Redefining Activation Dynamics: An Empirical Comparison of Oscillating, Mixture, and Adaptive Activation Functions in Deep Reinforcement Learning

Liong Gele
School of Computer Science
University of Nottingham
Malaysia
hfygl1@nottingham.edu.my

I. INTRODUCTION

Deep Reinforcement Learning (DRL) combines Reinforcement Learning's strategic prowess with Deep Neural Networks' (DNNs) robust representation abilities, advancing fields from gaming to autonomous navigation [10]. Central to DRL's effectiveness is DNNs' function approximation, essential for state-action mapping and reward prediction. However, DNN training faces hurdles like the vanishing gradient problem [1], addressed through pre-training methods like Hilton proposed pre-training algorithms to alleviate such issues, providing a more robust [2] Restricted Boltzmann Machines for improved weight initialization [3].

This paper focuses on a less explored but crucial aspect of DRL: the impact of activation function selection on network performance. Activation functions (***Fig 1)***, pivotal in controlling neuron firing rates, significantly influence learning dynamics. We explore this through key questions:

1. How do oscillating activation functions fare against traditional ones in enhancing Deep Q Networks (DQNs) performance in control tasks?
2. What performance variations arise with mixed activation functions in DQNs?
3. Do adaptive activation functions with gating mechanisms and hierarchical structures significantly outperform static ones in DQNs?

Our study provides an empirical comparison of oscillating, mixture, and adaptive activation functions in DQN frameworks, evaluating their effectiveness across various control tasks. We aim to highlight how these functions impact DRL and guide future advancements in neural network design. This research, focusing on diverse activation function categories including Fixed Shape (FSAFs) [5], Oscillatory (OAFs) [6], Mixture (MAFs) and Adaptive Activation Functions (AAFs) [4], redefines activation dynamics and aims to contribute to the evolution of more sophisticated and adaptable DQNs architectures.

Our investigation provides a comprehensive empirical comparison of oscillating, mixture, and adaptive activation functions within DQNs frameworks. We meticulously evaluate their efficacy across various control tasks to uncover their unique benefits and limitations. This study aims to offer a nuanced understanding of how these activation functions influence DRL, guiding future research towards more advanced and capable neural architectures.

II. BACKGROUND

In the landscape of neural network research, Fixed Shape Activation Functions (FSAFs) like Tanh and ReLU have been foundational, with Tanh aiding normalization but struggling with vanishing gradients, and ReLU being efficient but prone to 'dying neurons.' Recent studies, such as those detailed [11], have highlighted these limitations, emphasizing the need for innovation in activation functions. The emergence of Oscillatory Activation Functions (OAFs), particularly the Growing Cosine Unit (GCU) and Shifted Quadratic Unit (SQU), has opened new doors. These OAFs, as explored in [6], introduce enhanced

non-linearity, proving beneficial in convolutional networks and presenting novel approaches for deep learning applications.

Adaptive Activation Functions (AAFs) such as Parametric ReLU (PReLU) and Exponential Linear Unit (ELU), target the inherent shortcomings of ReLU. However, their application in deep networks, especially concerning parameter training and scaling effects, remains an area ripe for further exploration, as suggested by recent literature. Mixture Activation Functions (MAFs), proposing a synergistic blend of functions like LReLU and ELU, are at the forefront of creating adaptable models for varied datasets. The current research, including insights from [14] points towards the potential of these hybrid functions but also underscores the nascent stage of determining optimal combinations and training approaches. Gated Activation Functions (GAFs), introduced by S. Qian, represent a significant stride in dynamic training adaptability. These functions blend activations such as LReLU and ELU yet optimizing their gating parameters and evaluating their learning impact continues to be a challenge, as discussed in recent studies.

Lastly, Hierarchical Activation Functions, also a contribution of Qian, have brought a new dimension to neural networks. Their layered, dynamic structure, aligning closely with the concepts explored in [14], promises enhanced adaptability and data-specific activations. However, efficiently integrating these complex structures into various architectures remains a challenge, highlighting a crucial area for future research.

Table 1 shows all the formula of activation functions in RL Agent. **Fig 2** in Appendix shows all graph behaviour of each activation functions.

III. METHDOLOGY

a. Environment and Experiment Design

Our study investigates the impact of various activation functions on the learning dynamics of a DQNs using the "CartPole-v1" simulation. The objective is to maintain a vertical pole on a moving cart, rewarding upright stability while penalizing tilting or boundary breaches. We measure success by the agent's ability to balance the pole for 200 consecutive steps. We evaluate different activation functions by conducting 15 trials each, measuring the number of episodes to reach 300 timesteps and recording the mean cumulative duration of pole balance (Duration Reward) and the degree of pole verticality (Auxiliary Reward). The effectiveness of each function is quantified by the episodes required to reach the learning milestone and the precision of pole balance. Further experimental details will be elaborated in the following sections.

b. DQN model

Our experiment uses Adam Paszke's PyTorch-based DQNs model [15] as a foundation, supplemented by Dr. Tomas' methodologies for experimental structure and plotting functions. The 'run_experiments' function tests various activation functions within the DQN's architecture to assess their effects on learning efficiency and stability in a pole-balancing task. For the most of hyperparameters in **Table 2** (such as learning rate schedule, weight, etc), we follow the settings consistently released by original works. After configuring models, we train the DQN models with the regarding the activation functions that needed for the experiments.

c. Training Strategy

We explore a range of activation function strategies within Deep Q-Network (DQN) architectures. Our approach is informed by insights from the paper, which provides a detailed evaluation of 23 activation functions across multiple benchmarks [16].

From this analysis, we have carefully selected activation functions that represent the top performers in each category. Our selection includes two FSAFs (ReLU, Tanh), two OAFs (GCU, SQU), and two AAFs (PReLU, ELU). Next, the investigation of mixed activation strategies, we combine different activation functions in a single network with following equation: $f(x) = f_{\text{relu}}(x) + f_{tanh}(x)$. This is implemented via a concatenation approach,

where outputs from individual activation functions are merged and concatenated.

Additionally, we explore gated activations, as described where a gating mask is learned to dynamically modulate the input through functions. Gated activations adaptively learn a gating mask, offering a dynamic interplay between the inputs and activation functions. An example of a gated activation function: $f(x) = \sigma(\omega x) \cdot f_{relu}(x) + (1 - \sigma(\omega x)) \cdot f_{tanh}(x)$, with σ as the sigmoid and ω is the learnable gating mask.

Another experiment will focus on hierarchical activation experiment in a DQN, utilizing varied activation functions like ReLU, ELU, PReLU, Tanh, GCU, and SQU. Following the proposed structure shown in **Fig 3,** these are applied layer-wise and integrated using a winner-take-all strategy to choose the best activation response. Initially, the model processes inputs linearly, followed by channelling through the activations. The activations are then layered and filtered by selecting the highest responses, enhancing adaptability. This hierarchical method culminates in a final layer where a simple linear output is derived from the most effective previous activations.

We evaluate the performance of various activation functions in Deep Q-Networks (DQNs) based on three criteria: learning speed, reward accumulation, and consistency. Models that learn tasks faster, gather more rewards, and show lower variability in performance across trials are ranked higher. These metrics collectively determine the effectiveness of each activation function in DQNs, providing insights into their impact on learning. **Table 3** demonstrates the experimental design in table form.

RESULT

All the experiments result has been recorded in **Table 4** in the Appendix. In our comparative study, we assessed the performance of FSAFs and OAFs within DQNs applied to the CartPole environment. For FSAFs, the experimental results in **Table 4, Fig 4** and **Fig 5** demonstrate that Tanh exhibited rapid learning, completing the task by episode 26 with higher rewards with 80653, while ReLU showed more consistent performance over later episodes.

In the OAFs category, the fundamental SQU activation function, $f(x) = x^2 + x$ which initially led to unstable learning results which shown in **Fig 6**. Modifying SQU, $f(x) = \alpha(x^2) + x$ by introducing a scaling parameter (α) to the function, enhanced learning stability and performance, resulting in higher rewards and reduced variability, like in **Fig 7**. The result for GCU **(Fig 8)** provided stable learning, completing tasks by episode 35 with consistent and lower performance metrics. In contrast, the Shifted Quadratic Unit (SQU) demonstrated faster task resolution by episode 34 but with greater variability in performance in **Fig 9**. Overall, GCU was identified as the preferred oscillatory activation function for its predictability and stable learning curves.

In our DQN experiments comparing AAFs, the ELU outperformed PReLU in key areas which shown in **Fig 10** and **Fig 11**. ELU led to faster task completion with higher rewards and demonstrated greater consistency and stability, as indicated by lower standard deviations. Overall, ELU appears more advantageous for certain DQN scenarios, offering quicker learning and more stable rewards, though PReLU's flexibility still holds significance in diverse contexts.

The MAFs combining Tanh and ReLU completed the CartPole task in 26 episodes which shown in **Fig 12**, the Tanh and GCU in **Fig 14** mix but with more consistent performance. The Tanh and ReLU, MAFs exhibited stable learning, leveraging Tanh's normalization and ReLU's gradient maintenance for efficient learning. While the GCU-involved MAFs also performed well, it introduced greater variability, suggesting its potential in scenarios that benefit from added complexity. Overall, the Tanh and ReLU combination emerged as the most stable, efficient MAFs in our study **(Fig 13).**

Comparative analysis of DQNs utilizing Gated AFs, three distinct conditions, Condition 1, 2 and 3 which has shown in **Fig 15, Fig 16, Fig 17.** The combination of gated

activation in Tanh and GCU (Condition 3) has achieved the quickest task resolution by the 26th episode compared other conditions in **Table 4.** It also experienced greater variability in learning, as evidenced by the higher standard deviations. But Condition 2 exhibited the most stable learning with the lowest standard deviations among the three conditions, indicating a consistent learning process with less fluctuation.

In our comparative analysis of DQNs with adaptive Hierarchical Structure Activation Functions (HsAFs), we observed distinct outcomes across three distinct experimental conditions, Condition 1, 2 and 3 which has shown in **Fig 18, Fig 19, Fig 20.** Condition 2, utilizing Tanh and GCU, outperformed others by completing tasks rapidly with the highest rewards and showcasing stable learning patterns. Conversely, Condition 1, featuring PReLU and ELU, while yielding high rewards, was marked by slower task resolution and moderate variability. Condition 3, which integrated multiple activation functions, offered a middle ground, with a balance of task resolution speed and higher variability. Ultimately, Condition 2 emerged as the most effective, combining speed and stability in task performance.

In our DQN activation function study, Mixed AF (Tanh + ReLU) led the pack with swift task completion, high rewards, and demonstrating remarkable consistency. Gated OAF (GCU + SQU) claimed a close second for its quick learning and consistent rewards. The third-ranked Mixed OAF (GCU + SQU) offered a well-balanced learning and reward profile. Notably, ELU's fourth rank highlighted its high rewards and stability despite slower task resolution. In contrast, the SQU with and without a control parameter lagged, with the slowest learning and inconsistent results, suggesting the necessity for function adjustments.

DISCUSSION

Our study confirms that in specific control tasks, oscillating activation functions like GCU and SQU outperform traditional ones like Tanh. In the CartPole challenge, the symmetrical response of oscillatory functions proved essential for maintaining equilibrium, a key factor in their superior performance. Particularly noteworthy is the enhanced efficacy when GCU is paired with SQU in Mixed and Gated OAFs configurations, significantly outstripping the conventional ReLU function. These configurations, ranking second and third, demonstrate the benefits of oscillatory dynamics in DQN learning, enabling advanced gradient flows and improved exploration and stability in the action space.

MAFs, especially the Tanh and ReLU combination, emerged as top performers. This blend synergizes Tanh's normalization with ReLU's gradient preservation, resulting in accelerated, stable learning. The success of MAFs illustrates the potential of combining different activation functions to capitalize on their strengths and mitigate weaknesses.

The study also highlights the significant impact of adaptive and oscillating functions, as seen in the Gated OAF model. Ranking second, this model demonstrates the advantages of dynamic, training-responsive activation functions over static ones, underlining the importance of adaptability in complex learning environments.

However, our exploration of hierarchical structures indicates that their effectiveness is not always consistent and largely depends on the specific activation functions employed. Although promising for modelling complex data relationships, they do not uniformly outperform other methods.

Overall, the standout performance of the MAF, combining Tanh and ReLU, suggests the efficacy of blending different activation functions. This approach could be particularly beneficial in varied environments requiring diverse neural responses for optimal decision-making. The results advocate for a shift towards more complex and responsive neural architectures in DQN design, moving beyond traditional activation functions.

For future work, it is imperative to delve deeper into integrating a broader range of activation functions within hierarchical structures, assess their performance across different scenarios, and establish guidelines for optimizing these complex activation dynamics in deep reinforcement learning frameworks. This approach could further

enhance the adaptability and efficiency of DQN models in diverse real-world.

GitHub: https://github.com/lionggele/ReinforcementLearing-ActivationFunction

REFERENCES

[1] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," 2010. Available: https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

[2] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, doi: https://doi.org/10.1162/neco.2006.18.7.1527.

[3] G. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines A Practical Guide to Training Restricted Boltzmann Machines," 2010. Available: https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf

[4] M. M. Lau, "Review of Adaptive Activation Function in Deep Neural Network | IEEE Conference Publication | IEEE Xplore," *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/document/8626714 (accessed Dec. 13, 2023).

[5] M. M. Noel, A. L, A. Trivedi, and P. Dutta, "Growing Cosine Unit: A Novel Oscillatory Activation Function That Can Speedup Training and Reduce Parameters in Convolutional Neural Networks," *arXiv.org*, Jan. 12, 2023. https://arxiv.org/abs/2108.12943 (accessed Dec. 13, 2023).

[6] M. M. Noel, S. Bharadwaj, V. Muthiah-Nakarajan, P. Dutta, and G. B. Amali, "Biologically Inspired Oscillating Activation Functions Can Bridge the Performance Gap between Biological and Artificial Neurons," *arXiv.org*, May 10, 2023. https://arxiv.org/abs/2111.04020 (accessed Dec. 13, 2023).

[7] F. Manessi and A. Rozza, "Learning Combinations of Activation Functions," *arXiv.org*, Aug. 01, 2018. https://arxiv.org/abs/1801.09403 (accessed Dec. 13, 2023).

[8] "Educative Answers - Trusted Answers to Developer Questions," *Educative*. https://www.educative.io/answers/what-is-the-tanh-activation-function

[9] K. Pawar, "Tanh Activation Function," *InsideAIML*. https://insideaiml.com/blog/TanhActivation-Function-1032

[10] K. Arulkumaran, M. Deisenroth, M. Brundage, and A. Bharath, "IEEE SIGNAL PROCESSING MAGAZINE, SPECIAL ISSUE ON DEEP LEARNING FOR IMAGE UNDERSTANDING (ARXIV EXTENDED VERSION) 1 A Brief Survey of Deep Reinforcement Learning," Sep. 2017. Available: https://arxiv.org/pdf/1708.05866.pdf

[11] M. Gustineli, "A survey on recently proposed activation functions for Deep Learning," *arXiv.org*, Apr. 06, 2022. https://arxiv.org/abs/2204.02921

[12] Leon René Sütfeld, F. Brieger, H. Finger, S. Füllhase, and G. Pipa, "Adaptive Blending Units: Trainable Activation Functions for Deep Neural Networks," *Advances in intelligent systems and computing*, Jan. 2020, doi: https://doi.org/10.1007/978-3-030-52243-8_4.

[13] A. Hagg, "Evolving Parsimonious Networks by Mixing Activation Functions," *ar5iv*. https://ar5iv.labs.arxiv.org/html/1703.07122v1 (accessed Dec. 13, 2023).

[14] S. Qian, H. Liu, C. Liu, S. Wu, and H. S. Wong, "Adaptive activation functions in convolutional neural networks," *Neurocomputing*, vol. 272, pp. 204–212, Jan. 2018, doi: https://doi.org/10.1016/j.neucom.2017.06.070.

[15] A. Paske, "Reinforcement Learning (DQN) Tutorial — PyTorch Tutorials 1.8.0 documentation," *pytorch.org*. https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

[16] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark," *arXiv:2109.14545 [cs]*, Jun. 2022, Available: https://arxiv.org/abs/2109.14545

[17] "Gymnasium Documentation," *gymnasium.farama.org*. https://gymnasium.farama.org/environments/classic_control/cart_pole/

APPENDIX

Table 1. Activation Functions used in the RL Agent

Name	Equation
ReLU	$f(x) = \max(0, x)$
Tanh	$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
GCU	$f(x) = x \cdot \cos(x)$
SQU	$f(x) = x^2 + x$
PReLU	$f(x) = \begin{cases} ax, & x < 0 \\ x, & otherwise \end{cases}$
ELU	$f(x, a) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$

Table 1 has shown that several activation functions commonly used in Reinforcement Learning (RL) agents, along with their respective mathematical equations and formula. The graph above visualizes the behaviour of each activation function within a certain range of input values *x*. It illustrates how each function transforms its input to an output value.

Table 2: Hyperparameters in the agent:

Hyperparameter	Description	Value
num_trials	Number of trials	15
num_episodes	Number of episodes per trial	300
max_ep_len	Maximum episode length	300
leng_solved	Length of episode considered solved	200
batch_size	Batch size	128
num_nodes	Network layer width (number of nodes per layer)	128

gamma	Discount factor	0.99
eps_start	Starting value of epsilon for exploration	0.9
eps_end	Final value of epsilon for exploration	0.05
eps_decay	Rate of exponential decay of epsilon	1000
tau	Update rate of the target network	0.005
rl_lr	Learning rate for the AdamW optimizer	1e-3 (0.001)

Table 3. Experimental Design

Activation Function	Experiments
Traditional Activation Function (TAF)	*Tanh*
	PReLU
Oscillating Activation Function (OAF)	*Growing Cosine Unit (GCU)*
	Shifted Quadratic Unit (SQU)
Mixture Activation Function (concatenation)	*All FSAF: Tanh + ReLU*
	All OAF: GCU +SQU
	Both FSAF and *OAF: (Tanh, PReLu, GCU, SQU)*
Adaptive Activation Function	*Tanh* and *ReLU* ***(Gate Activation)***
	GCU and *SQU* ***(Gate Activation)***
	Tanh and GCU ***(Gate Activation)***
	PReLU and *ELU* ***(Hierarchical Structure)***
	Tanh and *GCU* ***(Hierarchical Structure)***
	All AF: (Tanh, RELU, GCU, SQU, PRELU, ELU) ***(Hierarchical Structure)***

FSAF = Fined Shape Activation Function, *OAF* = Oscillatory Activation Functions, *AF* = Activation Function.

Table 4: Different Activation Functions Evaluation in DQN

Method	Result							
	Solved at	**M.C. DR**	**M.C. AR**	**M.C. TR**	**M. std DR**	**M. std AR**	**M. std TR**	**Rank**
ReLU	33	81013	77432	158444	9.06	11.40	20.34	14
Tanh	**26**	**83518**	**80653**	**164171**	**5.84**	**7.29**	**13.04**	**5**
GSU (without learnable parameter)	**26**	**83068**	**79798**	**162866**	**9.26**	**11.26**	**20.40**	**11**
SQU (with control parameter)	**38**	**78183**	**72179**	**150902**	**19.11**	**42.11**	**42.11**	**15**
SQU (without control parameter)	36	56009	52126	108135	107.23	106.8	213.88	16
PReLU	30	81726	77881	159607	8.99	11.89	20.71	8
ELU	**27**	**83280**	**79665**	**162945**	**7.83**	**9.76**	**17.48**	**4**
Mixed AF (Tanh + ReLU)	**26**	**83449**	**80471**	**163919**	**5.04**	**6.56**	**11.51**	**1**
Mixed OAF (GCU + SQU)	29	82656	79379	162035	8.84	10.08	18.85	3
Mixed AFOAF (Tanh + GCU)	26	83439	80344	163784	7.44	9.01	16.38	10
Gated AF (Tanh + ReLU)	29	82322	78633	160955	10.28	13.0	23.15	7
Gated OAF (GCU + SQU)	**30**	**82450**	**79034**	**161484**	**6.74**	**8.47**	**15.13**	**2**
Gated AFOAF (Tanh + GCU)	26	82945	79656	162601	9.86	13.71	23.71	13
HAF (PRelu + ELU)	43	785842	75654	154196	10.54	12.3	22.75	12
HAF (Tanh + GCU)	**32**	**81534**	**78497**	**160031**	**8.6**	**10.21**	**18.72**	**6**
HAF (All)	39	79092	75147	154239	16.98	21.05	37.89	9

** AF = Activation Function, OAF = Oscillatory Activation Function, HAF = Hierarchical Activation Function, **Bolded = the outstadning pairs in during each comparison experiment**

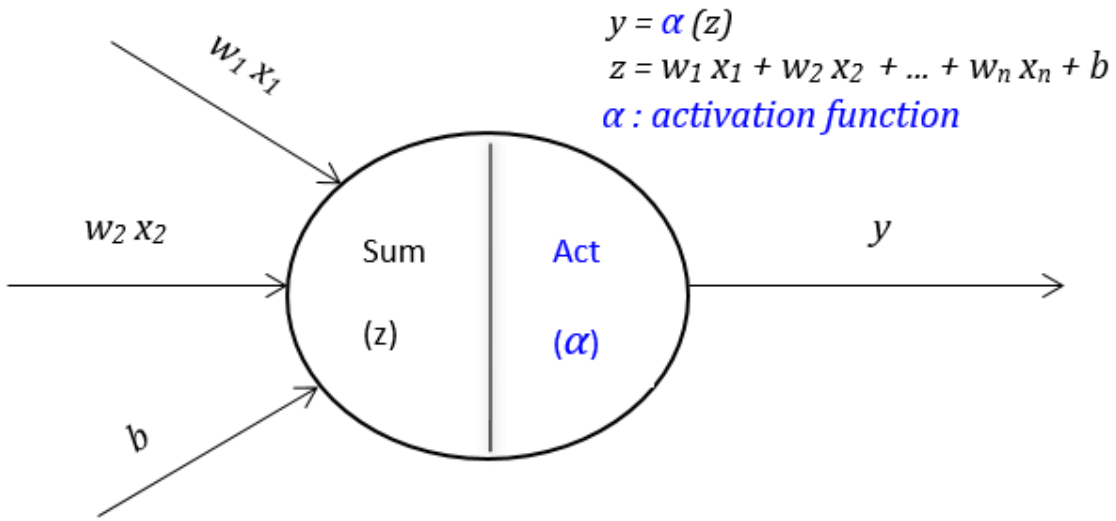

Fig 1: Artificial Neuron showcases the fundamental operation of an artificial neuron, where inputs x are weighted (*w*), summed with a bias (*b*), and passed through an activation function (α), resulting in the neuron's output (*y*).

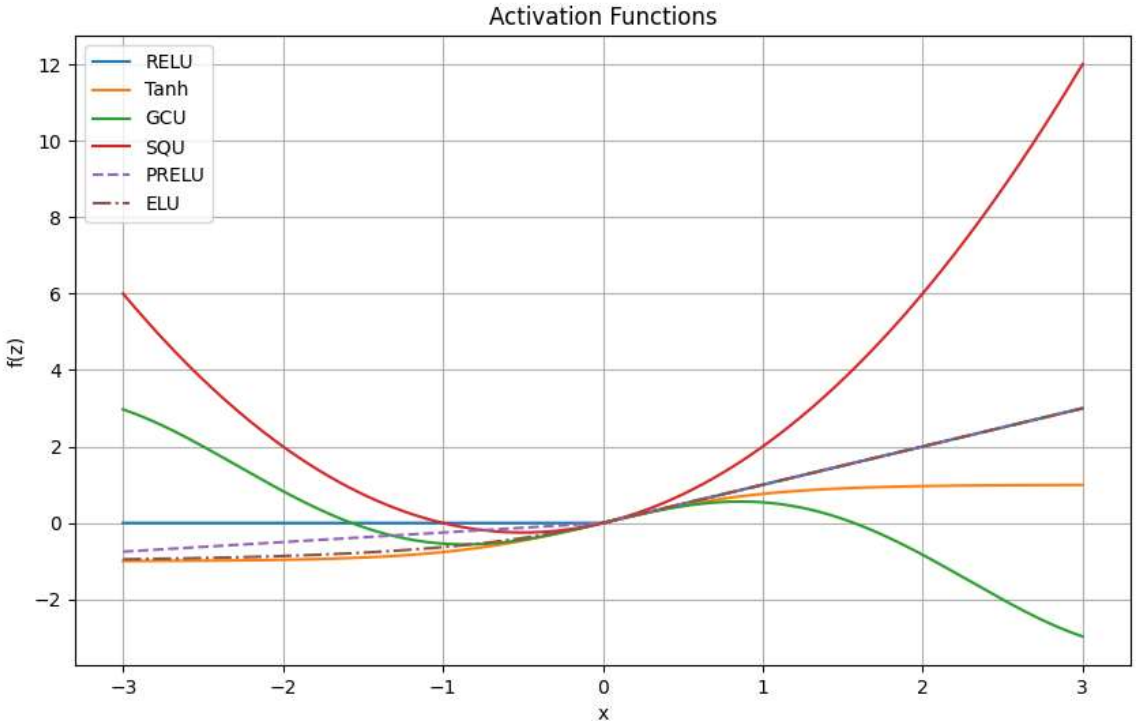

Fig 2: Graph of the behaviour of each activation function

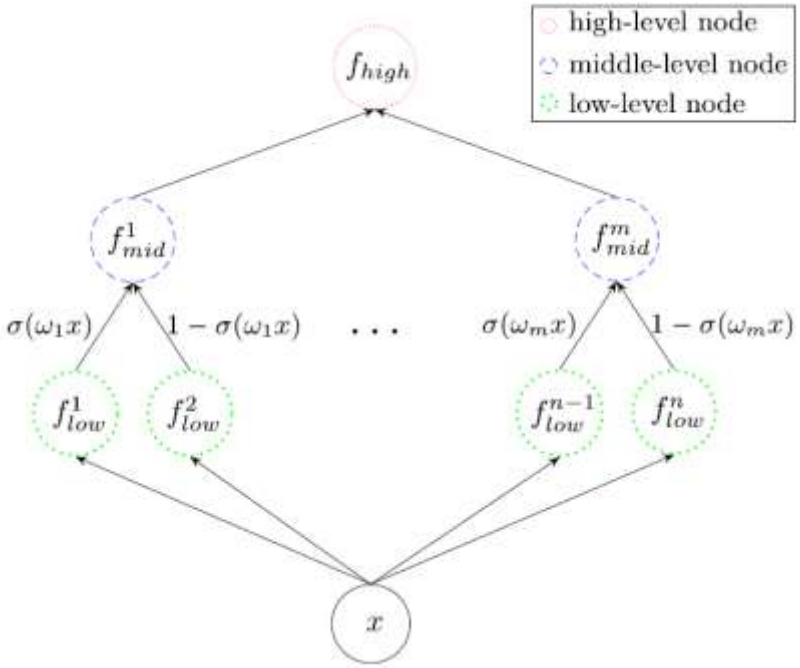

Fig 3: Hierarchical Activation Structure form low-level to high level.

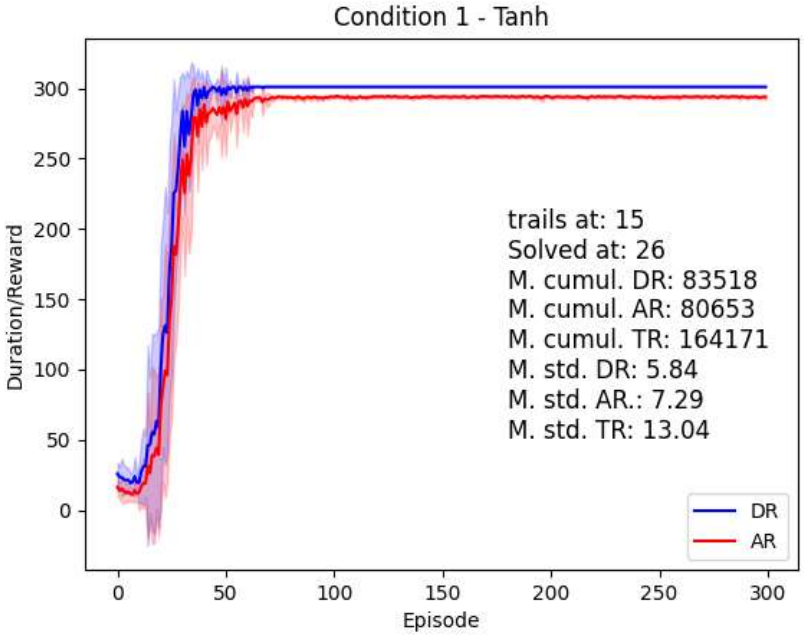

Fig 4: Graph of the *Tanh* Activation Function

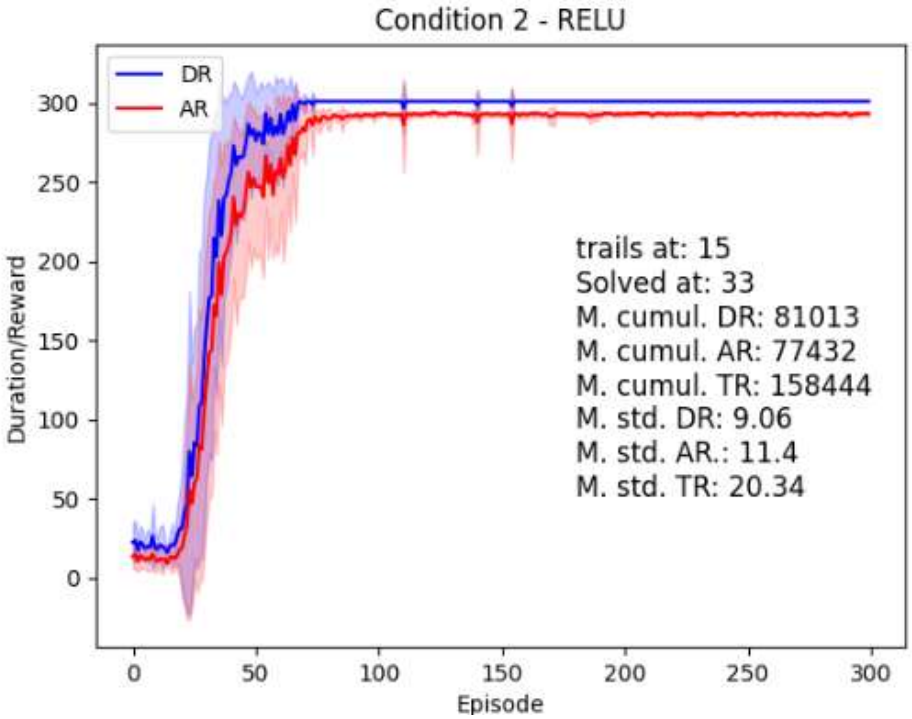

Fig 5: Graph of the *ReLU* Activation Function

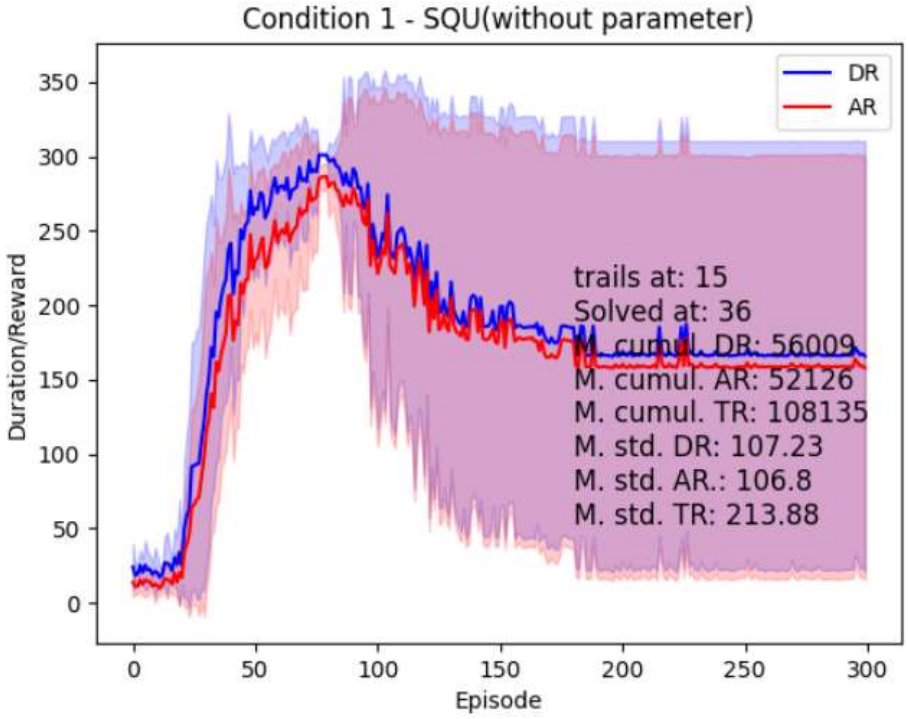

Fig 6: Graph of the *SQU(without parameter)* Activation Function

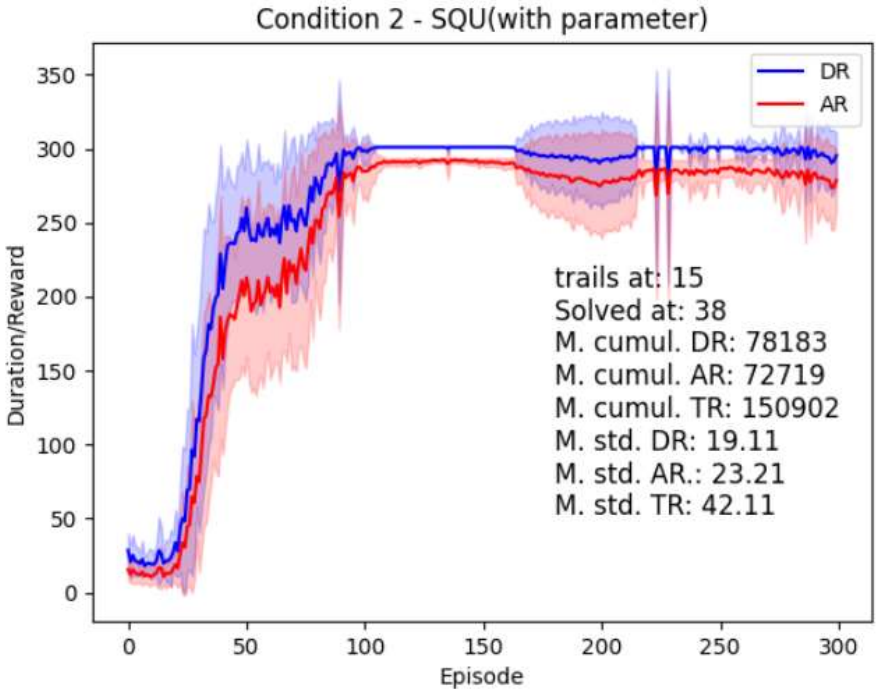

Fig 7: Graph of the *SQU (with parameter)* Activation Function

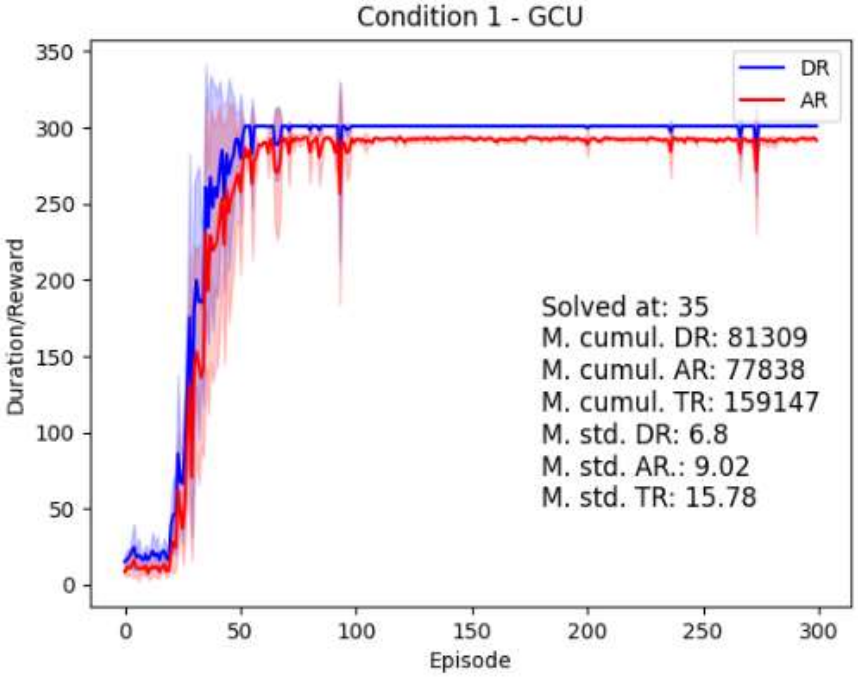

Fig 8: Graph of the *GCU* Activation Function

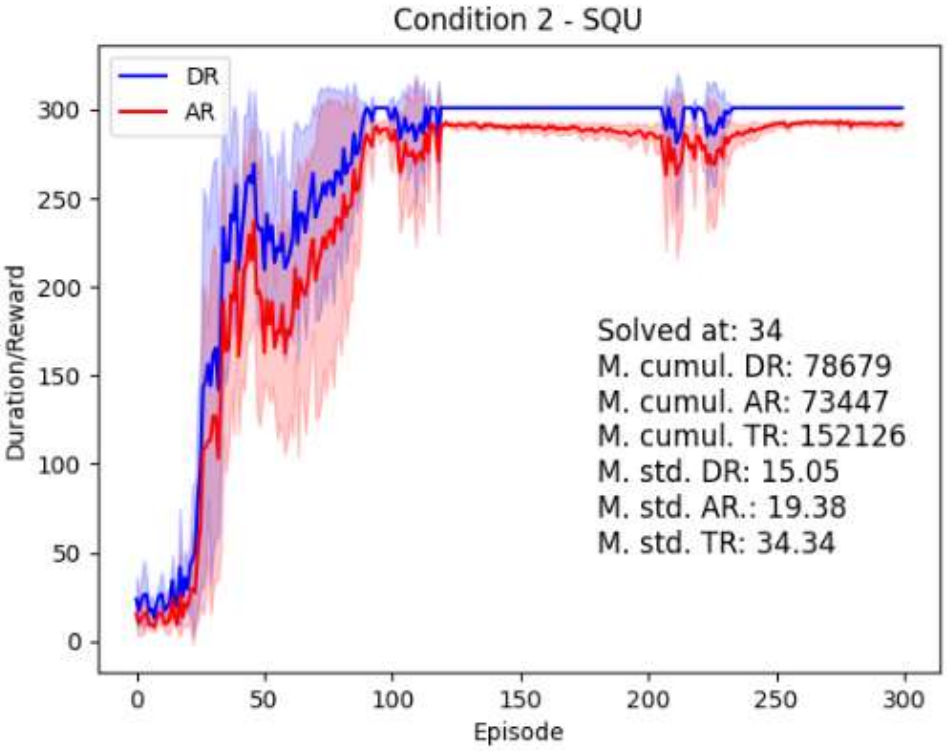

Fig 9: Graph of the *SQU* Activation Function

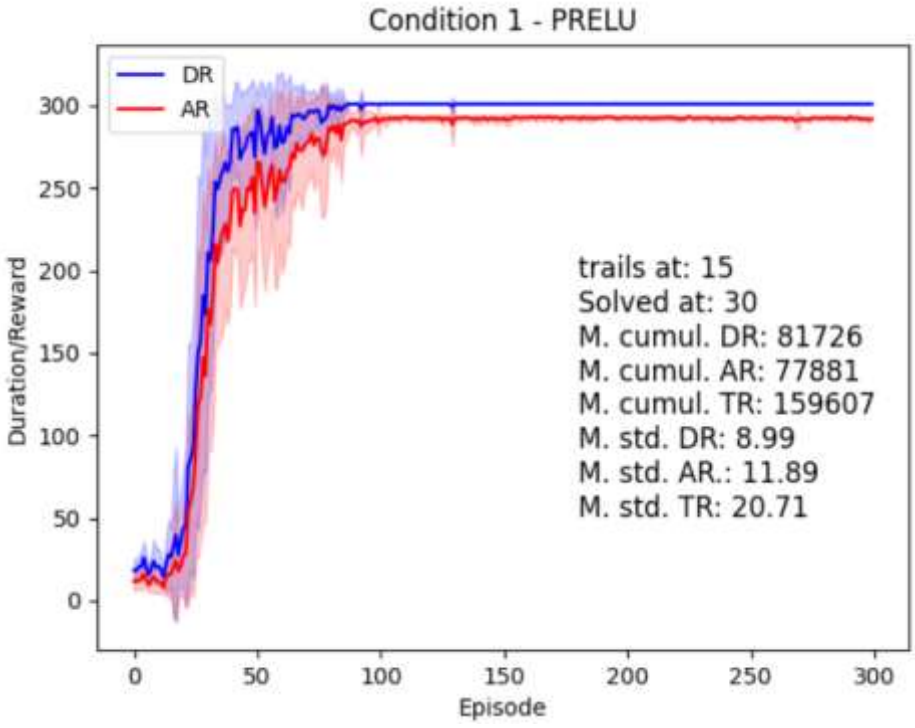

Fig 10: Graph of the *PReLU* Activation Function

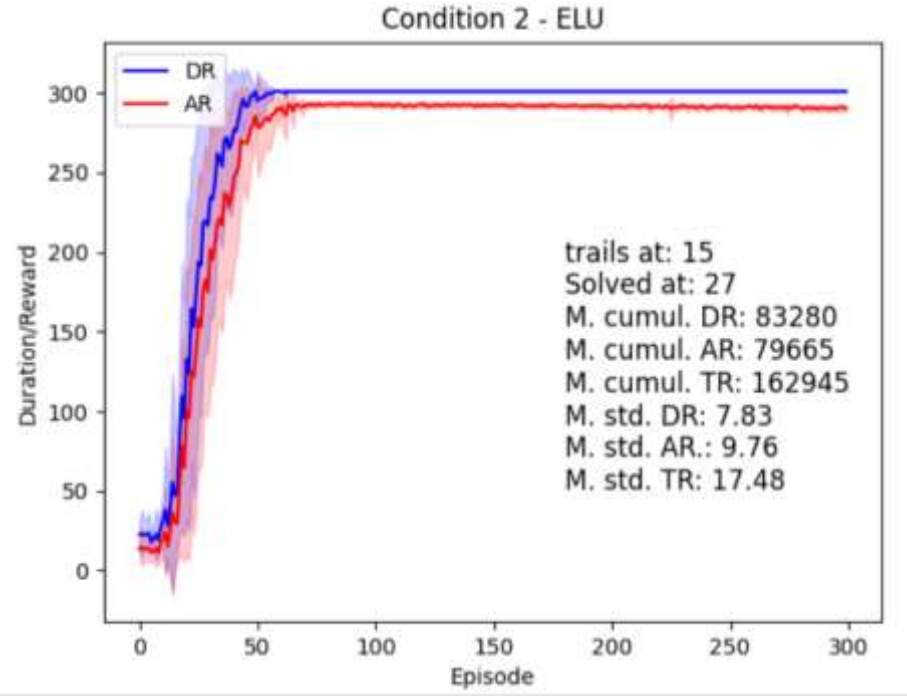

Fig 11: Graph of the *ELU* Activation Function

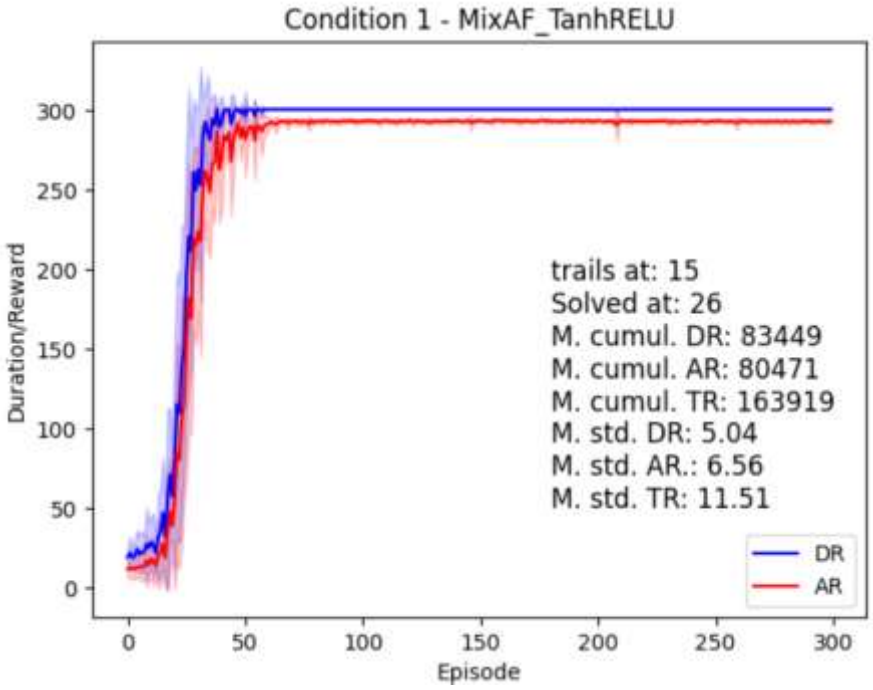

Fig 12: Graph of the *Mixed (Tanh & ReLU)* Activation Function

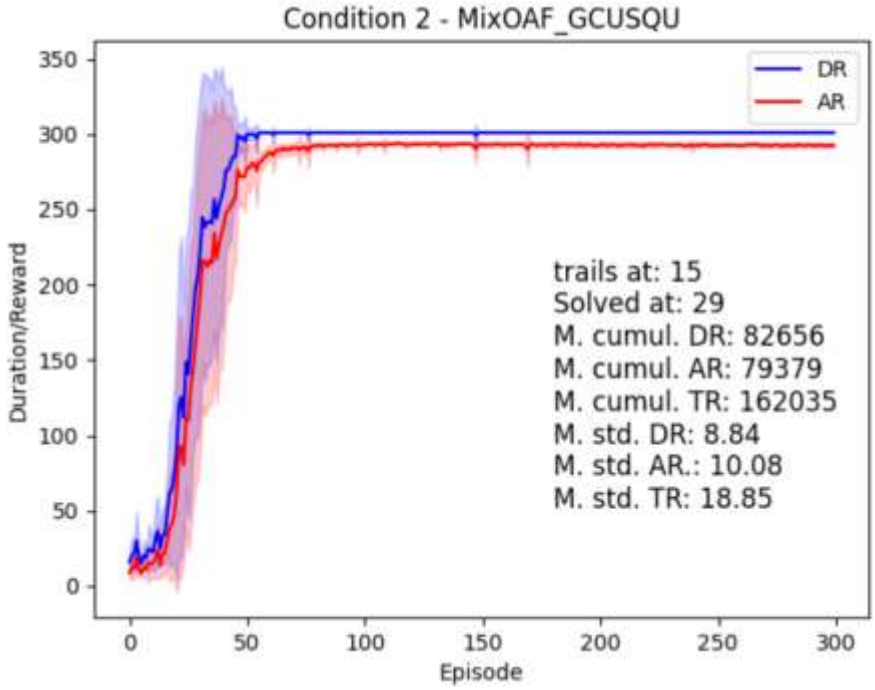

Fig 13: Graph of the *Mixed (GCU & SQU)* Oscillatory Activation Function

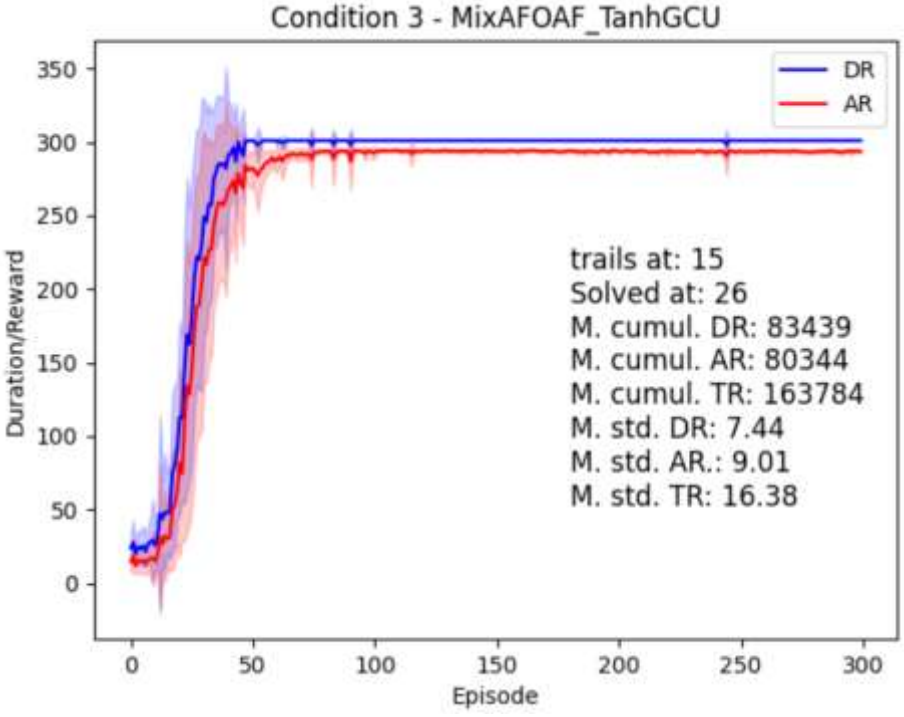

Fig 14: Graph of the *Mixed (Tanh & GCU)* Activation Function

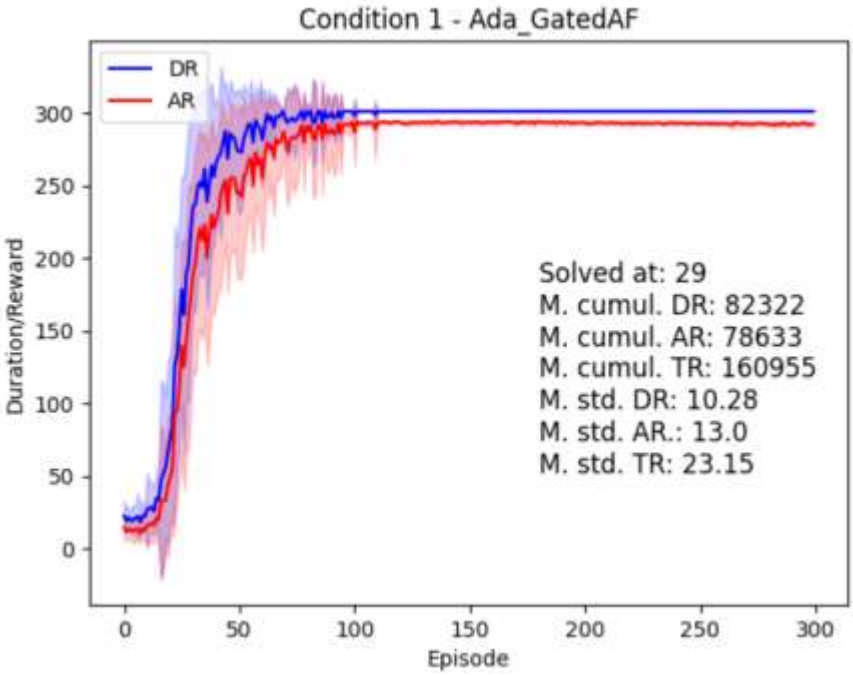

Fig 15: Graph of the *Gated (Tanh & ReLU)* Activation Function

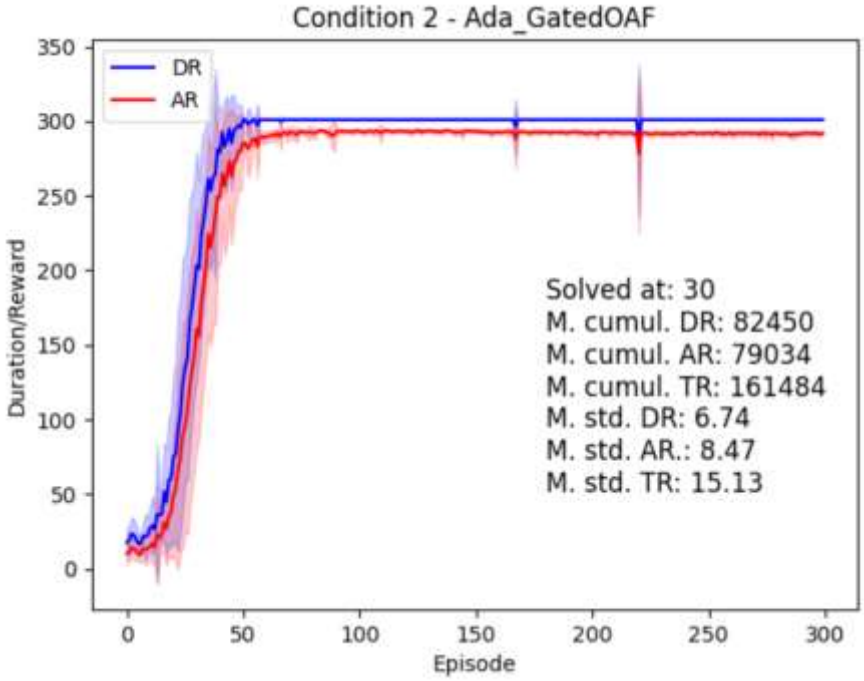

Fig 16: Graph of the *Gated (GCU & SQU)* Activation Function

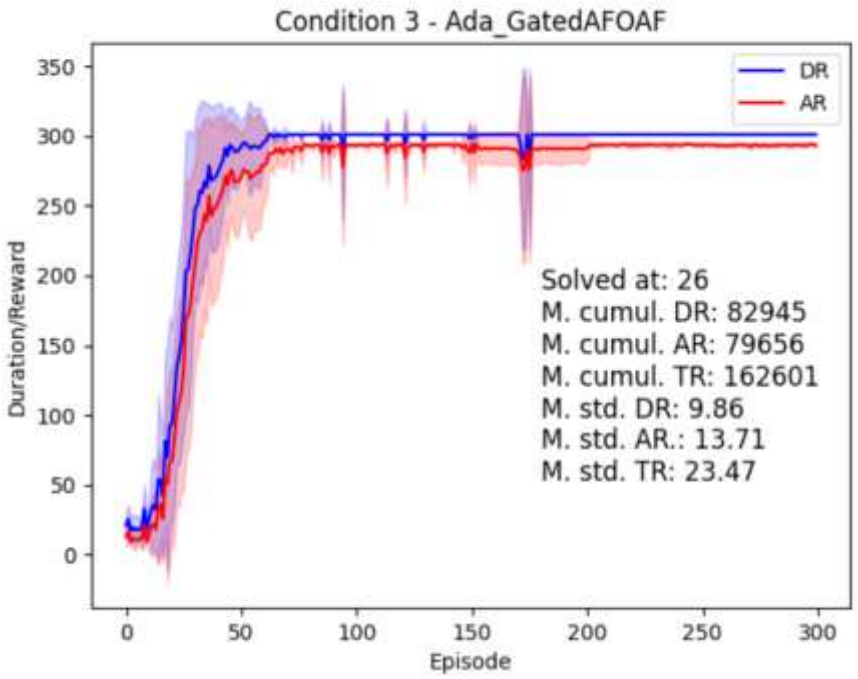

Fig 17: Graph of the *Gated (Tanh & GCU)* Activation Function

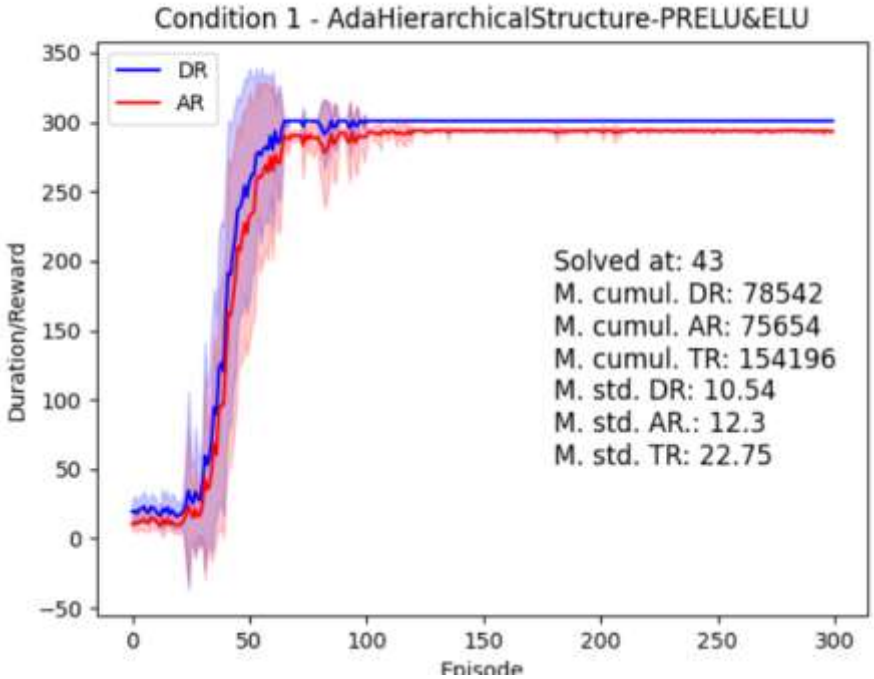

Fig 18: Graph of the *Adaptive Hierarchical Structure (PReLU & ELU)* Activation Function

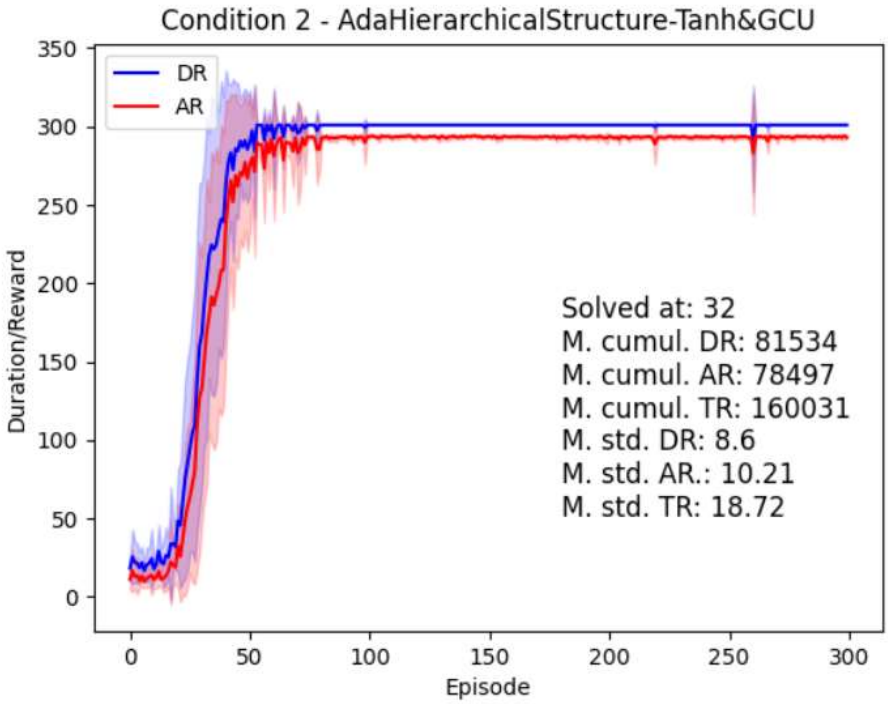

Fig 19: Graph of the *Adaptive Hierarchical Structure (Tanh & GCU)* Activation Function

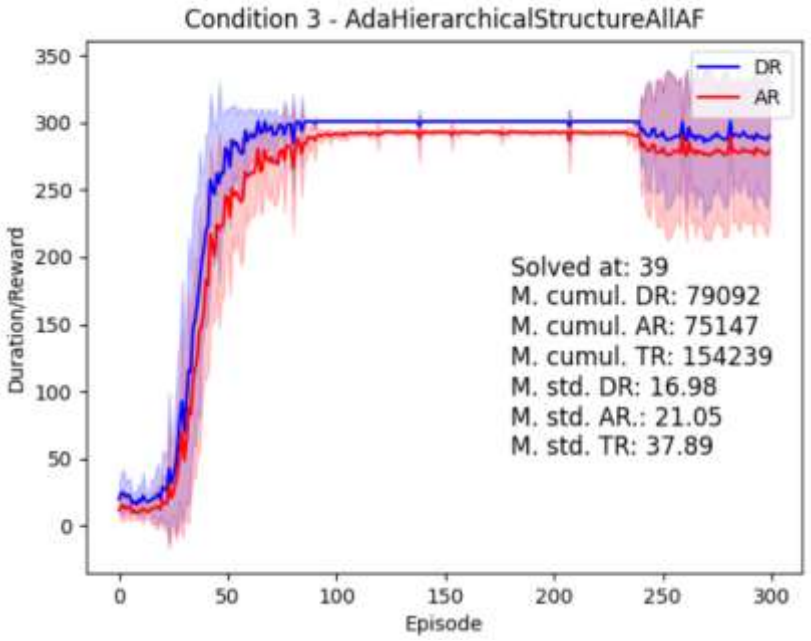

Fig 20: Graph of the *Adaptive Hierarchical Structure (All)* Activation Function