assume that demonstrations are near-optimal; however, in practice, demonstrations may be imperfect due to noise[9], biased estimations of the MDP, or suboptimal expert performance. Although techniques such as regularization and relaxed constraints have been proposed to address these issues, they often do not ensure policy invariance during the learning process. Moreover, agents typically rely on a limited set of demonstrated data compared to the diversity of self-generated samples, making them prone to overfitting. Agents also lack guidance for state-action pairs that are not included in D_E. These challenges are further exacerbated in environments with sparse reward feedback. During exploration, deviations from demonstrated trajectories and samples are common, especially in the early episodes, due to the inherent randomness in action selection.

2.3 Ex-RL: Experience-Based RL

[7] proposed a novel approach for TL in RL applications, Ex-RL, which synergistically integrates reward shaping with Learning from Demonstration techniques. Similar to demonstration-based methods, this technique utilizes data gathered from multiple near-optimal expert trajectories. However, rather than storing all transition tuples, a Hidden Markov Model (HMM) [12] is employed to identify patterns within these trajectories. Using Ex-RL obviates the need for memory expansion by continually adding new demonstrations, thereby optimizing computational efficiency.

The HMM subsequently functions as a critic for the new agent, incentivizing it to replicate the trajectories selected by the expert agents. This process is akin to reward shaping, wherein the HMM outputs the likelihood of multiple steps as an augmented reward, in combination with the environmental reward. Trajectories aligning closely with expert demonstrations are prioritized by reward function. In certain cases, the HMM behaves as a potential-based function, thus preserving policy invariance. Similar to existing reward shaping techniques, the proposed Ex-RL approach is model-agnostic. Authors have applied the same methodology across various reinforcement learning paradigms[??], including deep policy gradient methods and traditional methods like Temporal Difference or policy iteration learning. The HMM is especially good at dealing with imperfect examples because it is built on strong statistical methods. It can effectively reduce the impact of biases or errors that might be present in different sets of data. This makes HMM a valuable tool for maintaining the accuracy and reliability of the systems it models.

While previous studies often emphasize the $\mathcal{M}_t = \mathcal{M}_s$ condition, the Ex-RL framework overcomes this challenge through delta mapping. The authors introduce a delta mapping mechanism that divides any state space S into a set of super states. For example, in tasks like the ball-and-beam and cart-pole, the objective is to balance an object around a central line. Delta mapping breaks down the state space into four regions: moving forward or backward on either side of this balance line.

The ball-and-beam and cart-pole tasks, despite having distinct MDPs characterized by unique parameters such as action and state spaces, can be transformed into a unified task of object balancing through delta mapping. This approach involves the abstraction of numerical values into more generalized descriptions, thereby facilitating the learning and transfer of concepts across different MDPs by HMMs. While the agent learns the specific details of the environment, the HMM aids in navigating the more abstract objective of balancing the object, which in turn promotes more efficient exploration by the agent.

Despite Ex-RL addressing some challenges associated with reward shaping and demonstration methods, it still faces significant issues in certain applications. Consider the Pendulum and Mountain Car tasks as examples. In the Pendulum task, the agent aims to reach a target position by adjusting its momentum through forward and backward movements. Similarly, in the Mountain Car task, the agent must perform anti-balancing actions to climb the right hill. Although these two MDPs are fundamentally different, delta mapping theoretically allows for similar movement patterns, enabling the transfer of experience between them.

However, in practice, solving the Mountain Car task typically requires more steps due to the environment's physical dynamics, leading to a skewed behavior in the agent. In contrast, movements in the Pendulum task are analogous but occur much more rapidly. The HMM module fails to adequately account for temporal and speed variations in these patterns, further complicating effective transfer learning.

To address this issue, this paper focuses on enhancing the Ex-RL framework[6, 7] by mitigating the skewness in patterns for transfer learning. We propose a novel approach that incorporates a variation of the fuzzy elastic matching machine(FEMM)[16], which is designed to better handle differences in time and speed within movement patterns. This contribution aims to improve the robustness and effectiveness of transfer learning in Ex-RL, enabling more seamless experience transfer between disparate tasks.

3 Proposed Method

To address the problem of HMM in Ex-RL, Fuzzy Ex-RL uses a FEMM. FEMM is a fuzzy sequential pattern recognition tool that compares input data against a fuzzy elastic pattern. Consider a FEMM denoted as $\lambda = (R, P)$, where R and P are fuzzy vectors representing the duration(Run time) and features of patterns within a super state. A super state is defined as an array of observations.

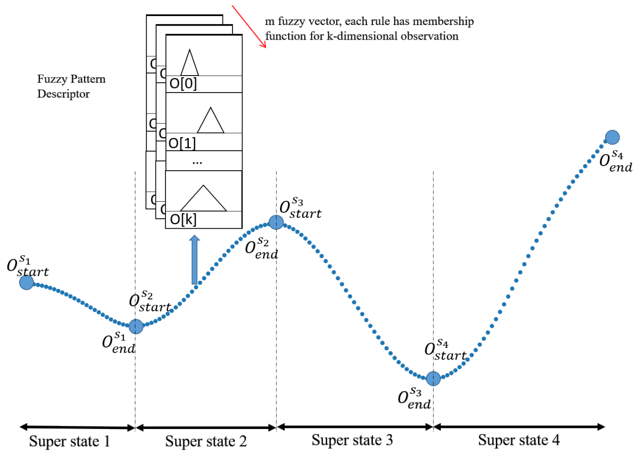

Figure 1: The completed pathway of the mountain car task is segmented using change points, with each segment forming a "super state" that encompasses specific observations. These super states are characterized by fuzzy pattern descriptors. Each dot in the figures represents an observation, where the Y-axis indicates the car's position.

An agent's observation may have k dimensions. For each dimension, m fuzzy descriptors can be employed, typically using the k-means method. Each pattern descriptor consists of k membership functions, which characterize each dimension of the observation. The term $P^l_{s_r}(O^n_i)$ denotes the membership level of the ith dimension of observation n, associated with the lth set of fuzzy membership functions relevant to the rth super state.

Let's assume that the centroid of cluster i is represented as c_i. We can then calculate the membership value to a pattern descriptor using equation 9.

$$\mu^x_{c_i} = \left(\sum_c \left(\frac{\|x - c_i\|}{\|x - c\|} \right)^{\frac{2}{m-1}} \right)^{-1} \tag{9}$$

The process of assigning a value to a k-dimensional fuzzy vector involves multiplying the membership values of each vector element. Subsequently, the highest membership value among all m vectors is selected. This value is calculated using the max-product composition method. In this compositional framework, the T-Norm functions as the multiplication operator, while the S-Norm serves as the addition operator. Equation 10 shows single observation membership value.

$$P_{s_i}(O^n) = \max_{j:1..m} \left(\prod_k P^j_{s_i}(O^n_k) \right) \tag{10}$$

A trajectory includes several observations made from different actions in an environment.The membership

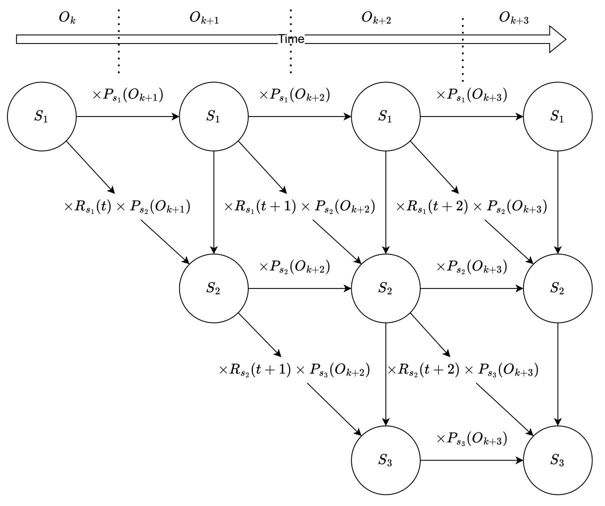
O_k
O_{k+1}
O_{k+2}
O_{k+3}
Time
$\times P_{s_1}(O_{k+1})$
$\times P_{s_1}(O_{k+2})$
$\times P_{s_1}(O_{k+3})$
S_1
$\times R_{s_1}(t) \times P_{s_2}(O_{k+1})$
$\times R_{s_1}(t+1) \times P_{s_2}(O_{k+2})$
$\times R_{s_1}(t+2) \times P_{s_2}(O_{k+3})$
S_2
$\times P_{s_2}(O_{k+2})$
$\times P_{s_2}(O_{k+3})$
$\times R_{s_2}(t+1) \times P_{s_3}(O_{k+2})$
$\times R_{s_2}(t+2) \times P_{s_3}(O_{k+3})$
S_3
$\times P_{s_3}(O_{k+3})$

Figure 2: FEMM state membership value analysis, transitions between different rows indicate a change in state, while movement within the same row signifies the state remains unchanged

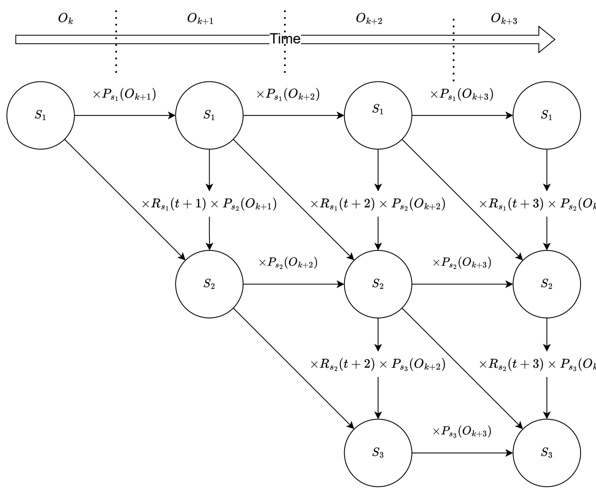
O_k
O_{k+1}
O_{k+2}
O_{k+3}
Time
$\times P_{s_1}(O_{k+1})$
$\times P_{s_1}(O_{k+2})$
$\times P_{s_1}(O_{k+3})$
S_1
$\times R_{s_1}(t+1) \times P_{s_2}(O_{k+1})$
$\times R_{s_1}(t+2) \times P_{s_2}(O_{k+2})$
$\times R_{s_1}(t+3) \times P_{s_2}(O_{k+3})$
S_2
$\times P_{s_2}(O_{k+2})$
$\times P_{s_2}(O_{k+3})$
$\times R_{s_2}(t+2) \times P_{s_3}(O_{k+2})$
$\times R_{s_2}(t+3) \times P_{s_3}(O_{k+3})$
S_3
$\times P_{s_3}(O_{k+3})$

Figure 3: The analysis of FEMM state membership values, where a single observation can belong to multiple states, is represented by a downward-sloping line

function for a trajectory is found by multiplying the membership values of each individual observation. Each trajectory has a duration, which should be represented by D, as described in the original study.

$$P_{s_i}\left(O^{t_1}, \ldots, O^{t_n}\right) = D_i(n) \prod_{t=t_1}^{t_n} P_{s_i}\left(O^t\right) \tag{11}$$

Figure 1 presents a trajectory of the Mountain Car, with the x-axis representing the steps and the y-axis depicting the car's position within the environment. Each scatter dot corresponds to a specific observation. Additionally, the membership functions for each super state are illustrated in the figure.

The FEMM employs a Viterbi-like algorithm[5] on a three-dimensional V matrix to determine the maximum membership value of a sequence. A higher membership value indicates a greater likelihood that the trajectory of observations originates from previously learned fuzzy patterns. In contrast, HMMs utilize probabilities to assess the likelihood that observations derive from previously learned patterns. By considering the duration in each state, FEMM is more suitable for learning elasticity in data. The $V(s,k,t)$ matrix records the maximum membership degree of a sequence of observations $(O_1, ..., O_k)$ to an uninterrupted path from the initial state to state s.

When using the Finite Element Method Magnetics (FEMM), three types of events can occur. First, the subsequent observation may belong to the current super state. In this case, the V value for the path is updated by incorporating the membership value of the observation, as shown in Equation 12. Second, if the subsequent observation belongs to the next super state, the duration model for the current state is considered, and the value is updated using the next membership value, as described in Equation 13. These two scenarios are depicted in Figure 2 using a stateful schema.

Occasionally, in fuzzy models, an observation may belong to two or more super states based on learned fuzzy patterns. In such scenarios, all membership values of the states are taken into account, and the duration is calculated up to the current state, as illustrated in Equation 14. Figure 3 demonstrates this condition.

$$V(s,k+1,t+1) = V(s,k,t) \times P_s\left(O^{k+1}\right) \tag{12}$$

$$V(s+1,k+1,1) = V(s,k,t) \times R_s(t) \times P_{s+1}\left(O^{k+1}\right) \tag{13}$$

$$\begin{aligned} V(s+1,k+1,1) = &V(s,k,t) \times P_s\left(O^{k+1}\right) \\ &\times R_s(t+1) \times P_{s+1}\left(O^{k+1}\right) \end{aligned} \tag{14}$$

By applying these equations, we can utilize the FEMM in place of the HMM.

3.1 Fuzzy Experience based RL

To elucidate the concept of a "superstate," we can define it as a subsequence of similar observations that adhere to common patterns. Consider, for example, a car in a Mountain Car environment. When the car initiates forward movement, the actions and intensity of the agent remain consistent, performing all necessary actions to maximize forward progress. This general behavior can

Algorithm 1 Finding Dense Points (Change Points)

```
Require: window length $w$
Ensure: List of change points $CP$
1: $density\_list = \{\}$
2: for $obs \in$ observations$[w/2 : \text{full\_length} - w/2]$ do
3:     $min\_obs = \min(\text{observations}[obs - w/2, obs + w/2])$
4:     $max\_obs = \max(\text{observations}[obs - w/2, obs + w/2])$
5:     $density\_list.append((obs, max\_obs - min\_obs))$
6: end for
7: for $index, (obs, density) \in \text{enumerate}(density\_list)$ do
8:     if $density \leq \min(\text{density\_list}[index - w/2, index + w/2])$ then
9:         $CP.append(obs)$
10:    end if
11: end for
12: return $CP$
```

be characterized as moving to the right through multiple steps. At certain positions, the agent changes direction upon approaching boundaries. At these boundary lines, the car's velocity decreases, prompting a change in direction.

When observations are sampled at equal intervals, there is an increased density of observations in specific areas. In the Mountain Car scenario, the car's velocity decreases each time it reaches either side, resulting in changes in position. Consequently, uniformly sampling the position leads to dense observations at these boundary locations. We refer to these points as Change Points (CP). To identify these points, we can slide a window over time and monitor the difference between the extrema of observations as a density metric. If the density within the window is minimal, we designate the point as a Change Point. The algorithm outlined in Algorithm 1 defines this process.

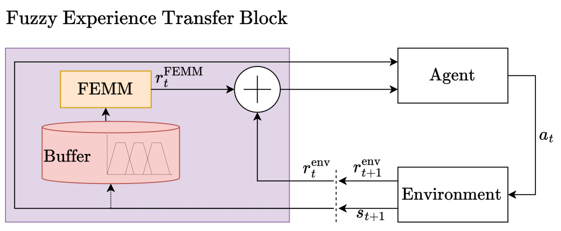

Figure 4: Incorporating a Fuzzy Experience Transfer Block into the RL loop introduces a model-agnostic system that evaluates the agent's experiences using external rewards.

By employing superstates, we can evaluate mini trajectories using the FEMM. The algorithm is similar to Ex-RL; however, the evaluation metric differs as it uses the value of a membership function instead of log-likelihood. Since all values fall within the range of zero to one, normalization parameters are unnecessary. To manage the numerical effects of FEMM, a control parameter is utilized.

Algorithm 2 Fuzzy Ex-RL

```
Require: $\tau \leftarrow$ Double-ended queue with the size of $t$
1: while Playing do
2:     $S \leftarrow$ Environment's current state
3:     $A \leftarrow \pi(S)$
4:     $S', R \leftarrow$ Take Action $A$ at state $S$
5:     Store $(S, A, R, S')$ in $\tau$
6:     if length$(\tau) = t$ then
7:         $R_{FSPR_\tau} \leftarrow FSPR(\tau) \times \alpha$
8:         for $(S_i, A_i, R_i, S'_i) \in reverse(\tau)$ do
9:             $R_{new} \leftarrow (R_{FSPR} + R_i)\lambda^i$
10:            store $(S_i, A_i, R_{new}, S'_i) inBuffer$
11:        end for
12:    end if
13:    Agent learn with new rewards
14: end while
```

Fuzzy-based algorithms use simple, human-driven rules, which simplifies task conditions since human task descriptions generally avoid intricate rules. For instance, an autonomous vehicle, which involves high-dimensional tasks, can employ fuzzy-based algorithms for maintaining lane balance as a high-level task. In the context of FEMM, calculations mainly depend on lookup tables, enhancing efficiency on hardware that supports fuzzy logic. However, this method may encounter challenges when applied to hardware that is optimized for crisp logic.

4 Experiment and Results

To conduct the experiment, we employed specific hardware and software configurations. The models were developed using Python 3.10 on the Ubuntu 22.04.4 LTS operating system. Evaluations were carried out on a system featuring 16 GB of RAM and an Intel i7-9750H CPU. Each experiment involved running one hundred agents, and the results were averaged across these runs. All environmental variables were kept constant to ensure a fair comparison between the Fuzzy Ex-RL method and the Ex-RL method.

We utilized four environments to compare the methods: Mountain Car, Pendulum, Cart Pole, and Ball and Beam. The first two environments, MC and PP, involve tasks where the agents aim to escape from equilibrium, while the latter two, CP and BB, focus on balancing an object in an abstract setting. The Success Rate (SR) quantifies the proportion of instances in which agents successfully complete the given task across all experiments. The Mean Episode (ME) indicates the average number of episodes required for all agents to converge on a solution. Additionally, the Execution Time per Episode (Tep) measures the average duration an agent takes to complete a single episode. The "c" character is added when changes based on a specific reference are indicated.

Table 1: Performance of Fuzzy Ex-RL: success rate and sample efficiency in four control environments using three RL algorithms.

Environment	Alg	SR	Sample Efficiency
CartPole	PPO	100%	+23%
	A2C	65%	+12%
	Q-learning	100%	+63%
Pendulum	PPO	64%	+11%
	A2C	57%	+18%
	Q-learning	100%	+37%
Ball & Beam	PPO	72%	+35%
	A2C	69%	+16%
	Q-learning	100%	+84%
MountainCar	PPO	45%	+16%
	A2C	42%	+21%
	Q-learning	60%	+87%

In Table 1, the proposed method is applied to three different RL algorithms. Fuzzy Ex-RL, like Ex-RL, is model-agnostic. The method is adaptable to any RL agent, including advanced versions of vanilla algorithms, because it critiques agent behavior based on experiential data rather than requiring complex modifications. The success rate (SR) and sample efficiency of these methods are compared to scenarios where pure RL is used without any experience guidance. Sample efficiency is defined as the number of episodes required by the agent to achieve the learning goal. Therefore, an increase in sample efficiency indicates that fewer episodes are needed.The sample efficiency of the proposed method is compared to that of the standard RL algorithm, in scenarios where the tasks are solvable by both approaches.

Table 2 compares Fuzzy Ex-RL with Ex-RL. Fuzzy Ex-RL, by understanding the elasticity in patterns, can map even extreme experiences that have differences in the skewness of patterns. Consider a scenario involving a mountain car with a steep slope that seeks to benefit from an experience involving a similar but much gentler slope. In the gentler scenario, the agent spends more time climbing the hill with lower forces. If an HMM had only encountered this gentler scenario, it would consistently penalize a new agent for spending less time on the right part of the hill. However, the new agent requires more force over a shorter distance to manage the steep slope. Although both patterns indicate forward movement, the difference in elasticity leads to faults in the learning process.

Table 2: Comparison of Fuzzy Ex-RL and Standard Ex-RL Performance

Environment	Alg	SRc	MEc	Tepc
CartPole	PPO	0%	+16%	+7%
	A2C	+7%	+14%	+8%
	Q-learning	0%	+17%	+11%
Pendulum	PPO	+7%	+12%	+10%
	A2C	+9%	+11%	+9%
	Q-learning	+14%	+16%	+13%
Ball & Beam	PPO	+9%	+4%	+5%
	A2C	+8%	+5%	+5%
	Q-learning	0%	+11%	+10%
MountainCar	PPO	+17%	+16%	+12%
	A2C	+22%	+17%	+11%
	Q-learning	+27%	+19%	+13%

The primary objective of leveraging experience in the learning process is to facilitate transfer learning. Table 3 illustrates two scenarios: transferring experience from the Mountain Car to the Pendulum, and from the Cart Pole to the Ball and Beam. The changes in results are calculated in comparison to Ex-RL. To address the issue of Ex-RL, consider the first scenario. Both agents in the Mountain Car and Pendulum tasks attempt to escape from equilibrium by employing forward and backward movement patterns. When plotting the positions of the agents, both exhibit bowl-shaped movement patterns to achieve their goals. Consequently, the abstract behavior should be transferable. However, the Mountain Car agent requires approximately 180 steps to reach the goal due to its physical parameters, whereas the Pendulum agent needs only around 70 episodes. Although the movement patterns are similar, the elasticity within these patterns presents a challenge. The HMM statistically learns the pattern without regard to the duration of the patterns, whereas Fuzzy Ex-RL effectively addresses this issue.

Based on the analysis of these three groups of results, it appears that substituting the HMM with an elastic pattern recognition model may be advantageous. In this study, we employ the FEMM due to its mathematical simplicity and its similarity to HMM. Both FEMM and

HMM offer significant benefits compared to traditional reward shaping or learning from demonstration methods.

Table 3: Performance outcomes of transferring experience from Mountain Car to Pendulum and from Cart-Pole to Ball and Beam, compared to standard Ex-RL.

Env	Alg	SR	SRc	MEc
Pendulum	PPO	53%	+32%	+6%
	A2C	49%	+36%	+2%
	Q-learning	100%	0%	+9%
Ball & Beam	PPO	63%	+26%	+3%
	A2C	59%	+31%	+1%
	Q-learning	100%	0%	+8%

5 Conclusion

This study aims to enhance transfer learning within RL algorithms. Previously, the Ex-RL algorithm demonstrated certain advantages over methods such as pure reward shaping and learning with demonstration, primarily due to its model-agnostic nature and its ability to function without requiring the MDPs to be identical. However, this research seeks to address a significant limitation within the Ex-RL algorithm to improve learning efficiency and transfer learning capabilities by incorporating fuzzy models.

The primary issue identified was the algorithm's inability to account for elasticity in patterns, which led to failures in certain scenarios. Our findings revealed an approximately 60% increase in success rate and a 20% improvement in sample efficiency. Furthermore, in the context of transfer learning, the Fuzzy Ex-RL algorithm outperformed the plain model by 25%.

All results and improvements are comprehensively detailed in Tables 1 to 3. These findings underscore the potential of integrating fuzzy models into RL algorithms to enhance both learning efficiency and transfer learning performance.

References

[1] Tim Brys et al. "Policy Transfer using Reward Shaping." In: *AAMAS.* 2015, pp. 181–188.

[2] Tim Brys et al. "Reinforcement learning from demonstration through shaping". In: *Twenty-fourth international joint conference on artificial intelligence.* 2015.

[3] Jessica Chemali and Alessandro Lazaric. "Direct policy iteration with demonstrations". In: *IJCAI-24th International Joint Conference on Artificial Intelligence.* 2015.

[4] Sam Michael Devlin and Daniel Kudenko. "Dynamic potential-based reward shaping". In: *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012).* IFAAMAS. 2012, pp. 433–440.

[5] M Sami Fadali. "Hidden Markov Models". In: *Introduction to Random Signals, Estimation Theory, and Kalman Filtering.* Springer, 2024, pp. 399–423.

[6] Ali Ghandi, Saeed Bagheri Shouraki, and Mahyar Riazati. "Deep ExRL: Experience-Driven Deep Reinforcement Learning in Control Problems". In: *2024 12th Iran Workshop on Communication and Information Theory (IWCIT).* IEEE. 2024, pp. 1–6.

[7] Ali Ghandi et al. "Ex-RL: Experience-Based Reinforcement Learning". In: *Information Sciences* (2024), p. 121479.

[8] Anna Harutyunyan et al. "Expressing arbitrary reward functions as potential-based advice". In: *Proceedings of the AAAI conference on artificial intelligence.* Vol. 29. 1. 2015.

[9] Mingxuan Jing et al. "Reinforcement learning from imperfect demonstrations under soft expert guidance". In: *Proceedings of the AAAI conference on artificial intelligence.* Vol. 34. 04. 2020, pp. 5109–5116.

[10] Bingyi Kang, Zequn Jie, and Jiashi Feng. "Policy optimization with demonstrations". In: *International conference on machine learning.* PMLR. 2018, pp. 2469–2478.

[11] Ezgi Korkmaz. "A Survey Analyzing Generalization in Deep Reinforcement Learning". In: *arXiv preprint arXiv:2401.02349* (2024).

[12] B Manjunatha et al. "Theoretical Foundations and Application of Hidden Markov Models". In: *Journal of Scientific Research and Reports* 30.8 (2024), pp. 837–849.

[13] Ashvin Nair et al. "Overcoming exploration in reinforcement learning with demonstrations". In: *2018 IEEE international conference on robotics and automation (ICRA).* IEEE. 2018, pp. 6292–6299.

[14] Andrew Y Ng, Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: 99 (1999), pp. 278–287.

[15] Tom Schaul. “Prioritized Experience Replay”. In: *arXiv preprint arXiv:1511.05952* (2015).

[16] Sina Shahmoradi and Saeed Bagheri Shouraki. “Evaluation of a novel fuzzy sequential pattern recognition tool (fuzzy elastic matching machine) and its applications in speech and handwriting recognition”. In: *Applied Soft Computing* 62 (2018), pp. 315–327.

[17] Gleice Kelly Barbosa Souza et al. “Transfer reinforcement learning for combinatorial optimization problems”. In: *Algorithms* 17.2 (2024), p. 87.

[18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[19] Mel Vecerik et al. “Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards”. In: *arXiv preprint arXiv:1707.08817* (2017).

[20] Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. “Principled methods for advising reinforcement learning agents”. In: *Proceedings of the 20th international conference on machine learning (ICML-03)*. 2003, pp. 792–799.

[21] Zhuangdi Zhu et al. “Learning sparse rewarded tasks from sub-optimal demonstrations”. In: *arXiv preprint arXiv:2004.00530* (2020).

[22] Zhuangdi Zhu et al. “Transfer learning in deep reinforcement learning: A survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).