Zander *et al. Complex Eng Syst* 2023;3:9
DOI: 10.20517/ces.2023.11

Complex Engineering Systems

Research Article

Open Access

Reinforcement learning with Takagi-Sugeno-Kang fuzzy systems

Eric Zander, Ben van Oostendorp, Barnabas Bede

DigiPen Institute of Technology, Redmond, WA 98012, USA.

Correspondence to: Eric Zander, DigiPen Institute of Technology, 9931 Willows Rd. NE.Redmond, WA 98012, USA.
E-mail: eric.zander@digipen.edu

How to cite this article: Zander E, van Oostendorp B, Bede B. Reinforcement learning with Takagi-Sugeno-Kang fuzzy systems. *Complex Eng Syst* 2023;3:9. http://dx.doi.org/10.20517/ces.2023.11

Received: 17 Mar 2023 **First Decision:** 12 Apr 2023 **Revised:** 16 May 2023 **Accepted:** 19 May 2023 **Published:** 14 Jun 2023

Academic Editor: Hamid Reza Karimi **Copy Editor:** Fanglin Lan **Production Editor:** Fanglin Lan

Abstract

We propose reinforcement learning (RL) architectures for producing performant Takagi-Sugeno-Kang (TSK) fuzzy systems. The first employs an actor-critic algorithm to optimize existing TSK systems. An evaluation of this approach with respect to the Explainable Fuzzy Challenge (XFC) 2022 is given. A second proposed system applies Deep Q-Learning Network (DQN) principles to the Adaptive Network-based Fuzzy Inference System (ANFIS). This approach is evaluated in the CartPole environment and demonstrates comparability to the performance of traditional DQN. In both applications, TSK systems optimized via RL performed well in testing. Moreover, the given discussion and experimental results highlight the value of exploring the intersection of RL and fuzzy logic in producing explainable systems.

Keywords: Explainable AI, Fuzzy systems, Takagi-Sugeno-Kang fuzzy systems, Adaptive neuro-fuzzy inference systems, Reinforcement learning

1. INTRODUCTION

Fuzzy sets have been introduced in [1] as a mathematical framework for modeling under uncertainty. Fuzzy systems employ a fuzzy inference engine to solve control problems in various frameworks and applications [2,3]. In a majority of fuzzy control applications, a dynamic model joins a fuzzy system to create a dynamic fuzzy control system [4]. However, there are various applications where dynamic models are either not available or

too complex for effective use in the fuzzy setting. Such applications tend to include video games and other interactive environments[5].

Reinforcement learning (RL) comprises a collection of learning algorithms that allow optimization of various control systems. RL trains agents to adapt to a given environment via a reward system[6]. The agent takes action by assessing the state of the environment, and RL allows the agent to maximize the expected reward. Such algorithms are well suited for complex environments; RL has shown great success in games[7] and industrial applications[8].

Deep RL[9] has recently seen significant developments as agents have successfully outperformed humans in games such as Go[7], Poker[10], and video games[11]. This approach also finds effective employment in applications such as autonomous vehicles[12], UAVs[13], and fine-tuning of large language models[14]. However, RL models and related deep learning architectures, in particular, tend to have a significant drawback in their relative lack of explainability.

Explainable machine learning is a subject of extensive research[15–17] concerned with rendering relevant algorithms more understandable, transparent, and trustworthy. Explainable fuzzy AI has afforded significant developments in this domain due to characteristics such as amenability to visualization and expression in natural language[18–20]. Notably, literature discussing directed RL in the setting of explainable fuzzy AI proves scarce despite recent advancements in the former and a capacity for mutual benefit; exploration of genetic algorithms proves more common historically[21–23].

To aid in developing RL algorithms capable of producing explainable models, fuzzy RL-based architectures[24] show promise. Fuzzy Q-learning was proposed and studied in a series of research papers[25–27] primarily concerned with performance in control applications rather than explainability in general. Additionally, these explorations are disconnected from recent developments in deep RL. This presents a gap in the literature. However, in[28], the author developed Takagi-Sugeno-Kang (TSK) fuzzy systems with successful applications in various environments where Deep Q-Learning Network (DQN)[29] has illustrated effectiveness. Similar to an Adaptive Network-Based Fuzzy Inference System (ANFIS)[30], this approach leverages RL to train an agent in these environments and shows the promise of such experimentation.

This paper offers further practical study of RL-based TSK fuzzy systems and ANFIS architectures as solutions for developing explainable AI. The latter is compared to traditional Deep Q-learning algorithms. After a preliminary section where we provide an overview of the conceptual building blocks of our systems, we discuss the successful and winning application of RL-based TSK systems to the Asteroid Smasher framework and corresponding 2023 Explainable AI competition[31]. Additionally, we offer a comparison of ANFIS and DQN in the CartPole environment. The value of the relationship between RL and fuzzy systems to explainability is highlighted and discussed in both cases.

2. PRELIMINARIES

2.1. Fuzzy systems

Fuzzy sets are defined as functions $A : X \to [0, 1]$ and are interpreted as sets with a continuum of membership grades[1]. They are used in modeling under uncertainty, especially in a rule-based environment.

Fuzzy rules describe an imprecise cause-effect relationship between certain variables controlling a given system. Fuzzy rules are of the form

$$\text{If } x \text{ is } A \text{ then } y \text{ is } B$$

where A and B are fuzzy sets on domains X and Y, respectively. Fuzzy rules can be organized into fuzzy rule

bases

$$\text{If } x \text{ is } A_i \text{ then } y \text{ is } B_i$$

with antecedents A_i and consequences B_i, $i = 1, ..., n$ being fuzzy sets.

Mamdani fuzzy systems were introduced in [2], with the goal of developing fuzzy rule-based control systems. Mamdani fuzzy systems defined by the above rule base evaluate a fuzzy output as

$$B'(y) = \vee_{i=1}^{n} A_i(x) \wedge B_i(y).$$

The output of the system is then calculated using a defuzzification as an example

$$COG(B') = \frac{\int_Y B'(y) \cdot y dy}{\int_Y B'(y) dy}$$

To model more complex systems, one can define systems with multiple antecedents and connect them with an aggregation operator.

TSK Fuzzy Systems have been introduced in [3,32] and are based on rules with a fuzzy antecedent and a singleton consequence

$$\text{If } x \text{ is } A_i \text{ then } y = y_i.$$

In this paper, we are using a rule with a singleton consequence; however, TSK systems can also be defined with linear or higher-order consequences. The output of a TSK fuzzy system can be calculated as

$$TSK(x) = \frac{\sum_{i=1}^{n} A_i(x) \cdot y_i}{\sum_{i=1}^{n} A_i(x)}.$$

In the case of multiple antecedents, the rule base is modified as an example

$$\text{If } x \text{ is } A_i \text{ and } y \text{ is } B_i \text{ then } z = z_i.$$

with the evaluation of the output being

$$TSK(x, y) = \frac{\sum_{i=1}^{n} A_i(x) \cdot B_i(y) \cdot z_i}{\sum_{i=1}^{n} A_i(x) \cdot B_i(y)}.$$

In the present paper, we use TSK fuzzy systems in the context of RL.

2.2. Reinforcement learning

2.2.1. Bellman's Equation

Bellman equation, named after Richard E. Bellman for his work in [33], is an equation for controlling systems based on states, rewards, and actions, which altogether allow us to compute the value for the state, i.e., the total expected reward in a given state, given by

$$V(s) = \max_a (R(s, a) + \gamma \sum P(s, a, s')V(s').$$

$V(s)$ is called the value of a state s. This function gets updated as the agent learns more information about the state, especially which actions (a) yield the highest cumulative reward. The reward r is given for a state-action pair and is not bound by any restrictions. γ is a discount factor, typically specified between 0 and 1, with a common value of 0.99. γ is tuned to balance "near" and "far" rewards with lower values prioritizing immediate rewards and higher values valuing future rewards more. P is the probability of ending in state s' (next state) by taking action a when starting from state s. $V(s')$ is the value of the next state. This recursive definition allows policies to learn which actions to take on a given state to maximize the cumulative reward of future states.

2.2.2. Q-learning

Q-learning[34] is a model-free RL algorithm, which will learn the Q-values or the expected reward for a state. Rather than attempt to model the environment, Q-learning aims to predict preferable actions to take in a specific state by extending the Bellman Equation as

$$Q^{new}(s,a) = Q(s,a) + \alpha \cdot (R(s,a) + \gamma \max_a Q(s',a) - Q(s,a)).$$

Q^{new} are the new Q-values of the state-action pair. These new values get updated from the previous Q-values and added to α, a learning rate, multiplied by the temporal difference (TD). The TD is the current reward of the state-action pair added to the discount factor γ multiplied by the maximum reward that can be earned from the next state before the current value of the state-action pair is subtracted. The Q-function aims to update the action that should be taken for a given state to maximize the cumulative reward.

2.2.3. Deep Q-learning

Common techniques for Q-learning include creating a Deep Neural Network (NN) for predicting the Q-function followed by optimization via backpropagation. This model is known as a Deep Q-Learning Network[29] and is a particular case of Q-learning that relies on a Deep NN architecture. This allows the agent to learn continuous states, learn continuous values in discrete states, and generalize to states not yet seen. For example, a simple Q-learning algorithm could involve creating a lookup table where each element is a state of an environment such as Grid World[6]. However, such an approach does not generalize to more complex environments such as StarCraft II[35]; the practically countless number of possible states renders their storage in an extremely inefficient table, and Deep Q-learning handles these tasks elegantly.

2.2.4. Improvements to Q-learning and Deep Q-learning

One very common problem with Q-learning is sampling inefficiency and over-estimation of the Q-values. There are several methods to address these problems. Here, we will discuss experience replay, double Q-learning, and actor-critic architectures.

Experience replay[36] allows Q-learning agents to be more sample efficient by storing transitions or collections of states, actions, rewards, and next states. Instead of exclusively learning from a current state, agents sample prior experiences. This allows the agent to revisit states it has already visited and learn more to speed up convergence of the agent's current policy. Popular and powerful extensions to experience replay for improving sample efficiency include prioritized experience replay (PER)[37] and hindsight experience replay (HER)[38].

Double Deep Q-Learning (DDQN)[39] is another technique to aid in the stability of training. A second model, which is a copy of the original network, is offline (frozen) for a set number of training iterations. The target Q-values are then sampled from the target network and used to compute the Q-values for the online (unfrozen) network. One issue with DDQN is the tendency of the online model to move too aggressively towards optimal performance and negatively impact the stability of training. To address this, a soft update, known commonly as Polyak Updating[40], is performed on the target network using the weights of the online model multiplied by a small value. This form of weight regularization allows the target model to slowly improve over time and prevents harsh updates to the learning.

An actor-critic RL architecture consists of an actor that acts out the current policy and estimates the current value policy and a critic that evaluates the current policy and estimates the policy gradient using a TD evaluation[41]. The system also encompasses a supervisor that controls the balance between exploration (performed by the actor) and exploitation (performed by the critic).

3. REINFORCEMENT LEARNING WITH FUZZY SYSTEMS

RL in the context of fuzzy logic was introduced in [24] and based on the Mamdani framework. It was successfully used in the CartPole problem. RL TSK fuzzy systems have been developed for various applications [28,42].

3.1. Reinforcement learning TSK fuzzy system

As learned behaviors in complex systems are often the result of intricate optimization processes, explainable RL constitutes a challenging problem. However, the interpretability of fuzzy systems indicates the potential value of a method of TSK fuzzy system optimization inspired by traditional Q-learning. In other words, a system in which one approximates the Q function with a TSK fuzzy system [28] stands to offer some unique benefits. This approach was successfully employed in applications such as a simulation in a simple discrete grid-world environment or in continuous environments such as CartPole or Lunar Lander [28].

The idea in the setting of a TSK-based Q-learning is to set the Q function to be approximated by a TSK fuzzy system, i.e.,

$$Q(s,a) = TSK(s,a).$$

TD Q-learning equation is given by

$$Q(s,a) = Q(s,a) + \alpha(R(s') + \gamma Q(s',a) - Q(s,a))$$

When updating the parameters of a system in RL we can use the general equation

$$w_i = w_i + \alpha(R(s') + \gamma Q(s',a) - Q(s,a))\frac{\partial Q}{\partial w_i}$$

with $w_i, i = 1, ..., n$ being parameters of the system. As a result, we can update the parameters of the fuzzy system using this approach in the context of a TSK fuzzy system as

$$w_i = w_i + \alpha(R(s') + \gamma TSK(s',a) - TSK(s,a))\frac{\partial TSK(s,a)}{\partial w_i}.$$

Here, w_i are the parameters of the fuzzy system. Just as in the case of RL, one may wish to mitigate instability via techniques such as experience replay and the usage of an actor-critic environment.

To give a theoretical foundation to our proposed algorithm, we include here an initial discussion on convergence. First, we observe that the Q-learning algorithm is known to be convergent [43,44] under standard assumptions on the given Markov Decision Process (MDP). Additionally, NNs with various membership functions are known to be universal approximators [45,46]. Combining the above results and the conclusions in [47,48], we can approximate the Q function by the output of the NN, which leads to the convergence of Deep Q-learning models. It is known that TSK fuzzy systems are universal approximators [49,50], i.e., they have similar approximation properties to those of NNs. Together, the ideas above allow the conclusion that replacing the NN in a DQN architecture with a TSK fuzzy system will retain the same properties as DQNs. In summary, the TSK fuzzy system is an approximator of the Q function. Therefore, the Q-learning algorithm with TSK fuzzy systems is convergent. The above theoretical motivation warrants a deeper investigation of the approximation properties of RL TSK fuzzy systems as a topic for future research.

3.2. Case study: Reinforcement learning TSK system for Asteroid Smasher

3.2.1. Problem description

To test the algorithm, we created optimized TSK fuzzy systems through RL to play a variant of the game Asteroids called Asteroid Smasher (see Figure 1). Developed by the University of Cincinnati for the Explainable Fuzzy AI Challenge (XFC 2022), the game environment incorporates additional complexities to increase both the difficulty and value of explainable automation. These include the addition of multiple agents, screen-wrap for ships and hazards, and unique scenarios to test edge cases.

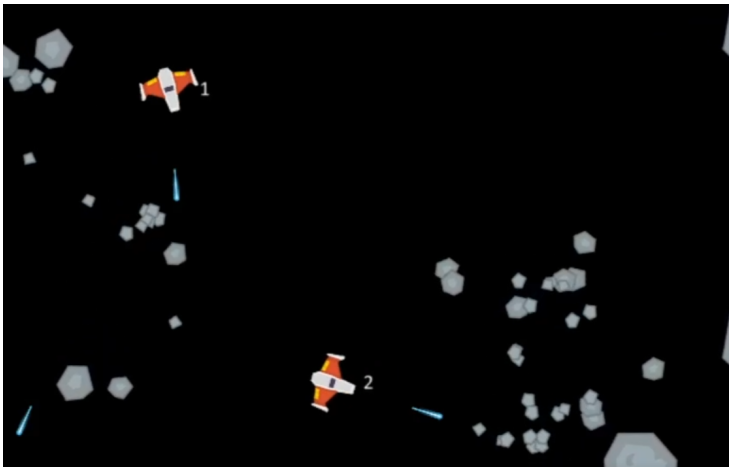


Figure 1. Example of the Asteroid Smasher environment.

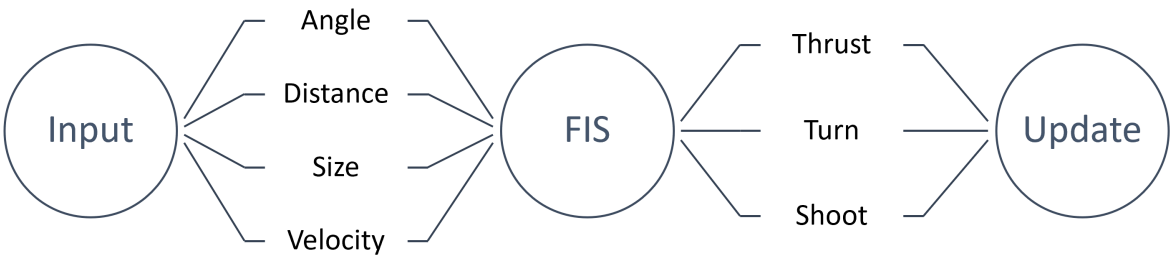


Figure 2. Simplified overview of the fuzzy inference system's threat inputs and behavioral output.

While the primary metric for performance concerned the total number of asteroids destroyed, other metrics included accuracy, number of lives lost, and execution speed. More importantly, the explainability of the system proved crucial as the motivating factor for the test case. In pursuit of balancing AI explainability with performance, TSK systems lending themselves to visualization and descriptions through natural language were developed and optimized via the described learning algorithm.

3.2.2. System description

We created a base model comprised of fuzzy systems to serve as a foundation for optimization.

To balance score and execution time, each ship tracks distances and angles to immediate threats and potential targets. These values take into account considerations such as screen wrap and object size. More importantly, they serve as inputs to fuzzy systems that determine shooting, turning, and thrust behavior. A general overview of this process is shown in Figure 2.

3.2.3. Learning algorithm

The employed algorithm for optimization approximates gradient descent by iteratively running scenarios and altering TSK outputs to more closely resemble the best performer. This requires establishing a performant objective function and additional hyperparameter tuning; the range of possible TSK output values, number of iterations, and learning rate all vary based on configuration. However, the theoretical upside is a more systematic approach to optimal performance when compared to approaches such as genetic algorithms. The approach used for this project is described in Algorithm 1.

Algorithm 1 Asteroid Smasher Iterative Learning

```
Initialize FIS
Set learning rate α
for each epoch do
    Create/choose a scenario
    Test scenario with the current FIS and save score in the interval [0, 1]
    for i in range(k) do
        Create a alteration vector h with adjustments to each TSK output parameter
        Create a new fuzzy system by adding h to the baseline's parameters
        Test scenario with altered fuzzy system and save score in the interval [0, 1]
    end for
    Calculate maximum score difference s = best − baseline
    Create the new current FIS with currentFIS = currentFIS + (α ∗ s ∗ h)
end for
```

In other words, we created a number of slightly modified fuzzy systems per scenario and borrowed slightly from the alterations made to the best performer. The degree to which the modification is added to the baseline is a product of the alteration, the given learning rate, and the difference between the maximum score and baseline. Some stability was gained by adding the last condition; scores that were only slightly better appeared to warrant proportionally smaller adjustments.

Scenario creation constitutes an important component of this process. For some problems, this may be trivial, but others may require consideration of the problem, goal, and available resources. To train for the competition, highly difficult and random scenarios with a very large number of asteroids were generated in lieu of a battery of scenarios for reasons primarily related to ease of implementation.

3.2.4. Experimental results

Applying the discussed iterative learning method resulted in a story of minor but noticeable performance increase across key competition metrics such as score, win rate in difficult scenarios, and number of successful hits on asteroids.

Performance comparisons of the base model, the product of optimization, and a model where TSK outputs were randomly initialized are shown in Figure 3 and Figure 4. It is important to note that, in this application, a manually tuned agent served as the starting point of optimization rather than the randomly initialized one.

Even a sizable rolling average across epochs still illustrates notable instability. Part of this is likely attributable to the randomness innate to the extreme training scenarios. However, testing methods similar to those described previously for increasing the stability of RL may constitute a key area of further research.

3.2.5. Explainability

An advantage of the FIS is its ability to encapsulate complex behavior in an interface that deals in partial truths and natural language; functions and interactions that would prove difficult to interpret otherwise are meaningfully organized in comprehensible fuzzy sets and rules. More fundamentally, dealing with partial truths can prove more familiar and intuitive than crisp logic to a human expert.

In this case, the developed systems are comprised of a natural language rule set and associated fuzzy sets that lend themselves to visualization. Figure 5 illustrates this with plots of antecedent fuzzy sets and 0^{th} order outputs describing varying levels of ship thrust. Included is the associated rule set in which the relationships

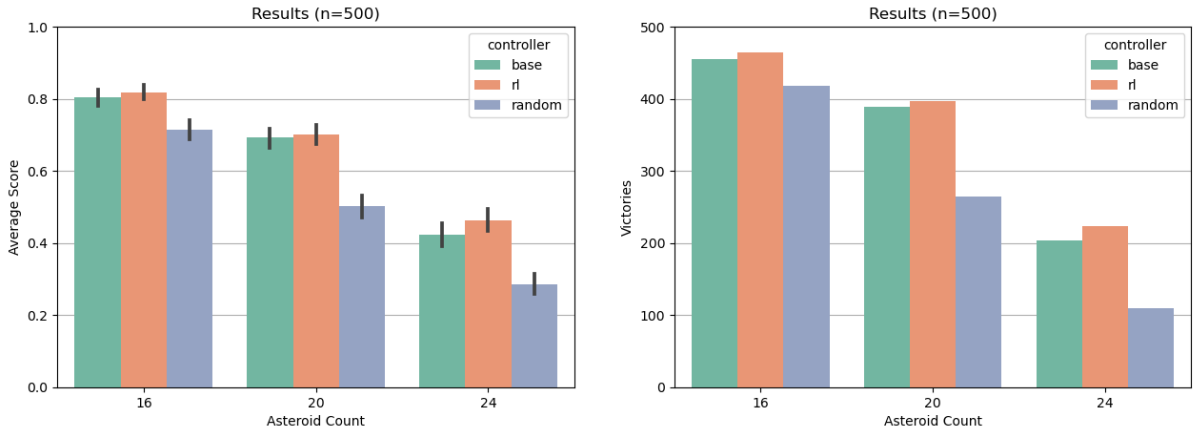


Figure 3. Comparisons of average game score and number of victories in challenging scenarios for the manually tuned model (base), the product of reinforcement learning (RL), and randomly initialized models (random).

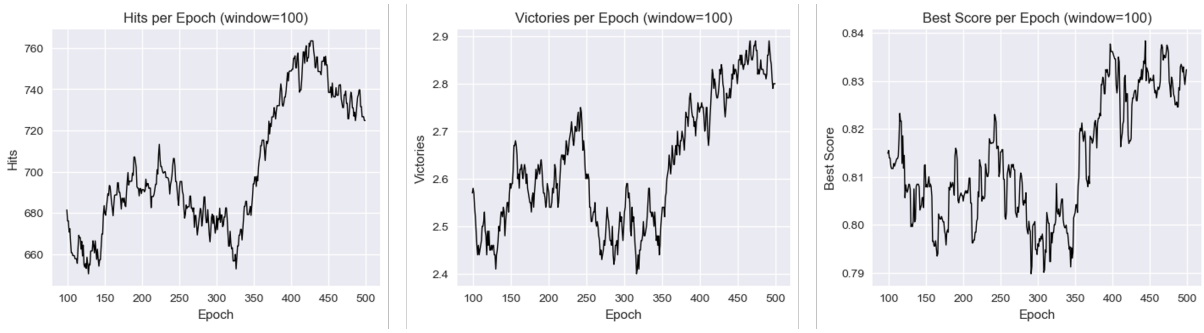


Figure 4. Rolling averages of several performance metrics throughout training.

between inputs and outputs are encoded in natural language. Similar systems were employed to determine the relative value of asteroids as targets and dictating turning and shooting behaviors.

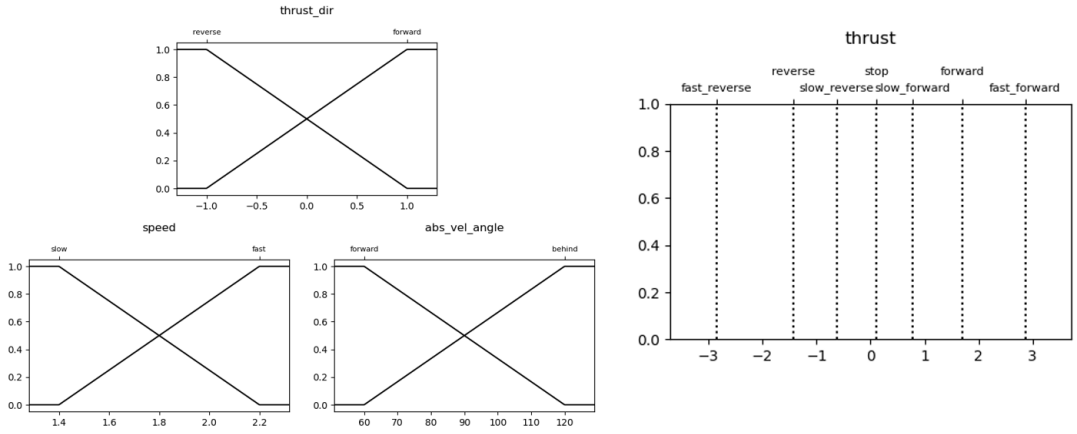
As previously discussed, the explainability of the FIS in applications such as control systems is already well established. Rather than further clarifying this, the described algorithm and Asteroid Smasher example instead serve to delineate and test an alternative to more commonly employed methods for the optimization of existing FIS-based architectures. Successful application here offers experimental evidence to join the relatively scarce literature on developing fuzzy systems with RL as opposed to historically prevalent approaches such as genetic algorithms.

3.3. Case study: reinforcement learning ANFIS for classic control environments

3.3.1. Introduction to ANFIS

The intersection of fuzzy logic and RL stands to offer more than an alternative method of post-hoc optimization for the former: notoriously opaque NN architectures also stand to benefit from integration with explainable fuzzy systems. In other words, neuro-fuzzy systems offer a means for taking advantage of the strengths of artificial NNs while peering a little further into the black-box models that find widespread use in RL applications and elsewhere.

To bolster experimental results in this domain, we tested an ANFIS[30] (Figure 6) on OpenAI's classical control environment CartPole[51]. An example of the environment is shown in Figure 7. The ANFIS extends the TSK system by allowing the parameters of the fuzzy rules to be learned via gradient descent optimization rather



if thrust-dir is reverse and abs-vel-angle is forward and speed is fast then thrust is fast-reverse
if thrust-dir is reverse and abs-vel-angle is forward and speed is slow then thrust is reverse
if thrust-dir is reverse and abs-vel-angle is behind and speed is slow then thrust is slow-reverse
if thrust-dir is reverse and abs-vel-angle is behind and speed is fast then thrust is stop

if thrust-dir is forward and abs-vel-angle is forward and speed is fast then thrust is stop
if thrust-dir is forward and abs-vel-angle is forward and speed is slow then thrust is slow-forward
if thrust-dir is forward and abs-vel-angle is behind and speed is slow then thrust is forward
if thrust-dir is forward and abs-vel-angle is behind and speed is fast then thrust is fast-forward

Figure 5. Antecedent fuzzy sets, optimized TSK consequents, and associated rules for determining ship thrust.

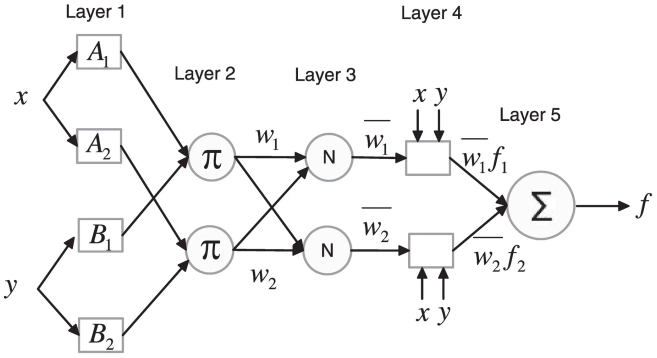


Figure 6. Example ANFIS Structure [53] with 2 antecedents and 1 consequent.

than using fixed constants for the parameters of the antecedents and the consequences. Towards this end, the ANFIS is combined with a NN to learn representations from the inputs, which are then fuzzified by the network. Outputs of the ANFIS layers are calculated by multiplying the fuzzified output of the NN by antecedents of rules. Then, the antecedents are multiplied by learnable parameters and summed to form the consequences. In the case of multiple-consequent models where the output dimension is more than a single value, we used a modified ANFIS structure: the Multioutput Adaptive Neuro-Fuzzy Inference System (MANFIS) [52]. This method creates a separate rule base for each output dimension but follows the defuzzification process as a typical ANFIS. This process is shown in Figure 8.

A NN of abstract shape is used and can be configured as desired. After passing the input through the NN, the

(a) Example of starting position.

(b) Example of state that would terminate the episode.

Figure 7. Snapshots of states for the CartPole [51] environment.

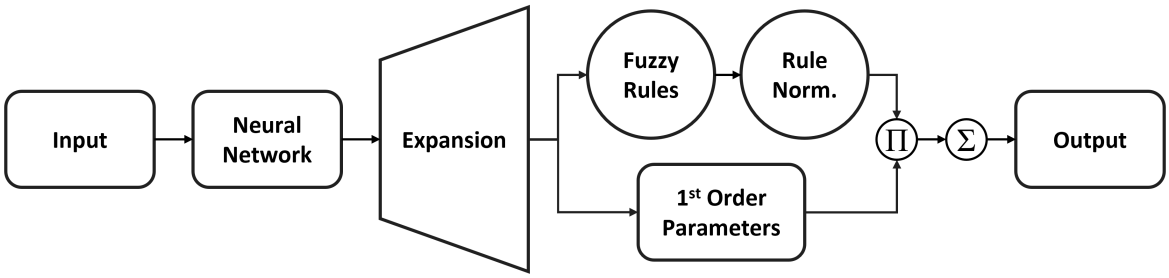


Figure 8. Flowchart for ANFIS model. NN output undergoes fuzzy rule evaluation and normalization. Separately, network output is transformed according to learned 1^{st} order parameters and multiplied by the normalized rule outputs. These values are summed and given as output.

output is expanded in dimensions to match the number of rules. This is then passed through the fuzzy rules to calculate their firing levels prior to normalization along each output dimension. Then, in the case of a 1^{st} order ANFIS, the input is multiplied by a parameter, and a bias is added. If the 0^{th} order ANFIS is used, the input is passed along to the next layer. Next, the normalized firing levels are multiplied by the inputs to form the rule evaluation. Finally, the output is summed along each rule base to form the final output with the correct dimension.

3.3.2. CartPole-v1

The CartPole-v1 environment has four variables for the input: the position of the cart, the velocity of the cart, the angle of the pole, and the angular velocity of the pole. The expected output has two actions: move left and move right. The job of the models is to learn and maximize the Q-values for each state to achieve a maximum possible reward of 500, where, in each frame, the agent gets a reward of 1 if the cart and pole are within the min and max values for each respective field. A reward of 500 means that during 500 frames, the agent is able to balance the pole. To perform an action in the environment, we take the action with the maximum Q-value.

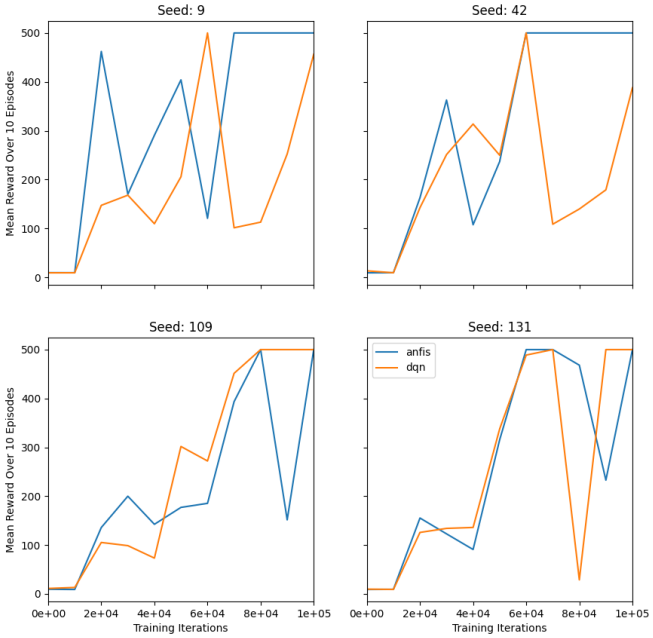


Figure 9. Comparisons between ANFIS DQN and DQN.

3.3.3. ANFIS DQN vs DQN

We compare the ANFIS DQN architecture proposed above with a standard DQN architecture. Both models are equipped with a Double DQN structure with soft updates and have a similar number of trainable parameters; 17,410 for the DQN and 17,407 for the ANFIS DQN. Both models are also equipped with the same optimization method, learning rate, etc., and trained for 100,000 iterations. The appendix contains parameters and hyperparameters.

3.3.4. Experimental results

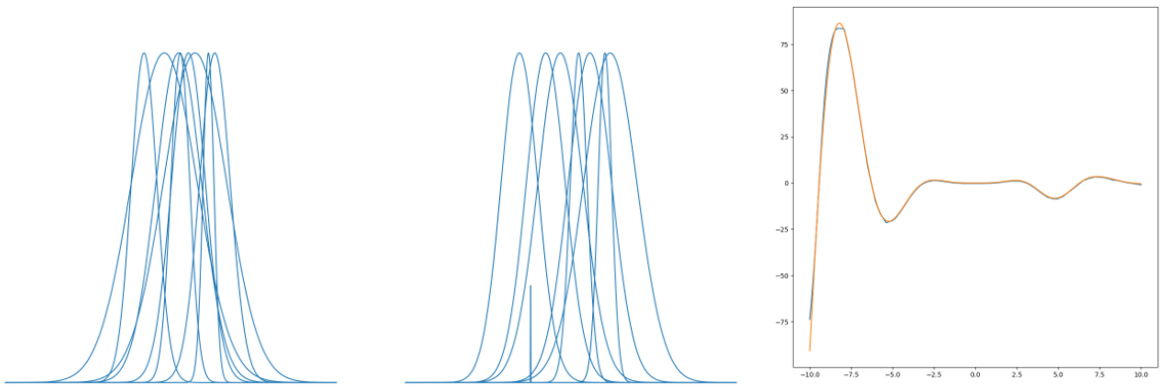
For testing the results of the two models, we look at the mean reward of 10 testing environments every 10,000 iterations. The goal is to have the agent hit a mean reward of 500 for all 10 test environments.

From Figure 9, we can see that both models can learn the environment. In some cases, we can see that the ANFIS DQN model learns the environment quicker, such as in seed 9 and seed 42. While in other cases, it seems to perform about as well as the DQN, such as in seeds 109 and 131. The most interesting case is in seed 42, where the ANFIS DQN and DQN solve the environment in the same number of steps, but the ANFIS DQN learns it with more stability while not falling off after learning and continues to have a solution after 60,000 iterations. This trend also appears in seed 9, where after 70,000 iterations, the ANFIS DQN has a stable solution. From these tests, we can see that the ANFIS DQN is able to match, if not slightly outperform, the DQN agent.

3.3.5. Explainability

As suggested, a primary advantage of this approach over non-fuzzy DQN concerns the capacity for explainability; one may visualize the fuzzy rule components before and after training to help in understanding the changes made during optimization and the resulting behavior.

To illustrate this, an example concerning the approximation of a mathematical function is shown in Figure 10. In this case, Gaussian functions corresponding to fuzzy rules expand and contract following gradient descent



(a) Rule base before training **(b)** Rule base after training **(c)** Results of function approximation

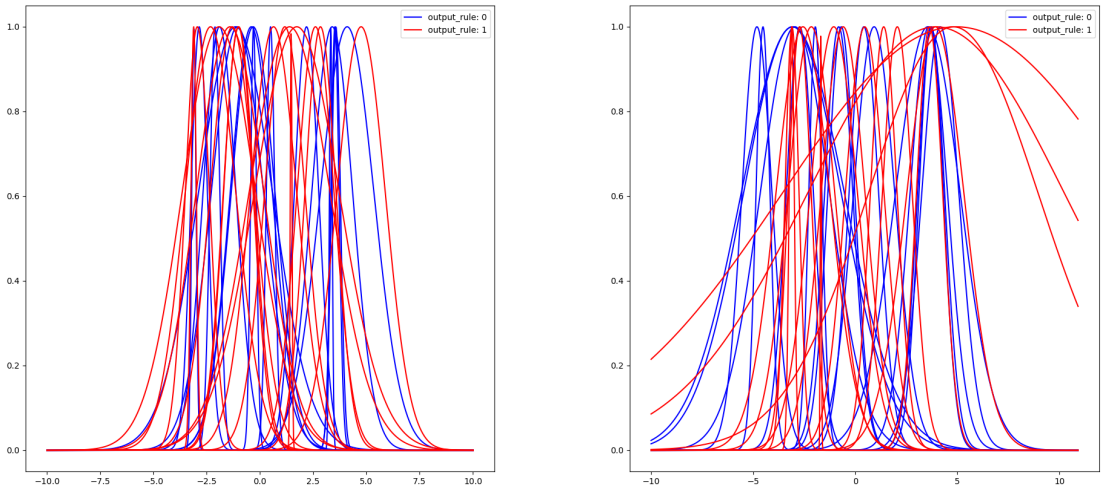
Figure 10. Comparison of rules for approximating a function before and after optimization. The straight line represents a rule that becomes insignificant after training.

to afford a close estimate of the function. Notably, one Gaussian function depicted by the straight line in Figure 10b becomes insignificant after training. Not only does this indicate potential robustness to network overparameterization, but the ability to visualize the components of the system in this way also highlights a propensity for troubleshooting architectural design and training.

The CartPole example represents a jump in complexity that corresponds with increased challenges to helpful visualization; the 32 rules for movement in either direction stand to benefit from a more directed form of presentation. However, the same principles of visual feedback for training and resulting behavior apply. Both before and after training, one may observe the rule base of the ANFIS DQN and extract important information about the decisions of the agent.

We can see how some of the behaviors are shaped by analyzing the distribution and coverage of antecedents in Figure 11. In the current visualization, it may be hard to explain all aspects of interaction, but we can investigate some possible behaviors. The blue curves indicate the rules for moving the cart to the left, and the red curves describe movement to the right. The domain is the domain for input space from CartPole. Each input that is passed into the rules comes from the output of a NN. The blue curves are related to moving the cart to the left, and the red curves for moving the cart to the right. While we cannot see exactly what inputs the rules are related to, we can see some trends that indicate what each rule may be impacting. In the beginning, both sets of colored curves are fairly uniform around the origin. After training, we can see that they have spread out to cover more domains, and some curves are wider. Some interesting things to note are there are two inputs in the observation space for cart velocity and pole angle. Negative values are associated with the pole and the cart moving to the left. We can see how some of the blue curves moved further to the left, and the red curves moved further to the right. This is because the intelligent action of moving the cart the opposite way the pole is moving can help correct the position of the pole. These small insights can give us some explanation as to what the network is doing when making decisions. With a better visualization, we could see what each rule is specifically looking at in the input related to the output.

It should be noted that there is also much unique potential for experimentation with novel explainability mechanisms. Further research of ANFIS DQN in this vein could support efforts in quantifying changes to rules after training, identifying rules that become insignificant, highlighting substantial activations for any one state, and exploring aggregations that describe overall trends in behavior. While similar ideas have certainly been explored to aid the interpretability of traditional NNs, the capacity for visualization offered by FIS-based ar-



(a) Rule base before training **(b)** Rule base after training

Figure 11. Comparison of rule antecedent distributions for moving left (blue) and right (red) in the CartPole environment. Input consists of encoded cart position, cart velocity, pole angle, and pole angular velocity.

chitectures represents a significant opportunity due to the reasons discussed and the relative lack of existing literature in this domain specifically.

4. CONCLUSION

In the present paper, we study fuzzy systems controlled via RL and multiple applications of this framework. The first environment considered is the Asteroid Smasher game in which the associated agent made use of fuzzy systems trained via an actor-critic RL algorithm. This agent won XFC 2022, an Explainable AI competition emphasizing fuzzy logic. We also compared ANFIS DQN architectures with classic DQN architectures and demonstrated that the fuzzy systems perform similarly or, in some cases, better than DQN systems. More significantly, we highlighted the amenability of TSK-based architectures to explainability and visualization when compared to more traditional NNs.

Grounds for further research include an exploration of possible improvements to the stability of the RL algorithm used to tackle the Asteroid Smasher environment. With respect to ANFIS architectures, additional exploration of applications, network architectures, membership functions, and unique explainability mechanisms could also prove valuable for both performance and explainability. In both cases, there are extensions such as improved experience replay, dueling DQN, and distributional learning that could afford improved results.

The most significant takeaway concerns the relatively underinvestigated and mutually beneficial relationship between RL and fuzzy logic. Despite trends in the former and the value that each stands to offer the other, alternative methods are more commonly employed to optimize fuzzy systems. Moreover, promising neuro-fuzzy frameworks face relative obscurity in light of the steadfast need for trust and interpretability in domains where NNs are pervasive. Consequently, additional experimentation serves to afford value when building performant and explainable applications.

DECLARATIONS

Authors' contributions

Wrote and reviewed the paper: Zander E, van Oostendorp B, Bede B

Availability of data and materials

There are no applicable datasets, but the implementation of a PyTorch compatible ANFIS layer is available here if relevant: https://github.com/Squeemos/pytorch_anfis.

Financial support and sponsorship

None.

Conflicts of interest

All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Zadeh LA. Fuzzy sets. *Inf Contr* 1965;8:338–53. DOI
2. Mamdani EH, Assilian S. An experiment in linguistic synthesis with a fuzzy logic controller. *Int J Man-Mac Studies* 1975;7:1–13. DOI
3. Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. In: *IEEE Trans Syst , Man, Cybern* 1985;8:116–32. DOI
4. Precup RE, Hellendoorn H. A survey on industrial applications of fuzzy control. *Comput Industry* 2011;62:213–26. DOI
5. Pirovano M. The use of fuzzy logic for artificial intelligence in games. *University of Milano, Milano* 2012. Available from: https://www.michelepirovano.com/pdf/fuzzy_ai_in_games.pdf [Last accessed on 6 Jun 2023]
6. Russell SJ. Artificial intelligence a modern approach. Pearson Education, Inc.; 2010.
7. Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge. *Nature* 2017;550:354–59. DOI
8. Lakhani AI, Chowdhury MA, Lu Q. Stability-preserving automatic tuning of PID control with reinforcement learning. *Complex Eng Syst* 2022;2:3. DOI
9. Li Y. Deep reinforcement learning: an overview. *arXiv preprint arXiv:170107274* 2017. DOI
10. Zhao E, Yan R, Li J, Li K, Xing J. AlphaHoldem: high-Performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36; 2022. pp. 4689–97. DOI
11. Shao K, Tang Z, Zhu Y, Li N, Zhao D. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:191210944* 2019. DOI
12. Chen J, Yuan B, Tomizuka M. Model-free deep reinforcement learning for urban autonomous driving. In: 2019 IEEE intelligent transportation systems conference (ITSC). IEEE; 2019. pp. 2765–71. DOI
13. Walker O, Vanegas F, Gonzalez F, Koenig S. A deep reinforcement learning framework for UAV navigation in indoor environments. In: 2019 IEEE Aerospace Conference. IEEE; 2019. pp. 1–14. DOI
14. Ouyang L, Wu J, Jiang X, et al. Training language models to follow instructions with human feedback. *J Inf Process Syst* 2022;35:27730–44. DOI
15. Mundhenk TN, Chen BY, Friedland G. Efficient saliency maps for explainable AI. *arXiv preprint arXiv:191111293* 2019. DOI
16. Borys K, Schmitt YA, Nauta M, et al. Explainable AI in medical imaging: an overview for clinical practitioners–Beyond saliency-based XAI approaches. *Eur J Radiol* 2023:110786. DOI
17. Holzinger A, Saranti A, Molnar C, Biecek P, Samek W. Explainable AI methods-a brief overview. In: xxAI-Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers. Springer; 2022. pp. 13–38. DOI
18. Hagras H. Toward human-understandable, explainable AI. *Computer* 2018;51:28–36. DOI
19. Mencar C, Alonso JM. Paving the way to explainable artificial intelligence with fuzzy modeling. In: Fuzzy Logic and Applications.

Springer; 2019. pp. 215–27. DOI
20. Viaña J, Ralescu S, Cohen K, Ralescu A, Kreinovich V. Extension to Multidimensional Problems of a Fuzzy-Based Explainable and Noise-Resilient Algorithm. In: Decision Making Under Uncertainty and Constraints: A Why-Book. Springer; 2023. pp. 289–96. DOI
21. Pickering L, Cohen K. Genetic Fuzzy Controller for the Homicidal Chauffeur Differential Game. In: Applications of Fuzzy Techniques: Proceedings of the 2022 Annual Conference of the North American Fuzzy Information Processing Society NAFIPS 2022. Springer; 2022. pp. 196–204. DOI
22. Pickering L, Cohen K. Toward explainable AI—genetic fuzzy systems—a use case. In: Explainable AI and Other Applications of Fuzzy Techniques: Proceedings of the 2021 Annual Conference of the North American Fuzzy Information Processing Society, NAFIPS 2021. Springer; 2022. pp. 343–54. DOI
23. Fernandez A, Herrera F, Cordon O, del Jesus MJ, Marcelloni F. Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to? *IEEE Comput Intell Mag* 2019;14:69–81. DOI
24. Berenji HR. A reinforcement learning—based architecture for fuzzy logic control. *Int J Approx Reason* 1992;6:267–92. DOI
25. Glorennec PY, Jouffe L. Fuzzy Q-learning. In: Proceedings of 6th international fuzzy systems conference. vol. 2. IEEE; 1997. pp. 659–62. DOI
26. Er MJ, Deng C. Online tuning of fuzzy inference systems using dynamic fuzzy Q-learning. *IEEE Trans Syst Man Cybern B Cybern* 2004;34:1478–89. DOI
27. Jamshidi P, Sharifloo AM, Pahl C, Metzger A, Estrada G. Self-learning cloud controllers: Fuzzy q-learning for knowledge evolution. In: 2015 International Conference on Cloud and Autonomic Computing. IEEE; 2015. pp. 208–11. DOI
28. Kumar S. Learning of Takagi-Sugeno Fuzzy Systems using Temporal Difference methods. DigiPen Institute of Technology; 2020.
29. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015 Feb;518:529–33. DOI
30. Jang JSR. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Trans Syst , Man, Cybern* 1993;23:665–85. DOI
31. King SD. Explainable AI competition; 2023. Accessed on 2/15/2023. Available from: https://xfuzzycomp.github.io/XFC/. [Last accessed on 6 Jun 2023]
32. Sugeno M, Kang G. Structure identification of fuzzy model. *Fuzzy Sets Syst* 1988;28:15–33. DOI
33. Bellman R. A markovian decision process. *Indiana Univ Math J* 1957;6:679–84. Available from: https://www.jstor.org/stable/24900506 [Last accessed on 6 Jun 2023]
34. Watkins CJCH, Dayan P. Q-learning. *Mach Learn* 1992 May;8:279–92. DOI
35. Vinyals O, Babuschkin I, Czarnecki WM, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 2019;575:350–54. DOI
36. Lin LJ. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach Learn* 1992;8:293–321. DOI
37. Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. arXiv; 2015. DOI
38. Andrychowicz M, Wolski F, Ray A, et al. Hindsight Experience Replay. arXiv; 2018. DOI
39. van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. arXiv; 2015. DOI
40. Polyak BT, Juditsky AB. Acceleration of Stochastic Approximation by Averaging. *SIAM J Control Optim* 1992 Jul;30:838–55. DOI
41. Rosenstein MT, Barto AG, Si J, et al. Supervised actor-critic reinforcement learning. *Learning and Approximate Dynamic Programming: Scaling Up to the Real World* 2004:359–80. Available from: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ec4de9c9729f9301aefa713ac42f194a364a4406 [Last accessed on 6 Jun 2023]
42. Yan X, Deng Z, Sun Z. Competitive Takagi-Sugeno fuzzy reinforcement learning. In: Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01)(Cat. No. 01CH37204). IEEE; 2001. pp. 878–83. DOI
43. Jaakkola T, Jordan M, Singh S. Convergence of stochastic iterative dynamic programming algorithms. *Adv Neural Inf Proces Syst* 1993;6. Available from: https://proceedings.neurips.cc/paper/1993/file/5807a685d1a9ab3b599035bc566ce2b9-Paper.pdf [Last accessed on 6 Jun 2023]
44. Melo FS. Convergence of Q-learning: a simple proof. *Inst Syst Robot, Tech Rep* 2001;1–4. Available from: https://d1wqtxts1xzle7.cloudfront.net/55970511/ProofQlearning-libre.pdf?1520268288=&response-content-disposition=inline%3B+filename%3DConvergence_of_Q_learning_a_simple_proof.pdf&Expires=1686128725&Signature=SZMvoQSM3Z7UPmyXfT4QOw8Co0pUvQM1h3NfUwa3aJXBPsj8ox1O9WI~QaTZrpZ5~Cr4NcfcDmsh~IUjT101xNeKR2-PCvewznfNXB38~UEGN736l3lniIQLKe1QdebMTHgvtL7iDOivntOKxrLAnzUx0I4dYlAuYUf3qBNk37aqJtIH6WTpCuJUKeH3pW282tY11MVEK0P~Czp-WsOkY8wtMOu8~NCNcS2sR6d1rhV1JeWPv1BuTAg6-hBUFFhbqLIY7SvJ8j6IWA0bJy~Miaz4Q2C37sOi2eo2~y819e3F3jiby3mMWeEpf1WYPUWK~0hB475dafC5FGZcEKTVjA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA [Last accessed on 6 Jun 2023]
45. Cybenko G. Approximation by superpositions of a sigmoidal function. *Math Control Signal Systems* 1989;2:303–14. DOI
46. Yarotsky D. Error bounds for approximations with deep ReLU networks. *Neural Netw* 2017;94:103–14. DOI
47. Melo FS, Ribeiro MI. Q-learning with linear function approximation. In: Learning Theory: 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA; June 13-15, 2007. Proceedings 20. Springer; 2007. pp. 308–22. DOI
48. Papavassiliou VA, Russell S. Convergence of reinforcement learning with general function approximators. In: IJCAI. vol. 99; 1999. pp. 748–55. Available from: http://people.eecs.berkeley.edu/~russell/papers/ijcai99-bridge.pdf [Last accessed on 6 Jun 2023]
49. Ying H. General Takagi-Sugeno fuzzy systems are universal approximators. In: 1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36228). vol. 1. IEEE; 1998. pp. 819–23. DOI
50. Bede B. Mathematics of Fuzzy Sets and Fuzzy Logic. Springer; 2013.
51. Brockman G, Cheung V, Pettersson L, et al. Openai gym. *arXiv preprint arXiv:160601540* 2016. DOI
52. Benmiloud T. Multioutput Adaptive Neuro-Fuzzy Inference System. In: Proceedings of the 11th WSEAS International Conference on

Nural Networks and 11th WSEAS International Conference on Evolutionary Computing and 11th WSEAS International Conference on Fuzzy Systems. NN'10/EC'10/FS'10. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS); 2010. p. 94–98.

53. Al-Hmouz A, Shen J, Al-Hmouz R, Yan J. Modeling and simulation of an adaptive neuro-Fuzzy inference system (ANFIS) for mobile learning. *IEEE Trans Learning Technol* 2012;5:226–37. DOI

APPENDIX: ANFIS AND DQN HYPERPARAMETERS

Most hyperparameters were taken from previous work/recommended defaults of libraries

1. DQN Structure
    - 2 linear layers with 128 nodes and ReLU activation functions
2. ANFIS Structure
    - 2 linear layers with 128, 127 nodes, respectively, and ReLU activation functions
    - 16 rules, with the mean of the Gaussian membership function sampled from the Normal distribution and scaled by 2 to increase rule-base coverage, and the standard deviations sampled from the uniform distribution
    - Learnable parameters/biases for summation sampled from the Normal distribution and scaled by 2
3. Optimizer: ADAM
4. Learning rate: .001
5. Gamma discount factor: 0.99
6. Max replay memory size: 10,000
7. Batch Size: 128
8. Begin training after: 1,000 iterations
9. Epsilon start: 1.0
10. Epsilon end: .01
11. Epsilon decay: After 20,000 iterations in a linear fashion
12. Gradient clipping norm: 10
13. Target network update iterations: 100
14. Tau soft update parameter: .001