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

Abstract- This paper presents an empirical comparison of various activation functions in Deep Reinforcement Learning (DRL), focusing on Oscillating, Mixture, and Adaptive Activation Functions within Deep Q-Network (DQN) frameworks. The study explores the impact of these advanced activation functions on the learning efficiency and stability in DRL, using the "CartPole-v1" environment from Gymnasium as a testbed. Traditional Fixed Shape Activation Functions (FSAFs) like ReLU and Tanh are compared with Oscillatory Activation Functions (OAFs) such as Growing Cosine Unit (GCU) and Shifted Quadratic Unit (SQU), as well as Mixture and Adaptive Activation Functions. The experiments demonstrate the effectiveness of these novel activation functions in enhancing learning speed, reward accumulation, and consistency in DRL tasks. The findings suggest that oscillating and adaptive activation functions, particularly when combined in hierarchical structures or gated mechanisms, can significantly improve DRL performance, offering a nuanced understanding of activation dynamics in neural network architectures.

Keywords—List frequently used words in this document.

I. Introduction

Deep Reinforcement Learning (DRL) combines the decision-making process of Reinforcement Learning (RL) with the representational capabilities of Deep Neural Networks (DNNs) [10]. Its applications range from mastering complex games to advancing autonomous systems. At the heart of DRL lies the ability of DNNs to approximate functions, which is crucial for mapping states to actions and predicting potential rewards.

DNNs have revolutionized computational models, mirroring the intricate processing capabilities of the human brain through layered networks of artificial neurons. These networks, armed with the power to assimilate raw data and attune to a multitude of real-world contexts, have seen accelerated progress, propelled by computational and algorithmic advancements. While they excel in tasks ranging from pattern recognition to predictive analytics, the journey of training DNNs is fraught with challenges, notably the vanishing gradient problem and the saturation of activation functions during the training process [1]. Hinton proposed a pre-training algorithm to eliminate the vanishing gradient problem happened during the backpropagation training process [2]. DNN was layer-wise pre-trained using Restricted Boltzmann Machine [3] and then fine-tune with backpropagation. This method provided a better initialization for weight parameter in the DNN [4].

Besides re-initialize the weights parameter properly, the choice and behaviour of activation functions are pivotal. These functions, determining the firing rate of artificial neurons, introduce non-linearity and play a crucial role in the learning process. Activation functions in neural networks can be broadly categorized based on their characteristics and effects on the training dynamics. Activation functions can be divided into 4 categories, Fixed Shape Activation Functions [5], Oscillatory Activation Functions [6], Mixture Activation Functions and Adaptive Activation Functions [4].

Fixed Shape Activation Functions are all the classic rectifier-based activation functions used in neural networks such as step function, sigmoid, tanh, RELU, LRELU, etc fall under this category [5]. They are static and do not adapt or change during the training process of a neural network. It has often been used for decision boundaries in the early of ANN development. Unlike the fixed Shape Activation Function that have a fixed or monotonous pattern, Oscillatory Activation function exhibits a repeating, wave like-pattern. These functions have a periodic nature, meaning they repeat their values in regular intervals. This paper has proposed 4 new oscillatory activations that enable individual artificial neurons to lean the XOR function like biological neurons [6]. Growing Cosine Unit (GCU) has been proposed with the advantages of using Oscillatory functions to improve gradient flow and alleviate the vanishing gradient problem. Mixture Activation Function is a type of activation function that combines multiple base activation functions through a weighted sum or another mixing method. This allows the activation function to learn a more complex and flexible non-linearity compared to traditional single-base activation functions. The paper "Learning Combinations of Activation Functions" by Li et al. (2018) provides theoretical and empirical support for the benefits of mixture activation functions. Adaptive Activation Function have parameters that are learned during training, allowing the function to dynamically adjust its behaviour based on the data it encounters. This makes them inherently adaptable to the training process.

This paper seeks to explore beyond the conventional activation dynamics by empirically comparing the performance of oscillating, mixture, and adaptive activation functions within DQN frameworks. We evaluate their effectiveness across various control tasks, seeking to provide insights into their benefits and limitations. Our goal is to furnish a nuanced understanding of how these activation functions impact DRL and guide future research towards more sophisticated and capable neural architectures and any robotic system.

II. BACKGROUND

Activation Function

An artificial neural network (ANN), comprised of a network of artificial neurons, utilizes activation functions in each neuron to determine their activation levels. These activation functions are critical as they introduce non-linearity to the network and limit the input data within a specific range This concept draws inspiration from the biological neurons found in the human central nervous system. A biological neuron, as illustrated in Fig 1, serves as a fundamental data-processing unit in the nervous system. Fig 2 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). This process enables the neuron to engage in complex pattern recognition and learning.

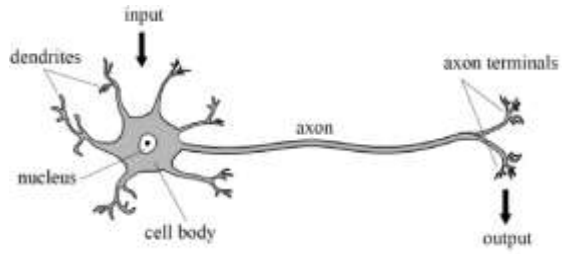

Fig 1: Biological Neuron

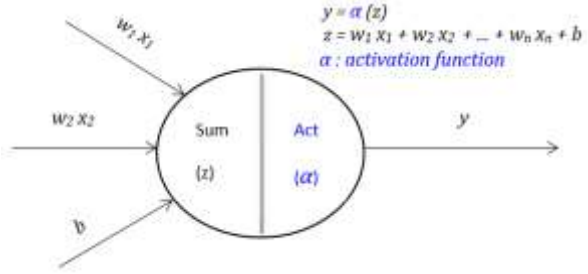

Fig 2: Artificial Neuron

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	$f(x) = \begin{cases} ax, & x < 0 \\ x, & \text{otherwise} \end{cases}$
ELU	$f(x, a) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$

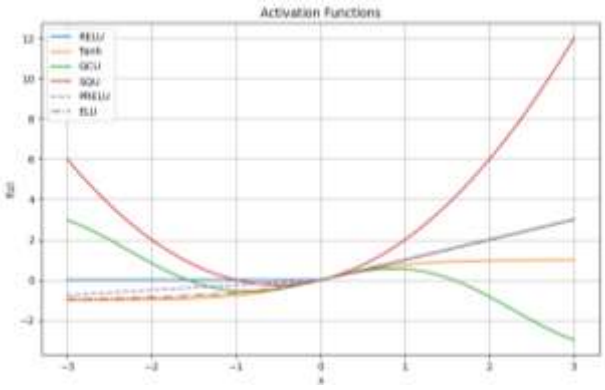

Fig 3: Graph of the behaviour of each activation function

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.

1. Hyperbolic Tangent (Tanh)

The hyperbolic tangent (Tanh) function, characterized by its S-shaped curve, has been historically favoured for its ability to model non-linear phenomena. It ranges between -1 and 1, which mean it is a zero-centred function., hence, reducing the number of epochs needed to be train the network as compared to any sigmoid function. However, Tanh suffers from vanishing gradient problems, particularly in deep networks, which hampers the network's ability to learn from data that has gradients approaching zero. [9]

2. Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) emerged as a potent alternative due to its computational simplicity and its ability to mitigate the vanishing gradient issue for positive inputs. Its half-rectification property—outputting zero for all negative inputs and a linear response for positive inputs—has been shown to accelerate convergence and promote sparse representations within the network. However, ReLU is not without its limitations, particularly the "dying ReLU" problem, where neurons can become inactive and cease to contribute to the model's learning process.

3. *Growing Cosine Unit (GCU) and Shifted Quadratic Unit (SQU)*

Growing Cosine Unit (GCU) and Shifted Quadratic Unit (SQU) [6] represent oscillating activation functions that introduce higher-order non-linearities into the network. These functions can enhance the model's expressiveness and capture complex patterns within the data. GCU utilizes the cosine function to modulate the linear response, while SQU employs a quadratic function to emphasize the contribution of inputs with larger magnitudes.

4. Parametric ReLU (PreLU)

Parametric ReLU (PReLU) offers a solution to the dying ReLU problem by introducing a small, positive slope in the negative domain, parameterized by α which is a learnable parameter, x is the input to the activation function. This allows for gradient-based learning even when the inputs are negative, potentially leading to more consistent predictions across the input space. [11]

5. Exponential Linear Unit (ELU)

The Exponential Linear Unit (ELU) extends the idea of PReLU by incorporating an exponential function in the negative domain [11]. This creates a smooth, non-zero gradient that saturates for large negative inputs, providing a balance between the benefits of ReLU and the need to mitigate the dying neuron phenomenon. The scaling parameter α dictates the saturation point for negative inputs, directly influencing the network's robustness to variations in the input data.

Mixture Activation Function

Mixture activation functions, a novel concept, propose combining the properties of different activation functions to leverage their unique advantages [7]. The hypothesis is that such a hybrid approach could yield a more flexible and robust model capable of adapting to diverse datasets and learning scenarios. There are some studies provided the similar concept. In this paper, it is called Adaptive Blending Units (ABUs), a trainable linear combination of a set of activation functions [12]. A novel methods for learning mixed activation functions within neural networks has been proposed, yielding a 3.01% top-1 accuracy boost for AlexNet on ILSVRC-2012, showcasing enhanced model efficacy [13]. S.Qian has suggested that mixed activation can be done through concatenation with basic activation functions. The mixed activation function combines LReLU and ELU, formulated as $f_{\text{mix}}(x) = p \cdot f_{\text{lrelu}}(x) + (1-p) \cdot f_{\text{elu}}(x)$. The coefficient p ranges from 0 to 1, allowing for a tailored blend of these activations [14].

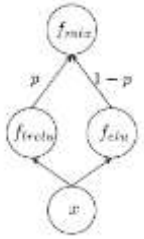

Fig 5: Mixed sample: LReLU-ELU activation

Adaptive Activation Function

Gated activation

In recent advancements, gated activation has emerged as a key concept in neural network architecture, leveraging the combination of different activation functions to create a dynamic system that adapts during the learning process. This gated mechanism, often represented as a weighted sum of functions like LReLU and ELU, adjusts the activation based on the inputs and a gating parameter, leading to an adaptable and data-driven response. This is the sample structure of gated activation which has been proposed by S Qian [14].

$$f(x) = \sigma(\omega x) \cdot f_{relu}(x) + (1 - \sigma(\omega x)) \cdot f_{tanh}(x)$$

Hierarchical Activation

In the exploration of advanced neural network architectures, hierarchical activation emerges as a sophisticated approach designed to enhance the network's capacity to capture complex patterns in data. Unlike traditional activation functions with predefined parameters, hierarchical activation allows for the functions themselves to be learned, adapting more dynamically to the inputs they receive.

The concept of hierarchical activation is rooted in organizing basic activation functions into a multi-tiered structure, reminiscent of a binary tree, where the learning process is fine-tuned at varying levels of abstraction. Low-level nodes in this hierarchy are directly associated with learnable activation functions, providing the foundational transformations. Each middle-level node corresponds to an output which is the aggregate of a pair of low-level nodes, determined through a learned gating mechanism. This gating mask guides the combination process, ensuring that the activation function adapts to the incoming data more precisely [14].

AS shown in Fig 6, at the highest level, the activation operation is governed by the winner-take-all principle, where the maximum value across all middle-level nodes is selected, yielding the final output of the hierarchical activation structure. This method ensures that the most significant features from the middle levels are carried forward, enhancing the network's decision-making capability.

$$f(x) = \begin{cases} f_{\text{low}}^{m}(x) & \text{if low-level nodes} \\ \sigma(\omega x) f_{\text{mid}}^{m,\text{left}}(x) + (1-\sigma(\omega x)) f_{\text{mid}}^{m,\text{right}}(x) & \text{if middle-level nodes} \\ \max\limits_{m\in\{1,2\}} f_{\text{mid}}^{m}(x) & \text{if high-level node} \end{cases} \quad (12)$$

Fig 6 Hierarchical Activation Structure form low-level to high level.

The hierarchical activation structure offers a significant advantage in that it is fully differentiable, meaning the entire hierarchy of activation functions can be trained end-to-end alongside the deep network. This level of integration between the activation functions and the network's architecture allows for a more nuanced and refined learning process, as the network can develop its own custom activation responses based on the specific demands of the data it encounters.

]

III. METHDOLOGY

Environment and Experiment Design

The "CartPole-v1" environment from Gymnasium is used to evaluate different activation functions within a DQN (Deep Q-Network) framework in reinforcement learning (RL). The objective in "CartPole-v1" is to balance a pole on a cart by applying forces left or right. The action space of this environment consists of two actions: moving the cart left or right. The state of the environment includes four observations: cart position and velocity, plus pole angle and tip velocity. The agent earns a reward each timestep the pole remains upright, with an episode ending if the pole tilts too far, the cart strays off limits, or a set number of timesteps elapse. Reward shaping provides additional incentives for keeping the pole more vertical angle -- meaning the agent receives more reward for keeping the pole more vertical (closer to zero degrees). The shaping function adds a scaled reward based on the absolute angle of the pole, encouraging the agent to not only keep the pole from falling but to minimize its angle of deviation. Thus, promoting a policy where the pole stays as upright as possible, enhancing both learning stability and agent performance. For a more detailed overview of each environment, refer to the OpenAI Gym documentation.

In our experimental design, we aim to redefine the activation dynamics with accessing different activation functions on the training speed and stability of a Deep Q-Network (DQN) in reinforcement learning. We consider CartPole to be solved when the agent reached 300 steps and can keep the pole balanced for 200 consecutive timesteps in a single episode.

Next, we trained models, which will be further elaborated on in the next subsection, with varying numbers of parameters in the environments with 15 trials. We recorded the first number of episodes that reached 300 timesteps. We calculated and recorded the mean cumulative duration the pole is balanced (Duration Reward) and the mean degree to which the pole remains vertical which is the reward (Auxiliary Reward) will be recorded after the model trained. The design seeks to determine if the choice of activation function significantly affects these metrics. This is assessed using statistical tests, like the Mann-Whitney U Test, to compare the cumulative rewards across different activation functions. In Training Strategy, we will discuss more about the Experimental design and its strategies.

DQN model

Our experimental framework utilizes a base DQN model code developed in PyTorch by Adam Paszke [15]. This code serves as a robust and well-structured foundation, enabling a focused exploration into the realm of activation function dynamics. Additionally, the plotting functions and overall structure of the experiment draw inspiration from the teachings of Dr. Tomas, further enriching the study's methodological approach.

The core function 'run_experiments' orchestrates a series of trials designed to evaluate the efficacy of diverse activation functions within the neural network architecture of the DQN. This assessment aims to elucidate the impact of these activation functions on learning efficiency and stability in the context of the pole-balancing task

For consistency and control in our experimental setup, we have maintained a fixed set of hyperparameters, detailed in the accompanying table. Central to our investigation is the implementation of a variety of activation functions, including standard ones like ReLU and Tanh, as well as other novel functions. This varied approach is pivotal in our quest to gain deeper insights into how different activation dynamics influence the learning trajectories in DQNs, ultimately guiding us towards a more profound understanding of activation function roles in deep reinforcement learning scenarios.

Table . 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)

a. Training Strategy

In our study, "Redefining Activation Dynamics: An Empirical Comparison of Oscillating, Mixture, and Adaptive Activation Functions in Deep Reinforcement Learning," we explore a range of activation function strategies within Deep Q-Network (DQN) architectures. Our approach is informed by insights from the paper “A Comprehensive Analysis of Deep Learning Activation Functions” , 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 fixed-shape activation functions (ReLU, Tanh), two oscillatory types (GCU, SQU), and two adaptive varieties (PReLU, ELU). This allows us to conduct a thorough comparison within and across these distinct activation function types. A key aspect of our research is the investigation of mixed activation strategies. Here, we combine different activation functions in a single network. For

instance, a mixed activation function might be represented by the 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, our study delves into gated activation functions, an innovative concept highlighted in the referenced paper from "Adaptive activation functions in convolutional neural networks". Unlike mixed activations with fixed coefficients, gated activations adaptively learn a gating mask, offering a dynamic interplay between the inputs and activation functions. An example of a gated activation function:

$$f(x) = \sigma(\omega x) \cdot f_{relu}(x) + (1 - \sigma(\omega x)) \cdot f_{tanh}(x)$$

In this equation, σ denotes the sigmoid function and ω is the learnable gating mask.

Another experiment will focus on the hierarchical structure. A hierarchical activation framework in a deep Q-network, deploying a diverse set of activation functions—ReLU, ELU, PReLU, Tanh, GCU, and SQU—at each layer. These functions, spanning fixed to adaptive types, are unified through a winner-take-all mechanism that selects the optimal activation response per input. Initially, the network linearly processes inputs, then channels them through the six activations. The resulting activations are then stacked and subjected to a maximum selection operation, resulting in a layer output that takes forward only the most responsive activations. This approach is replicated in the second layer, enhancing the adaptability of the activations. A Winner-take-all Strategy is used to select the dominant activations. The final layer is a simple linear transformation that takes the refined activations from the second layer to produce the action values required by the deep Q-network.

In our study, we plan to rank the performance of various activation functions in Deep Q-Networks based on three key factors: how quickly the models learned the task, their ability to gather rewards, and the consistency of their performance. A model that learned the task in fewer episodes was considered to learn faster and was therefore ranked higher. We also looked at the total rewards each model accumulated during the learning process, with more rewards pointing to a more efficient model. Finally, we wanted models that performed reliably across multiple trials, so we preferred those with lower standard deviations in their reward metrics. By balancing these factors—speed of learning, reward efficiency, and consistency—we determined each model's overall rank, giving us a clear picture of which activation functions led to the most effective learning in DQNs.

For the most of hyperparameters (such as learning rate schedule, weight, etc), we follow the settings released by original works. After configuring models, we train the DQN models with the regarding the activation functions that needed for the experiments.

Table 2. 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.

RESULT AND DISCUSSION

a. Comparison of Fixed Shape Activation Function and Oscillatory Activation Function

We make a comparison between fixed shape activation function. We use DQN with activation function (*ReLU*, *Tanh*) for Fixed Shape activation function. The experimental results in **Table 3, Fig 7** and **Fig 8** demonstrate Tanh leads in quick learning and high reward acquisition, ReLU demonstrates a steadier and more reliable performance. The network with Tanh Activation concluded its learning task by episode 26 compared to ReLU which is at later episodes 33, suggesting a quicker adaptation early on, as evidenced by a higher mean cumulative Duration Reward (DR) of 83,518, which overshadows the ReLU's 81,013.

This early performance peak of Tanh is further underscored by its higher Average Reward (AR) and Total Reward (TR), the latter peaking at 164,171, indicating that the Tanh network not only learned quickly but also accumulated rewards efficiently. However, the ReLU network notably exhibited lower standard deviations for DR (9.06 compared to Tanh's 7.89) and AR (11.4 compared to Tanh's 5.24), signifying a more uniform performance across trials and a steadier learning progression. Overall Tanh is outperformed ReLU. The potential reasons why Tanh outperformed ReLU in this scenario could relate to the nature of the CartPole environment were having both positive and negative activations could provide a more nuanced control

strategy that ReLU's non-negative output range may not offer. The Tanh function's output range of -1 to 1 can provide a more balanced and fine-tuned control signal compared to ReLU's 0 to infinity range. ReLU could lead to aggressive responses where a more subtle adjustment is needed. Tanh's negative activations can effectively counterbalance positive activations, which might help the model learn a more stable policy for balancing the pole. This symmetry in activations can be particularly advantageous in tasks that require maintaining equilibrium, like CartPole.

Next, we examined two conditions of Oscillatory Activation Functions, Growing Cosine Unit, (GCU) and Shifted Quadratic Unit, (SQU). Before comparing both conditions, SQU has been modified to enable the stability of learning by employing the Shifted Quadratic Unit (SQU) activation function, with and without an additional scaling parameter.

The SQU activation function in its fundamental form, expressed as $f(x) = x^2 + x$ which proposed by this paper "Biologically Inspired Oscillating Activation Functions Can Bridge the Performance Gap between Biological and Artificial Neurons" was used in the DQNSQU1 model in the code. Due to the instability of result which shown in **Table 3, Fig 9** and **Fig 10**This leds to significant variability in learning, as evidenced by wide confidence intervals in DR and AR. The network's slower learning curve culminated in a solution at episode 36, with comparatively lower mean cumulative rewards and higher standard deviations, suggesting less consistency in learning outcomes.

Condition 2's introduction of the α parameter within the SQU function seemed to regulate the activation values effectively, which is corroborated by the smoother learning curves and lower standard deviations. The DQNSQU2 model employed an augmented version, $f(x) = \alpha(x^2) + x$, where α is a parameter tailored to modulate the expansion of the quadratic term. This parameter acts to temper the nonlinearity of the quadratic term, potentially averting excessive activation values which could destabilize the learning process. In the result, the mean cumulative rewards for DR and AR were higher, indicating improved network performance. Additionally, the standard deviation values were substantially reduced, underscoring a more reliable and consistent learning process.

The moderated SQU activation maintains the beneficial aspects of quadratic activations—such as nonlinearity—while mitigating risks associated with unbounded output values. This evidence positions the parameterized SQU as a preferable choice for stable and effective deep reinforcement learning model training.

After getting a moderated SQU, we will perform an experiment by employing Growing Cosine Unit (GCU) and Shifted Quadratic Unit (SQU) activations. This GCU activation resulted in a stable learning curve, reaching task completion by episode 35 with low variability in performance metrics. This consistency is evident from the standard deviations in Duration Reward (DR) and Average Reward (AR), which stood at 6.8 and 9.02, respectively. Although it recorded commendable cumulative rewards, the total reward did not reach the highest potential, suggesting room for improvement. Conversely, Condition 2, featuring SQU activation, achieved the task slightly faster by episode 34 but at the cost of greater volatility in learning outcomes, as indicated by higher standard deviations which is DR: 15.05; AR: 19.38. The quicker resolution time was overshadowed by the erratic nature of the learning process, which was less consistent and displayed cumulative rewards which is DR: 78,679; AR: 73,447 marginally lower than those of the GCU condition.

So, it becomes apparent that the GCU activation underpins a more predictable and stable learning paradigm which shown in the **Fig 11,** likely attributable to its cosine modulation that may avert common pitfalls like gradient vanishing. On the other hand, the SQU activation as shown in **Fig 12**, while slightly quicker in task resolution, displayed increased volatility due to its quadratic nature, which can introduce more variable gradient responses. This comparative analysis underscores the significance of a deliberate selection of activation functions to tailor and optimize deep Q-network performance. GCU has become the choice of Oscillatory Activation Function for further experiments.

b. Comparison of Adaptive Activation Function

Next up, we have an experiment to be compared in different adaptive activation functions (PRELU and ELU). Our evaluation of activation functions in deep Q-networks (DQNs) compares the Exponential Linear Unit (ELU) with the often-preferred Parametric Rectified Linear Unit (PReLU). Based in **Table 3, Fig 13** and **Fig 14**, ELU surpassed PReLU in some key areas. It achieved task resolution in fewer episodes and provided slightly higher mean cumulative rewards for Duration Reward (DR) and Average Reward (AR), pointing to a potential for faster learning and greater efficiency.

ELU also demonstrated lower standard deviations, especially in Total Reward (TR), indicating a more consistent and stable learning trajectory. This consistency might stem from ELU's design, which mitigates the vanishing gradient problem and remains robust to noise for negative inputs. Such attributes could contribute to a more balanced nonlinearity and stability during training.

Despite these findings favouring ELU in our specific setup, it's important to recognize the situational efficacy of PReLU. Its parameterized adaptability is often essential for optimizing the learning process across various tasks and environments. In sum, our results suggest that ELU could be the more advantageous choice for certain DQN scenarios, offering quicker convergence and reward stability. However, this does not diminish PReLU's utility in contexts where its adjustable nature is beneficial.

c. Comparison of Mixture Activation Function

We will explore the next experiment into Mixed Activation Functions (MAFs) within deep Q-networks (DQNs) has yielded informative results as shown in **Table 3, Fig 15, Fig 16** and **Fig 17**.

In **Fig 15,** Condition 1, employing a Tanh and ReLU MAF, completed the task in 26 episodes, showcasing the most consistent learning curve, as evidenced by the lowest standard deviation values across Duration Reward (DR), Average Reward (AR), and Total Reward (TR). This MAF appears to effectively blend Tanh's output normalization with ReLU's gradient-preserving qualities, enhancing both learning efficiency and stability. In **Fig 16,** Condition 2's combination of GCU and SQU activation functions achieved task resolution by episode 29. Despite the slightly delayed completion, this condition presented increased variability in performance, indicated by higher standard deviations. The GCU's periodic modulation alongside SQU's quadratic transformation likely introduces a higher degree of learning complexity. In **Fig 17,** Condition 3, integrating Tanh with GCU, paralleled the episode completion rate of Condition 1 but with marginally elevated standard deviations. This suggests that while Tanh's normalization effects are beneficial, the additional modulation by GCU introduces a slight increase in learning variability.

The experiment's outcomes underscore the dualistic nature of MAFs, connecting the strengths of individual activation functions while also merging their inherent limitations. The Tanh activation's bounded output range, beneficial for network stability, may concurrently limit representation of large value magnitudes. Conversely, ReLU's unbounded positive outputs, although conducive to maintaining healthy gradients, could result in unstable updates if not properly regulated. Interestingly, in second condition, the GCU's cosine-based periodicity offers a modulating effect on the activations, which can be advantageous but also adds a layer of complexity to the network's learning pattern. The task-dependent effectiveness of MAFs is evident, suggesting that specific environmental and data distribution considerations can significantly influence the utility of MAFs in DQN training.

Our findings reveal that the MAF comprising Tanh and ReLU delivers the most stable learning trajectory. Nonetheless, the GCU-involved MAF configurations maintain competitive performance, suggesting that the modulation benefits of GCU might be harnessed in scenarios where a controlled introduction of complexity can be advantageous.

d. Comparison of Gated Activation Function

Based on **Table 3, Fig 18, Fig 19** and **Fig 20,** in the comparative analysis of deep Q-networks utilizing adaptive gated activation functions (AGAFs), three distinct conditions were evaluated for their contribution to learning stability and efficiency.

In **Fig 18**, Condition 1, featuring a combination of Tanh and ReLU within the AGAF framework, achieved task completion by the 29th episode, showcasing a balanced learning pace. This configuration yielded moderate cumulative rewards and total reward of 160,955, with standard deviations suggesting moderate variability. In **Fig 19**, Condition 2, which employed a mix of GCU and SQU activation functions, slightly improved mean cumulative rewards for DR and AR over Condition 1 yet presented a lesser total reward. It resolved the task by the 30th episode, indicating a similar learning rate. Notably, it exhibited the most stable learning with the lowest standard deviations among three of the conditions pointing to a consistent learning process with less fluctuation across trials. In **Fig 20**, Condition 3 harnessed the rapid learning properties of Tanh and GCU, achieving task resolution by the 26th episode, the quickest among the three. While it boasted the highest mean cumulative rewards which is DR: 82,945 and AR: 79,656, it also experienced greater variability in learning, as evidenced by the higher standard deviations.

The divergent outcomes across conditions are reflective of the distinct activation function properties and their adaptive interactions. Condition 1, leveraging Tanh's normalizing effect with ReLU's gradient facilitation, yielded a balanced yet slightly restrained learning process. In contrast, Condition 2, through the fusion of GCU's periodicity and SQU's quadratic behavior, achieved a more uniform learning experience, suggesting that their combined attributes may introduce a stabilizing regularization despite increased computational complexity. Condition 3's rapid learning pace, attributed to the GCU's effective modulation of activation, underscores the potential for accelerated learning when AGAFs are optimally aligned with the task's demands. Concluding the observation, Condition 2, with its GCU and SQU activation functions, stands out due to its combination of GCU and SQU activation functions within the AGAF framework. It is notable for its balance of high rewards and consistent performance, marked by the lowest standard deviations among the conditions tested, highlighting its efficiency and stability in learning.

e. Comparison of Hirerachical Structure

There represent three conditions utilizing different activation functions within an Hierarchical Structure for Deep Q-Networks, notable differences emerge. In **Fig 21,** Condition 1, deploying PReLU and ELU, demonstrated effective learning, as evidenced by high mean cumulative rewards, but was slower to solve the task by the 43rd episode and showed moderate variability in performance. In **Fig 22,** Condition 2, incorporating Tanh and GCU, not only solved the task more swiftly at episode 32 but also achieved a higher mean cumulative DR, signifying a more efficient learning process with the lowest variability, suggesting a stable learning experience. In **Fig 23,** Condition 3, which amalgamated a wider array of activation functions, struck a middle ground in the speed of task resolution, completing the task by episode 39. However, it recorded slightly lower mean cumulative rewards and exhibited the greatest variability across all metrics, potentially indicating a compromise between learning efficiency and consistency. These observations

suggest that while the introduction of oscillating and adaptive activation functions can enhance learning speed, the complexity of interaction within a hierarchical activation structure introduces trade-offs between learning efficiency, stability, and overall performance variability.

The comparative performance analysis across the three conditions highlights the nuanced influence of activation function selection on Deep Q-Network (DQN) learning dynamics. Condition 2, with its Tanh and GCU activation functions, emerges as the most balanced, exhibiting swifter task resolution and greater reward efficiency, coupled with lower variability in learning outcomes. This suggests a synergistic effect where Tanh's tendency towards output saturation could be counteracted by the oscillatory nature of GCU, resulting in a more stable gradient flow. In contrast, Condition 1, despite effective learning rates, displays greater performance oscillations, possibly due to the mismatched dynamics between PReLU and ELU within its hierarchical structure. Condition 3 underscores the complexity inherent in integrating diverse activation functions, which does not unconditionally enhance performance, instead potentially increasing the unpredictability of learning trajectories and susceptibility to overfitting. This critical analysis underscores the necessity for a deliberate and context-aware selection of activation functions in DQN architectures to optimize learning efficiency and stability.

Condition 2, integrating Tanh and GCU, stands out for its accelerated learning, achieving task completion by episode 32, and showcasing superior reward accumulation efficiency. Notably, it exhibits the most consistent performance, with the lowest standard deviations across all reward metrics, indicative of a reliable learning process. The harmonious interplay between Tanh and GCU likely contributes to this stability, suggesting that their combined attributes effectively balance learning dynamics within the DQN framework.

Table 3: 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**

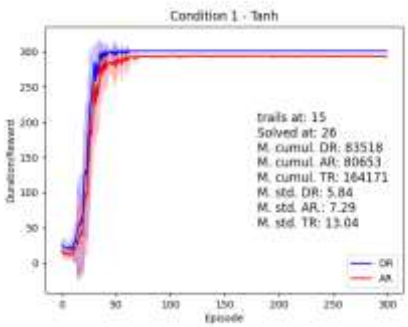

Fig 7: Graph of the *Tanh* Activation Function

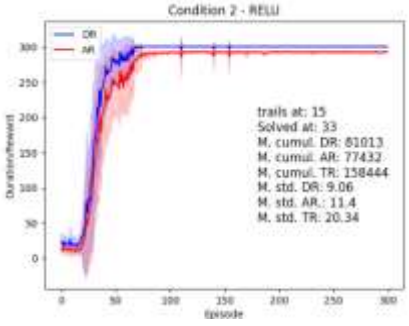

Fig 8: Graph of the *ReLU* Activation Function

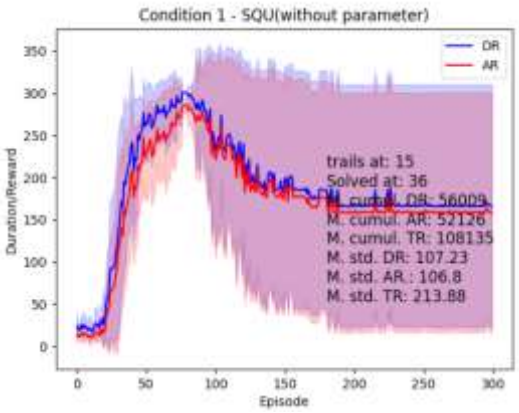

Fig 9: Graph of the *SQU(without parameter)* Activation Function

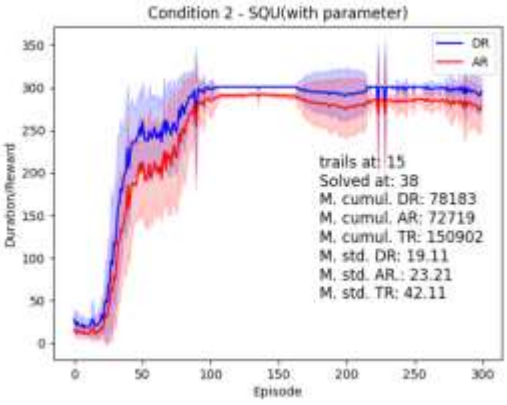

Fig 10: Graph of the *SQU(with parameter)* Activation Function

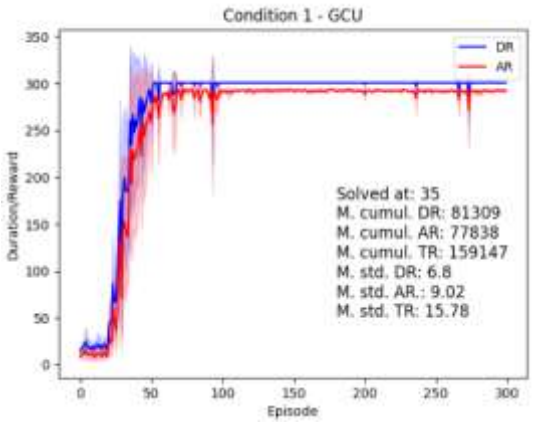

Fig 11: Graph of the *GCU* Activation Function

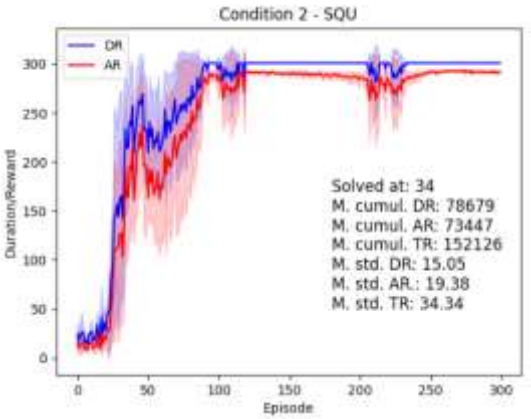

Fig 12: Graph of the *SQU* Activation Function

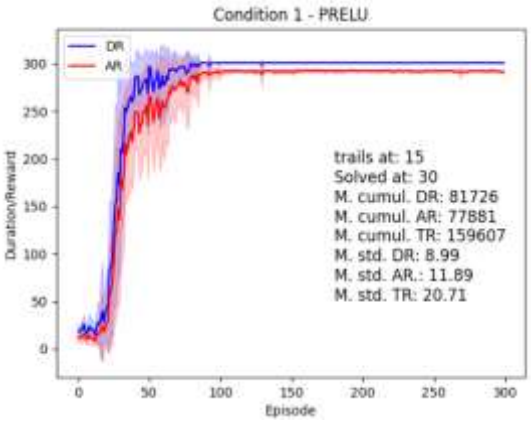

Fig 13: Graph of the *PReLU* Activation Function

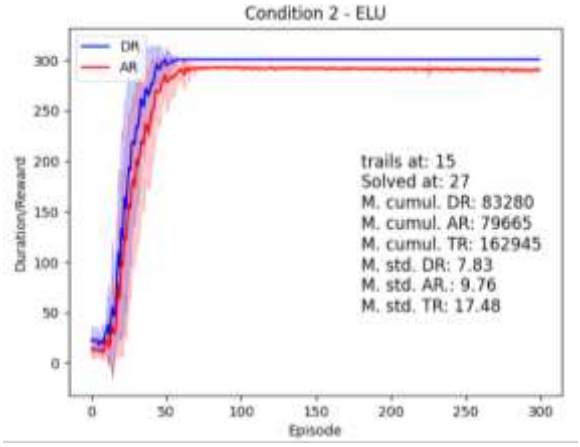

Fig 14: Graph of the *ELU* Activation Function

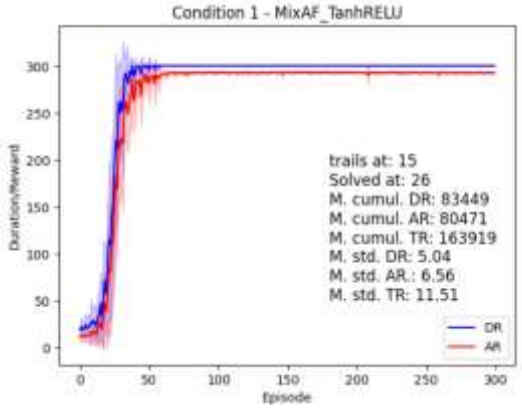

Fig 15: Graph of the *Mixed (Tanh & ReLU)* Activation Function

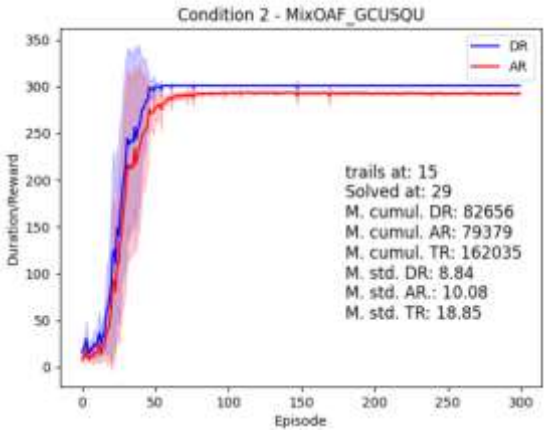

Fig 16: Graph of the *Mixed (GCU & SQU)* Oscillatory Activation Function

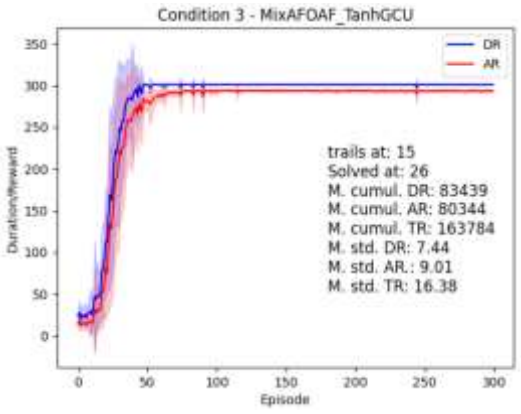

Fig 17: Graph of the *Mixed (Tanh & GCU)* Activation Function

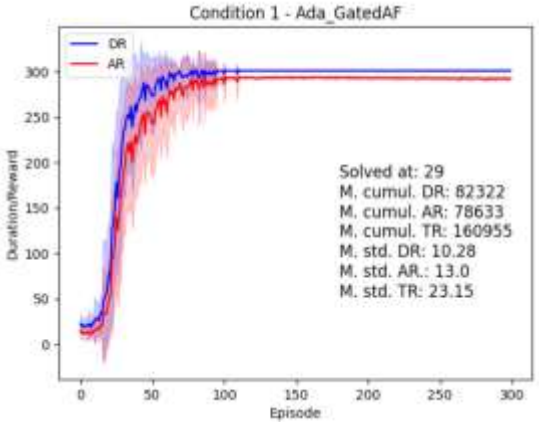

Fig 18: Graph of the *Gated (Tanh & ReLU)* Activation Function

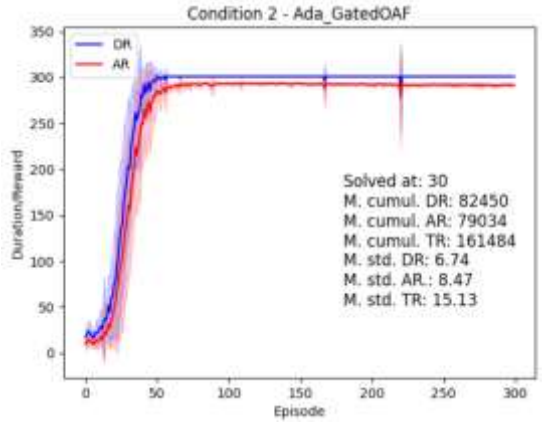

Fig 19: Graph of the *Mixed (GCU & SQU)* Activation Function

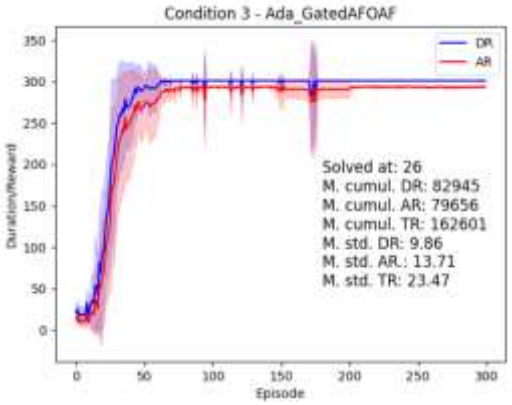

Fig 20: Graph of the *Gated (Tanh & GCU)* Activation Function

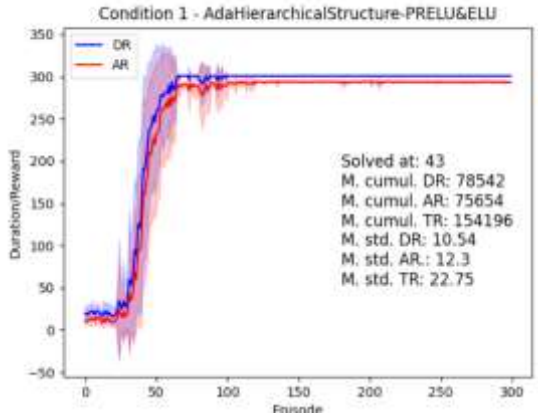

Fig 21: Graph of the *Adaptive Hierarchical Structure (PReLU & ELU)* Activation Function

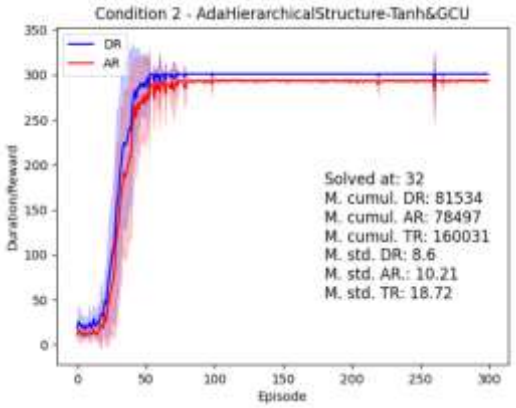

Fig 22: Graph of the *Adaptive Hierarchical Structure (Tanh & GCU)* Activation Function

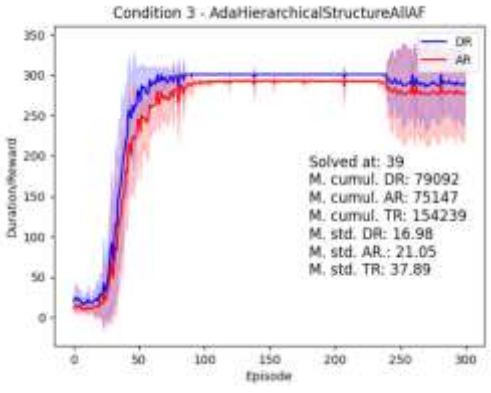

Fig 23: Graph of the *Adaptive Hierarchical Structure (All)* Activation Function

DISCUSSION

Based on our experiments, we have confirmed that the hypothesis that there traditional activation functions may be suboptimal compared to their oscillating counterparts in certain linear control tasks. Our experiments highlight that Tanh, a traditional activation function, is surpassed by oscillating functions such as GCU, as well as by the SQU activation function, with and without a control parameter.

However, an interesting finding the oscillating GCU function is combined with SQU, as seen in both the Mixed OAF and Gated OAF configurations, the results demonstrate a marked superiority over the conventional ReLU function. The second and third rank achieved by Gated OAF and Mixed OAF, respectively, as opposed to the fourteenth rank for ReLU, underscores the advantage of incorporating oscillatory patterns into the activation functions. This indicates that the introduction of oscillatory dynamics through GCU enhances DQN learning by facilitating more sophisticated gradient flows, which is conducive to improved exploration of the action space and overall learning stability.

Second hypothesis that has been answered is that employing mixed activation functions in DQN architectures do help in improving the performance of mixed activation functions, the synthesis of Tanh and ReLU manifests as the frontrunner, securing the top rank. This might stem from Tanh's proficient scaling and normalization harmonized with ReLU's capacity to preserve gradient integrity, a combination that accelerates and stabilizes the convergence of the learning process.

Furthermore, our study reveals that DQNs equipped with adaptive and oscillating activation functions, like those in Gated OAF setups, significantly improve learning. Gated OAF's second-place rank highlights the benefits of activation functions that adjust during training, offering a clear advantage over static ones. This flexibility, enhanced by gates and hierarchical structures, allows the network to choose the best activation for each learning stage.

The data shows oscillating functions, like GCU, prevent common issues such as vanishing gradients and promote thorough exploration and stable learning. Adaptive functions, seen in the success of Gated OAF, adapt to training data, which is particularly useful in complex environments where various activations are needed.

For hierarchical structures, 3 of the conditions do not show any outstanding performance compared to other approach. It can potentially offer more nuanced modelling of complex relationships in data; their increased complexity does not always lead to better performance. The effectiveness of such structures is highly dependent on the specific activation functions chosen too.

The standout performance of Mixed AF suggests that combining activation functions can leverage their strengths while minimizing weaknesses. This approach may be especially useful in diverse scenarios requiring different neural responses for the best decision-making. Our findings support moving beyond traditional activation functions to more complex and responsive neural architectures for improving DQN performance.

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/