2023

Explaining Deep Q-Learning Experience Replay with SHapley Additive exPlanations

Robert S. Sullivan
Technological University Dublin, Ireland

Explaining Deep Q-Learning Experience Replay with SHapley Additive exPlanations

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Robert S. Sullivan

A dissertation submitted in partial fulfillment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computer Science (Data Science)

June 16, 2023

Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Science), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Robert S. Sullivan
June 16, 2023

Abstract

Reinforcement Learning (RL) has shown promise in optimizing complex control and decision-making processes but Deep Reinforcement Learning (DRL) lacks interpretability, limiting its adoption in regulated sectors like manufacturing, finance, and healthcare. Difficulties arise from DRL's opaque decision-making, hindering efficiency and resource use, this issue is amplified with every advancement. While many seek to move from Experience Replay to A3C, the latter demands more resources. Despite efforts to improve Experience Replay selection strategies, there is a tendency to keep capacity high. This dissertation investigates training a Deep Convolutional Q-learning agent across 20 Atari games, in solving a control task, physics task, and simulating addition, while intentionally reducing Experience Replay capacity from 1×10^6 to 5×10^2. It was found that over 40% in the reduction of Experience Replay size is allowed for 18 of 23 simulations tested, offering a practical path to resource-efficient DRL. To illuminate agent decisions and align them with game mechanics, a novel method is employed: visualizing Experience Replay via Deep SHAP Explainer. This approach fosters comprehension and transparent, interpretable explanations, though any capacity reduction must be cautious to avoid overfitting. This study demonstrates the feasibility of reducing Experience Replay and advocates for transparent, interpretable decision explanations using the Deep SHAP Explainer to promote enhancing resource efficiency in Experience Replay.

Keywords: Deep Reinforcement Learning, Experience Replay, Experience Replay, SHapley Additive exPlanations, eXplainable Artificial Intelligence

Acknowledgements

I am immensely grateful for the incredible support and guidance I received throughout my journey of pursuing my passion for Data Science amidst the challenges of the Covid-19 pandemic and my full-time engineering work. Without the unwavering support of the following individuals, this achievement would not have been possible.

First and foremost, I express my deepest gratitude to the esteemed faculty members of TU Dublin's computer science department. Their exceptional mentorship and introduction to the captivating world of computer science have been invaluable to my growth.

I am indebted to Brendan Tierney for his invaluable guidance, which led me to connect with experienced faculty members and expand my knowledge.

Special thanks to Sarah Jane Delany and Robert Ross for their exceptional teachings in Machine Learning and Deep Learning methods. Their dedication provided me with a solid foundation in these areas.

Heartfelt appreciation goes to Luis Miralles, whose passion for Reinforcement Learning served as a constant inspiration, motivating me to embrace new challenges and explore the lesser-known side of Data Science.

I am deeply grateful to my supervisor Luca Longo for his expert guidance and patience, encouraging me to step out of my comfort zone and enhance my research skills, which led to publishing a late-breaking paper for XAI2030, presenting a poster at the conference in Lisbon and publishing a paper in the journal of Machine Learning and Knowledge Extraction.

Lastly, I extend my sincerest gratitude to my partner Emma, whose loving support, encouragement, and belief in my abilities have been my anchor throughout this journey. She gave me the strength and determination to persevere, for without her, none of this would have been possible.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, Reinforcement Learning (RL) has emerged as a powerful technique for optimizing complex control and decision-making processes (Y. Li, 2019). However, this potential has not been fully realized in regulated industries such as manufacturing (C. Li, Zheng, Yin, Wang, & Wang, 2023), finance (X. Wu et al., 2020), and healthcare(Yu, Liu, Nemati, & Yin, 2021) due to a critical challenge - the lack of explainability in Deep Reinforcement Learning (DRL). The absence of transparency in DRL models has resulted in difficulties in debugging and interpreting the decision-making process. These challenges, highlighted by (Vouros, 2022; Strubell, Ganesh, & McCallum, 2020) have led to the development of inefficient models that strain available resources. Moreover, as Deep Learning continues to advance, particularly with improvements to critical components like Experience Replay, the challenges related to debugging, interpretation, and inefficiency become even more pronounced (Thompson, Greenewald, Lee, & Manso, 2021). To address these pressing issues, an emerging field known as Explainable Reinforcement Learning (XRL) aims to provide solutions. One prominent tool in the XRL toolkit is SHapley Additive exPlanations (SHAP), which offers insights into the contributions of input features to a model (Heuillet, Couthouis, & Díaz-Rodríguez, 2021).

1.1 Background

Deep Reinforcement Learning (DRL) can optimize control and decision-making processes that are complex however, it lacks explainability, limiting its widespread use in regulated industries, where rising cost, safety, and ethical concerns exist. Experience Replay is a data sampling and storage technique, inspired by neurons during sleep (Hayes et al., 2021), to break data correlation and stabilize deep off-policy learning. Although Explainable Reinforcement Learning (XRL) is emerging, SHapley Additive exPlanations (SHAP) are a popular tool to explain model predictions. SHAP can help improve Deep Q-Learning (DQL) debugging and interpretation reducing inefficiencies that burden resources.

This dissertation leverages Deep SHAP, developed by (Shrikumar, Greenside, &

Kundaje, 2017; Lundberg & Lee, 2017), as an additional tool to visualize Experience Replay, enhancing the debugging and interpretability of Deep Q-Learning (DQL). The aim is to find in simulations of varying complexity, an optimal Experience Replay capacity size from 1×10^6 used by (Mnih et al., 2015), where the average cumulative game score remains high while Experience Replay capacity is minimized.

1.2 Research Project

Experience Replay is a data structure used to store a history or buffer of transitions experienced by the Agent, which are randomly sampled to train a neural network used to decide what actions to do next. Experience Replay size is important because it ensures that there is a diverse set of experiences available for training, and a small buffer size can help stabilize the training process. The network's weights may be initially unstable and sensitive to specific experiences but by having a small diverse set of experiences, the network is less likely to become biased towards any particular set of experiences, leading to more stable updates and faster convergence. However, If Experience Replay is too small, the Agent may repeatedly sample from a limited set of experiences, leading to overfitting of the neural network and poor generalization to new environments.

A smaller Experience Replay size will depend on the complexity of the environment, the size of the state and action spaces, and the computational resources available for training. Since Experience Replay is not well understood a default size of 1×10^6 used by (Mnih et al., 2015) or from 1×10^4 recommended by (S. Zhang & Sutton, 2017) is used for most applications. To improve the explainability of the Experience Replay, SHAP values can be used to show how each transition that is fed into the neural network affects its training. Furthermore, SHAP can help improve confidence in reducing the Experience Replay capacity size, which will reduce the burden of resources.

Therefore, this dissertation asks the following research question:

> "For simulations of varying complexity, what is the smallest Experience Replay size allowed in Deep Q-Learning and why?"

1.3 Research Objectives

The objective of this research is to investigate the effectiveness of reducing Experience Replay and observe rewards received by a Deep Q learning system in simulations of varying complexity to find the lowest size allowed and why, by:

1. Conducting a literature review for Deep Q-Learning, Experience Replay, Deep Reinforcement Learning, and Explainability AI techniques, that will provide a comprehensive understanding, identify gaps, and motivate a research question and new experiment.
2. Design experiments where Experience Replay is reduced and its effect on rewards received measured.
3. Implement a Deep Q-Learning, Deep Convolutional Q-Learning, and xAI Explainer model in simulations of varying complexity to run experiments and generate both reward scores and explanations.
4. Discuss findings in the context of XAI.

1.4 Research Methodologies

This research is devoted to understanding the minimal Experience Replay required for a Deep Q-Learning system, to learn an optimal set of actions or Policy in simulation that maximizes its total reward.

In order to achieve this goal, the experiment includes training a DQL learning system on a combination of different simulations, both vector and image-based, then graphing reward over the number of episodes completed to determine the minimum size of Experience Replay allowed. Graphing reward by episode will explain HOW the Agent performed and graphing SHAP values will explain WHY the Agent took an action in a state.

In detail, a DQL Agent will be created with Experience Replay capacity sizes set to 1×10^6 and reduced to 5×10^5, 1×10^5, 5×10^4, 1×10^4, 5×10^3, 1×10^3, and 5×10^2 transitions respectfully. It will be placed inside the Open-Ai's Gym simulation platform, and run for a set amount of time or episodes to create a test dataset. Reward by number of episodes will be recorded and saved as a CSV file. After all simulations have run each CSV file is merged together and an Analysis Of Variance (ANOVA) or its non-parametric equivalent Kruskal-Wallis test is subsequently executed, and a post-hoc test either Tukey or Dunn's test is run on the distribution of rewards. Deciding to use ANOVA over Kruskal-Wallis is based on a Shapiro-Wilk test confirming that reward data is normally distributed. Although the central limit theorem could be applied since a lot of samples are available this study remains cautious and lets the result of a Shapiro-Wilk test decide if Kruskal-Wallis is used instead. Ideally, low Experience Replay Size and high reward are desired for each simulation.

A SHAP Deep Explainer will be run with a sample of unseen data from Experience Replay and the trained DQL Agent model, to create SHAP values. These SHAP values will be graphed as heatmaps to explain why the Agent took actions in particular situations. Experience Replay will then be reduced to find the smallest allowed size where the reward remains high. The resulting images from the Deep SHAP Explainer are not being presented as support for the hypothesis, only as a visual illustration of the outcomes and trends observed during the training process.

1.5 Scope & Limitations

While some previous works extensively compared various aspects of DQL, such as (Mnih et al., 2015) comparing different algorithms, (S. Zhang & Sutton, 2017; Bruin, Kober, Tuyls, & Babuška, 2018) comparing different selection strategies, and (Fedus et al., 2020) who explored both increasing capacity and sampling ratio from 1×10^6 and 0.25 respectfully, the approach in this study is distinct. This study focuses solely on testing the effect of reducing Experience Replay capacity on the average cumulative reward received by a DQL learning system inside different Open-Ai Gym Environments: Classic Controls, Box2D Physics, Atari, and a custom-built Addiction simulator (Bellemare, Naddaf, Veness, & Bowling, 2013; Brockman et al., 2016).

Both this and visualizing experience replay have not been explored before. This paper also chose to use a Deep Convolutional Q-learning (DCQL) algorithm as (Fedus et al., 2020) did but to only test it in 20 Atari games, a classic controls problem, a physics problem, and a real-world simulation, due to hardware constraints. The aim is to find an optimal Experience Replay capacity size from 1×10^6 used by (Mnih et al., 2015), where the average cumulative game score is maximized, and Experience Replay capacity is minimized. The custom-built Addiction simulator is to emulate a regulated industry problem. This custom gym environment was built to resemble a tabular Addiction Lab Simulator by (Keramati, Durand, Girardeau, Gutkin, & Ahmed, 2017), who uses Homeostatic Reinforcement Learning to mimic experimental data on cocaine addiction in lab rats.

Since the states in all these simulations may contain vector information or images, actions will be limited to discrete values and depending on the simulation could be between 2 and up to 18 discrete values. State information will be processed by either an Artificial Neural Network containing 30 hidden units if vector or a Convolutional Neural Network containing 3 convolutions and 1 flattening layer will be used if images. The number of neurons and convolutions were chosen by performing an informal search on games like Breakout and Space Invaders, ideally to generalize the learning system so it can be used on many different kinds of tasks. Training a DQL system can take a long time and given the limited hardware resources available (see Section 7.3), the total number of episodes for each simulation will be constrained to 200. This will mean each simulation should only take an hour to train. Since a single image does not contain velocity or position change information the Convolutional Neural Network will require several instances of each image before it can train successfully, to that end 10 images will be batched every step, and training will only begin on that n-step instead of training on every step. Experience Replay samples will be held at 128 batches plus 10% to be collected and kept hidden from the DQL system. This is so they can be used later to generate Deep SHAP explanations. Other hyperparameters will also be held constant: $\gamma = 0.9$, $\alpha = 0.001$, $T = 100\%$. Experience Replay will be reduced in increments from 1×10^6, 5×10^5, 1×10^5, 5×10^4, 1×10^4, 5×10^3, 1×10^3, and 5×10^2 transitions respectfully.

1.6 Document Outline

The following chapters of this research are structured as follows:

Chapter 2 - Literature Review

In the next chapter literature relating to Deep Reinforcement Learning, Experience Replay, and Explainable AI is reviewed, highlighting recent progress and current gaps is summarized in a table chronologically by year. This chapter helps define the Research Question: "For simulations of varying complexity, what is the smallest Experience Replay size allowed in Deep Q-Learning and why?"

Chapter 3 - Design and Methodology

In this chapter, a research hypothesis is defined that helps answer the research question. Model architecture and evaluation methods are described. Finally, four experiments are detailed that will be used to test the alternative hypothesis: "There is a significant difference ($p < 0.05$) in reward scores when Experience Replay is reduced". Low Experience Replay Size and high reward are desired for each simulation.

Chapter 4 - Results, Evaluation, and Discussion

In this chapter results from experiments conducted in Open-Ai Gym environments: Classic Controls, Box2D Physics, Atari, and a custom Addiction simulator, are provided in a summary table, along with the results of a Kruskal-Wallis and Dunn's test. Finally, a discussion of the results in the context of xAi is provided with an answer to the research question.

Chapter 5 - Conclusion

In this chapter, a conclusion of all the steps involved in this research is given with the research contribution and ideas for future work, such as trying different methods and different simulations with higher rule density.

2 Literature Review

In this chapter, literature relating to Deep Reinforcement Learning, Experience Replay, and Explainable AI is reviewed to better understand gaps in current research. Gaps identified are used to develop a research question focused on improving Deep Reinforcement Learning resource efficiency to gain wider societal acceptance in regulated industries. Information about Deep Reinforcement Learning, its internal components, and Explainable Ai is derived from the following sources: (Sutton & Barto, 2018; de Ponteves, 2019; Bilgin, 2020; Ras, Xie, van Gerven, & Doran, 2022; Bhattacharya, 2022; Mishra, 2023; Mnih et al., 2015, 2016). The latest research is reviewed and gaps are presented chronologically from classical methods to state of art. A summary of this is provided in a table at the end, with a research question.

2.1 Classic Reinforcement Learning

(Bilgin, 2020) tells us that in supervised learning, a machine learning model has a supervisor giving a ground truth to each data point. The model learns to minimize the difference between its prediction and the ground truth. The dataset needs to be labeled. In unsupervised learning we let the model learn about the distribution of and patterns within the data without a ground truth. Now in Reinforcement Learning (RL), the model or learning system is called an Agent. Without a labeled dataset it learns to complete tasks through trial and error by receiving a reward indicating how well it is performing at the task. A reward is received when the Agent performs an action inside an environment where the Agent resides (Bilgin, 2020). As seen in Figure 1 the environment provides information to the Agent about the world and provides rewards to it when it performs actions.

2.1.1 Markov Decision Process structures Problems

Other than simple Multi-Arm Bandit (MAB) and Contextual Bandit problems, solving a problem or controlling a process can be structured as a tuple called a Markov Decision Making Process (MDP) (Bilgin, 2020; de Ponteves, 2019). This makes it easier to deal with high-dimensional contexts and complex sequential decision-making problems. Firstly, MDP contains a set of different states s such as a vector of encoded values or an image that contains all the information needed to describe what happened at time t, this can be also known as the state space. Secondly, it holds a set of actions a that can be chosen at time t (e.g. four actions: forward, back, left, right). Thirdly, it contains a Transition Rule or probability distribution, describing the probability of the next state at time $t+1$ given the current state and action selected. Finally, it holds a Reward r that returns a score for selecting an action in a given state. MDP can be applied to many real-world applications according to (White, 1993), such as insurance, sales, inventory management, and patient admissions. It assumes that the stochastic process to be modeled has the Markov property, meaning the conditional probability distribution of being in a future state does not depend on any previous states or actions, except the current ones. The following formally defines the Transition Rule and Reward Function:

$$\text{T:}(\text{a}_t \in A, s_t \in S, s_{(t+1)} \in S) \rightarrow P(s_{(t+1)}|s_t, a_t) \tag{1}$$

$$\text{R:}(\text{a}_t \in A, s_t \in S) \rightarrow r_t \in R \tag{2}$$

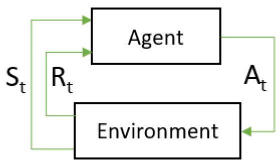

Figure 1: MDP: A learning system called an Agent inside an MDP environment, at time t observes a state s_t, selects an action a_t and receives from the environment a reward r_t then moves into the next state s_{t+1}.

2.1.2 Policy selects Actions

To know what action the Agent should take, it follows a Policy π.

$$\pi : s_t \in S \rightarrow a_t \in A \tag{3}$$

It can be thought of as a set of instructions to follow, a strategy, or just a function that maps states to actions. It can be a set of random actions, selected by a subject matter expert or dynamically created by a learning system. The best Policy π^* out of a collection of policies Π is one that maximizes the cumulative reward (de Ponteves, 2019).

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \sum_{t \geq 0} R(\pi(s_t), s_t) \tag{4}$$

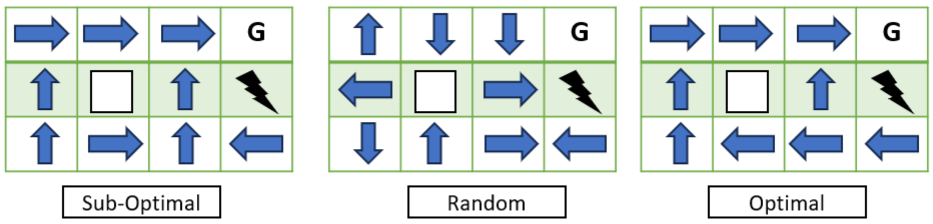

Figure 2: Policies: An example of a sub-optimal, random, and optimal Policy for an Agent navigating a maze. G is the goal where the Agent will receive a reward and the bolt of lighting will give the Agent a negative reward.

2.1.3 Accumulating Reward

Reward in Reinforcement Learning is predetermined when defining the MDP. Examples of MDP systems could be pulling a lever on a slot machine to reward a gambler with coins, receiving revenue while limiting expenses to reward a business with profit, navigating from an airport to downtown to reward a traveler with their desired destination, and successfully playing a computer game to reward a player with a high score. Reward received by selecting an action in a given state at time t, may not always be the highest reward possible in every state-action combination, therefore all possible rewards for all actions in all states, must be taken into account. A certain combination of states will also lead to better future rewards than others. In Figure 2 the Agent will receive a score of 1 if it reaches the goal and -1 if it touches the lightning bolt. All other states will have a discounted reward based on getting towards the future reward of 1. Gamma γ is used to signify the discount factor and is always set to less than 1 (e.g. 0.95 or 0.99). When γ is close to 1 the Agent will optimize for future rewards but anything greater than 1 will cause the discount reward to reach infinity. When γ is close to 0 the Agent will optimize its current reward (de Ponteves, 2019). The reward for each state can be defined as follows:

$$\mathrm{R}_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} ... + \gamma^{n-t} r_n \tag{5}$$

The Agent explores the world to receive more information to make better decisions or to exploit good decisions it already knows. Several algorithms exist to develop exploration strategies such as A/B/n testing, Epsilon-greedy, Upper Confidence bounds, and Thomson Sampling (Bilgin, 2020).

2.1.4 Dynamic Programming & Monte Carlo (MC) (Model Based)

Dynamic Programming (DP) is an optimization technique that proposes optimal solutions to MDPs. They are more efficient to direct search and linear programming methods but suffer from the curse of dimensionality as most real-world problems are too complex to solve this way. Policy Iteration and Value iteration algorithms must iterate over an entire state space multiple times before finding an optimal Policy. Asynchronous Dynamic Programming is an alternative approach that only focuses on states that are more likely encountered. However, not all parts of the state space are important. These methods also expect that we fully know the transition probabilities which in most cases we simply won't know. Monte Carlo methods are a powerful way to learn an optimal Policy only from the experience we collect by interacting with the environment without knowing everything about it (Bilgin, 2020).

2.1.5 Model-Based, Model-Free, On-Policy & Off-Policy

Algorithms discussed later in Section 2.1.6, are model-free, which means unlike DP and MC discussed in Section 2.1.4, they don't assume any knowledge about the transition dynamics of the environment and only learn from sampled experiences. However, model-based methods do exist, that can lead to better sample efficiency in some problems which is appealing, like in robotics research where experience collection can be costly (Bilgin, 2020) but are more computationally expensive. Model-based methods come in all shapes and sizes such as derivative-based methods, derivative-free methods, open-loop, closed-loop, and model predictive control (MPC). Examples of such algorithms are the Random Shooting procedure, Cross-entropy method, covariance matrix adaption evolution strategy (CMA-ES), and Monte Carlo Tree Search. With model-based methods, migration of model uncertainty is important especially when neural networks are used with limited data (Bilgin, 2020). This can be handled with Bayesian Neural Networks (Ghavamzadeh, Mannor, Pineau, & Tamar, 2015) and Ensemble models like bootstrapping. There was an approach in the early 90s to unifying model-based and model-free methods with the Dyna class of methods. Dyna is widely used in Automated Guided Vehicles (AGV) path planning but Dyna cannot directly handle continuous actions in RL (G. Wu et al., 2022).

Exploration is necessary to find the optimal Policy during training but once found there is no real need to do any more exploration. Instead, the Agent should select the best actions according to the Policy. The Policy we follow to explore the world is called the Behavior Policy and the optimal one we want to find and then follow is called the Target Policy. This is in essence what On-Policy and Off-Policy are. Off-policy methods estimate the state and the action values for a Policy that is different from the Behavior Policy. Whereas On-Policy methods estimate these values only for the Behavior Policy (Bilgin, 2020). The sub-optimal Policy in Figure 2 would be an example of a Behaviour Policy that implements Epsilon Greedy as an exploration strategy. Off-policy methods are more efficient as we can reuse past experience to solve current problems. They can also use data from non-RL controllers such as classic PID controllers. On-policy methods are good when the cost of exploration is high for example an Agent in a real-world self-driving car, trying to explore whether driving on a sidewalk full of pedestrians is good or not. They are also easier to work with when action spaces are continuous.

2.1.6 Temporal Difference & Q-Learning (Model Free)

The quality of selecting an action given a state is called the Q-value or $Q(a, s)$. According to (de Ponteves, 2019), at $t = 0$ all Q-values for every state-action pair are set to zero, at every step in time t we enter a state, select an action, receive a reward then move to the next state.

Temporal Difference means the difference between the reward from action selected plus a percentage of the best known future action Q-value $r_t + \gamma \max_a(Q(a, s_{t+1}))$, less the current state Q-value from action played $Q(a_t, s_t)$. Temporal Difference is similar to an intrinsic reward. The Agent likes when it's high and hates when it's low (de Ponteves, 2019). It can be formally defined as follows:

$$\mathrm{TD}_t(a_t, s_t = r_t + \gamma \max_a(Q(a, s_{t+1})) - Q(a_t, s_t) \tag{6}$$

Q-learning is then an extension of Temporal Difference in that we measure the accumulation of high or low Temporal Differences associated with pairs of states and actions. State-action pairs that have a high Temporal Difference are reinforced while those that are low are weakened. The Agent then wants to learn Q-values that will give it the maximum Temporal Difference using the Bellman Equation (Bellman, 1957) below:

$$\mathrm{Q}_t(a_t, s_t) = Q_{t-1}(a_t, s_t) + \alpha TD_t(a_t, s_t) \tag{7}$$

2.1.7 Epsilon Greedy vs Softmax Policy

An Agent's decision to choose an action or not in a given state depends on how it interprets Q-values $Q(a_t, s_t)$. As mentioned in Section 2.1.3, many methods exist to explore which actions are the best. Epsilon Greedy and Softmax are two popular methods that have subtle differences (Tokic & Palm, 2011). Epsilon Greedy is a simple exploration method that exploits actions with the highest estimated value most of the time but will randomly select other actions based on a small epsilon. Softmax exploration is more probabilistic where it assigns probabilities based on their Q-values. This exploration based on values allows the Agent to make a more informed decision. Both allow for all actions to be explored but Softmax provides smoother exploration that considers the probability of all actions based on a Temperature T parameter. Epsilon Greedy in contrast is more binary, either exploring or exploiting. Softmax prefers actions with higher estimated Q-values making it more focused on exploration. It can utilize past experience as Epsilon Greedy balances action exploration and exploitation randomly regardless of their estimated value. The higher the Softmax Temperature the more exploration is carried out and a low Temperature makes action selection more deterministic (de Ponteves, 2019). A small epsilon value favors exploitation while a large epsilon increases exploration. Softmax is preferred over epsilon-greedy in Deep Reinforcement Learning when a more nuanced and continuous exploration profile is needed. The formula for the Softmax method is as follows:

$$\mathrm{W}_s : a \in A \to \frac{\exp\left(Q(s,a)\right)^T}{\sum_{a'} \exp\left(Q(s,a')\right)^T} ; T \geq 0 \tag{8}$$

2.2 Deep Q-Learning

(Mnih et al., 2015; Sutton & Barto, 2018) introduced DQL to optimize the control process in complex environments. The Agent uses equation (7) through trial and error to learn the quality of taking an action in a given MDP state to find an optimal policy that maximises its total reward.

2.2.1 Loss Function for Backpropagation

A neural network is used to approximate Q-values. This allows for the Agent to find an optimal Policy in an extremely large and complex state space. As seen in Figure 3, information about the state is passed into the input layer of the Neural network, and the output layer is comprised of predicted Q-values for each action available. The chosen action is then determined by Softmax. The target is based on the reward from the action played plus a percentage of the Q-value of the best known future action $r_t + \gamma \max_a(Q(a, s_{t+1}))$, less the current state Q-value from action played $Q(a_t, s_t)$ as discussed in 2.1.6. Similar to traditional Deep Learning the loss error is the backpropagation of the squared temporal difference (de Ponteves, 2019) using the following formula:

$$\text{Loss} = \frac{1}{2}(r_t + \gamma \max_a(Q(a, s_{t+1})) - Q(a_t, s_t))^2 = \frac{1}{2} TD_t(a_t, s_t)^2 \tag{9}$$

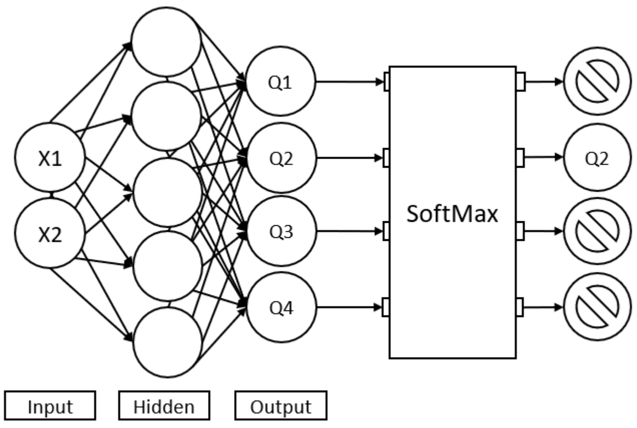

Figure 3: DQL: Deep Q Learning with Softmax output Policy.

2.2.2 Experience Replay

Experience Replay is an internal sampling technique for Deep Reinforcement Learning, inspired by neurons during sleep (Hayes et al., 2021), to break data correlation and stabilise deep off-policy learning. Approximating Q-values with a neural network in large and complex states destabilizes learning, so (Mnih et al., 2015) used Experience Replay (Lin, 1992) to sample and store data from the environment in an array called the Experience Replay memory buffer M for the neural network approximator to reuse later. However, drawbacks of Experience Replay included correlated samples, limited capacity causing an Agent to forget information, outdated samples from non-stationary environments, and overfitting from samples memorized. Several variations of Experience Replay were developed to solve these, such as Prioritized Experience Replay (PER) (Schaul, Quan, Antonoglou, & Silver, 2016) and Attention-based Experience Replay (Ramicic & Bonarini, 2017)

Understanding Experience replay is crucial for efficiency. A general trend for deep learning has seen performance increase at the cost of an enormous resource expansion. For example, the deep image classification model AlexNet trained for 5 to 6 days on two GPUs in 2012 was surpassed by NASNet-A in 2018. NASNet-A cut the error rate in half but at the expense of 1,000 times as much computing required (Thompson et al., 2021). Deepmind's Agent57 (Kapturowski et al., 2023) which beat human champions in Atari contained 80 billion frames of experience to achieve optimal performance. Consequently many consider Experience Replay flawed with most wanting it replaced. Asynchronous Actor-Critic (A3C) by (Mnih et al., 2016) is a popular alternative. It trains multiple Agents in parallel, to explore the environment, and update a shared network, requiring more resources but converging faster. Experience replay, although slower, is more memory efficient only requiring stored transitions and not multiple copies of the network.

(S. Zhang & Sutton, 2017) highlighted that the size of Experience Replay M is a neglected hyperparameter and if large hurts performance, but (Bruin et al., 2018) stated to keep it high using 90% of total environment transition steps as a rule of thumb. (Fedus et al., 2020) stated with the lack of understanding of Experience Replay most default to a value as high as Mnih's 1 million transition steps for size. Experiments in Atari showed increasing Experience Replay from 1 million to 10 million transitions while also decreasing the age of the oldest Policy did improve performance. However, any increase in the size of Experience Replay further burdens resources.

2.3 Deep Convolutional Q-Learning

2.3.1 Working with Pixels

(Mnih et al., 2015) were able to get a Q-Learning Agent to play Atari games from pixels. This was due in part to extending DQL with Convolutional layers to convolve over sub-samples of screen pixels learning feature detectors from input data. According to (Kapturowski et al., 2023), since this key milestone, progress has exploded with DQN improvements such as Double DQN, Rainbow, Prioritised Replay, and Distributed Reinforcement Learning. In 2019 R2D2 introduced Long Term Short Term Memory (LSTM), in 2019 Episodic Memory was introduced with memory networks and transformers, while exploration improvements introduced Curiosity and Intrinsic Motivation. Now with Agent57 and its eventual successor MEME (Efficient Memory-based Exploration), Meta-Controllers based on Bandits (discussed in Section 2.1.1) are allowing for Human-level Atari 200 times faster than Agent57. Based on this research, progress seems to be heading towards meta-controllers which may result in more handcrafted model-based approaches as discussed in Section 2.1.5, away from generalized Agents. Focusing on the original Deep Q-Learning and Deep Convolutional Q-Learning systems, to improve their sub-component efficiency and explainability may help alter this course.

2.3.2 The Convolutional Layer & Feature Maps

Samples added to Experience Replay contain Convolutional layers of pixels from the state space.

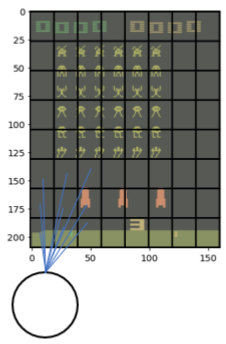

Figure 4: CNN: A small local feature together with other local features creates a convolutional layer. Each of these features in turn results in individual feature maps.

Each convolutional layer has a width, height, and depth. Depth is the number of filters or neurons. Each neuron allows another feature to be learned. All neurons at the same depth will share the same weights (de Ponteves, 2019). As per Figure 5 filters can overlap each other to sample from each other's pixels. This is known as stride.

Feature maps containing features about the state space are then pooled and flattened into a single layer. The 1-dimensional flattened layer that represents features from the state space is passed to a neural network that outputs Q-Values, which Softmax uses to select an action.

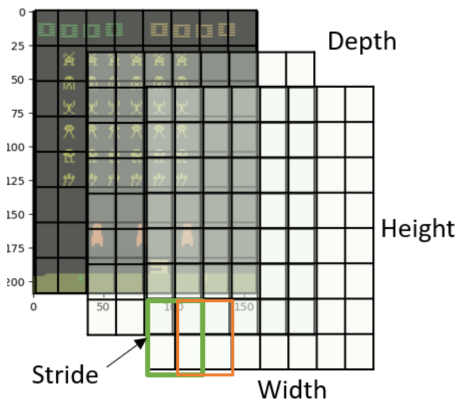

Figure 5: CNN: Parameters of a convolutional layer.

2.3.3 N-Step Q-Learning

Eligibility Trace according to (Lanham, 2020) is a technique used to assign credit to past actions in a sequence of states and actions, to update the weights of a neural network model. It helps address the problem of delayed rewards and enables more effective learning in environments with long-time delays between actions and their consequences. This is useful for learning to play complex games from pixels. During the learning process, the Eligibility Trace is updated at each time step by decaying its value and adding the gradients of the weights with respect to the loss function. When an update is performed, the Eligibility Trace determines how much credit is assigned to each weight based on its previous contributions to the current outcome. By accumulating the Eligibility Traces over time, the algorithm can assign credit to actions that lead to future rewards, even if those rewards are delayed.

2.4 Using Simulation to Train & Evaluate Models

(Bilgin, 2020) explains that Reinforcement Learning requires more data than regular deep learning. Complex models can take many months to train with millions of iterations. Since it is impractical to collect data like this, RL relies heavily on simulated environments. This causes a lot of issues when working with RL. Businesses don't have access to simulated models of their processes. Simulations that do exist are often too simplistic causing the RL model to overfit the simulation and fail when scaled to the real world. The solution here is calibrating the simulation to reflect reality as training and testing of machine learning models must follow the same distribution, but calibrating a simulation is costly and time-consuming, this is known as the Sim2Real gap. Increasing the realism in simulation increases computational consumption making it difficult to quickly experiment and deploy RL. Many simulations are not generic enough. A lot of commercial software is hard to integrate with RL or may not be flexible enough to work with models. (Bilgin, 2020) states that simulations should be fast, accurate, and scalable to many sessions.

Simulated environments, such as the *Arcade Learning Environment* and *Open-ai gym* (Bellemare et al., 2013; Brockman et al., 2016), mimic real-world problems and generate valuable training data in a secure manner. Evaluation of performance is comparing the Agent in these simulations to a handcrafted, human expert, or random Policy. In *Open-Ai Gym* there are four types of environments of varying complexity: *Classic controls*, *Box2d Physics*, *Atari*, and *Custom created environments.*

- **Classic Controls Environments**: Acrobot, CartPole, Mountain Car, Continuous Mountain Car, and Pendulum. All stochastic from their initial state, within a given range and noise is added when actions are taken. These environments are considered easy to solve.
- **Box2d Physics Engine**: Toy games based around physics control.
- **Atari 2600**: A set of Atari 2600 environments simulated through Stella and the Arcade Learning Environment.
- **Custom Gym Environment**: Researchers can create their own simulations or use many third-party environments which include multi-Agent RL (PettingZoo), offline-RL (Minari), gridworlds (Minigrid), robotics (Gymnasium-Robotics), multi-objective RL (MO-Gymnasium) many-Agent RL (MAgent2) and 3D navigation (Miniworld).

2.5 Explainable Reinforcement Learning

Explainable Artificial Intelligence (XAI) aims to enhance the transparency and interpretability of AI systems, making their decision-making processes easily understandable. For Deep Reinforcement Learning models like Deep Q-Learning to become more prevalent, there is a need to improve the debugging and interpretation of their decision-making process. XAI is crucial for building trust, accountability, complying with regulations, and ethical considerations (Vilone & Longo, 2021b). Various methods have been developed to achieve explainability in AI systems (Longo, Goebel, Lécué, Kieseberg, & Holzinger, 2020; Vilone & Longo, 2021b, 2021a). A common approach involves generating post-hoc explanations, where models' decisions are explained after they have made predictions. Techniques like feature importance visualization, saliency maps, and SHAP (Shapley Additive exPlanations) fall into this category (Vilone & Longo, 2021b). Researchers are exploring hybrid models that combine the power of complex models with interpretability layers. Methods such as TreeExplainer, a modified version of SHAP for tree-based ML models, and combining SHAP with the Lorenz Zonoids decomposition, to determine relevant features by generating a receiver operating characteristic (ROC) curve. This paper considers Deep Explainer (Deep SHAP), which builds on the DeepLIFT (Deep Learning Important FeaTures) algorithm (Shrikumar et al., 2017), a method to deconstruct the output prediction of a neural network based on a specific input by backpropagating the contributions of all neurons in the network to every feature of the input and SHapley Additive exPlanations (SHAP) (Lundberg & Lee, 2017). SHAP recreates a model's outcome by quantifying the marginal contribution of features to that model for a single instance. For example, a salary predicting model could start with a base of 50 euro per hour and three feature inputs (age, gender, and experience) the model then outputs 40 euro per hour, as each feature is removed to determine its marginal contribution and it is found that age negatively impacts the model by -10 euro per hour when compared to the base. The hybrid model Deep SHAP approximates SHAP values by creating connections with DeepLIFT and is trained on the distribution of base samples instead of a single reference value. In a Deep SHAP plot, the input image is presented in transparent grayscale while the output is presented as different colored images. Red pixels indicate an increase in the model's output while blue pixels decrease the output. The sum of the SHAP values equals the difference between the expected model output, which is averaged over the background images, and the current model output. In Deep Convolutional Q-Learning, this would be an image from a game and predicted Q-values for each action, n-steps into the future. A SHAP heat map could highlight why an action is getting a higher q-value, for example in SpaceInvaders, this could be due to the Agent expecting a mothership to appear causing the agent to favor shooting in an empty space to hit the ship and receive 300 points.

The minimum Experience Replay size allowed in DQL is unknown, but using Deep SHAP explainability while reducing it can provide insight into the behavior of the algorithm, training dynamics, or other relevant aspects, highlighting any unexpected observations or patterns to spark further interest or investigation. (Longo et al., 2020; Vilone & Longo, 2021b, 2021a; Lundberg & Lee, 2017). Many custom explainers for DQL exist (Keramati et al., 2017; Miralles-Pechuán, Jiménez, Ponce, & Martinez-Villaseñor, 2020; K. Zhang, Zhang, Xu, Gao, & Gao, 2022; Thirupathi, Alhanai, & Ghassemi, 2022) to try to understand simulation events or the Agent within them, but not for Experience Replay samples that the model uses as training data to predict Q-values. Within Explainable Reinforcement Learning (XRL) (Heuillet et al., 2021; Ras et al., 2022; Vouros, 2022), SHAP (Lundberg & Lee, 2017) is a popular choice for black-box explanation (Kumar, Vishal, & Ravi, 2022; Thirupathi et al., 2022; K. Zhang et al., 2022). RL-SHAP diagrams exist to explain grid world environment features and their effect on action selection, but it has not been used for image-based simulations nor on samples stored in Experience Replay itself. Similarly, Experience Replay can be partitioned into clusters and given explainable labels based on rule density to select environment features (Sovrano, Raymond, & Prorok, 2021), but this is an experience selection strategy and not a post-hoc explanation for debugging a DCQL model. This dissertation proposes the use of Deep SHAP Explanations to generate images of Experience Replay samples allowing patterns and trends to be visualized, to assist in debugging the DQL model as Experience Replay capacity is reduced. In detail, the following section describes a primary research experiment devoted to searching for a minimum Experience Replay in Deep Q-Learning and using Deep SHAP to visualize Experience Replay during the training process.

2.6 Summary of Literature Review

Reviewed papers on progress made with Deep Q-Learning, Experience Replay, and Explainable Ai are summarised in Table 1 below. As can be seen from the table, introducing Deep Learning to Reinforcement Learning allowed RL to handle complex state spaces such as pixels, the field then exploded with more improved models able to outperform humans at various games such as those in Atari. Since then explainability of these systems has emerged as a focal point, especially to improve efficiency.

2.6.1 Summary Table of Reviewed Papers

Table 1: Summary of Reviewed Papers

Year	**Author**	**Contributions**	**Gaps**
1957	(Bellman, 1957)	Optimally Equation	Not applied
1992	(Lin, 1992)	Experience Replay	Model complexity
1993	(White, 1993)	MDP applications	N/A
2013	(Bellemare et al., 2013)	Atari env	Performance metrics
2015	**(Mnih et al., 2015)**	**Control from pixels**	**Planning strategies**
2015	(Schaul et al., 2016)	PER	uniform Sampling bias
2016	(Mnih et al., 2016)	A3C	Experience replay with A3C
2016	(Brockman et al., 2016)	Gym	Transfer learning
2017	**(S. Zhang & Sutton, 2017)**	**CER**	**CER is a workaround**
2017	(Ramicic & Bonarini, 2017)	ABERE	State class criteria
2017	(Keramati et al., 2017)	Addiction Sim	Prior addiction
2017	(Lundberg & Lee, 2017)	SHAP	XRL based SHAP
2018	**(Bruin et al., 2018)**	**Buffer size ROT**	**Retention strategy**
2018	(Sutton & Barto, 2018)	RL	N/A
2019	(de Ponteves, 2019)	RL Info	N/A
2020	**(Fedus et al., 2020)**	**Replay size study**	**Replay interactions**
2020	(Miralles-Pechuán et al., 2020)	Explainable Planning	Country specific
2020	(Bilgin, 2020)	RL Info	N/A
2021	(Hayes et al., 2021)	Experience replay biological view	N/A
2021	**(Sovrano et al., 2021)**	**Clustering explainer for experience replay**	**WHY performance evaluation**
2021	(Heuillet et al., 2021)	Explainability Info	N/A
2021	**(Liessner, Dohmen, & Wiering, 2021)**	**Shap XRL**	**Not problem agnostic**
2021	(Vilone & Longo, 2021b, 2021a)	Explainability Info	N/A
2022	(G. Wu et al., 2022)	Dyna Info	N/A
2022	(Ras et al., 2022)	Explainability Info	N/A
2022	(K. Zhang et al., 2022)	Explainable power ctrl	Not model agnostic
2022	(Vouros, 2022)	XRL survey	Field still emerging
2022	(Thirupathi et al., 2022)	Explainable startups	Not model agnostic
2022	(Bhattacharya, 2022)	Explainable Info	N/A
2022	**(Kapturowski et al., 2023)**	**Agent57**	**MEME**
2023	(Mishra, 2023)	Explainable Info	N/A

2.6.2 Gaps in Literature

The following gaps were identified in the literature:

- (Mnih et al., 2015) sampled from 50 million pixels (38 days of experience) to play Atari games. Set capacity size to 1,00,000.

- (S. Zhang & Sutton, 2017) stated Experience replay is not well understood and most default to 1,000,000. Studied experience capacity on Grid World, lunar lander, and Atari Pong. Did not use images directly with CNNs. Suggested at 10^7 all transitions are assumed, 10^6 is better than 10^5 but worse than 10^2, 10^3 is best and anything less decreases learning quality but did not directly test images with CNNs only raw vector from sim.

- (Bruin et al., 2018) investigated how the utility of different experiences is influenced by the control problem and proposed guidelines on how to use prior knowledge about the problem to choose an experience selection strategy. Recommended as a rule of thumb to keep Experience Replay high using 90% of total environment transition steps. (Mnih et al, 2015) would have set to 45 million transitions following this rule-of-thumb.

- (Fedus et al., 2020) showed in the Atari Learning Environment that increasing Experience Replay from 10^6 to 10^7 transitions while also decreasing the age of the oldest Policy did improve performance but did not reduce to find a minimum size.

- (Sovrano et al., 2021; Liessner et al., 2021); RL-SHAP diagrams exist to explain grid world environment features and their effect on action selection, but it has not been used for image-based simulations nor on samples stored in Experience Replay itself. Similarly, Experience Replay can be partitioned into clusters and given explainable labels based on rule density to select environment features but this is an experience selection strategy and not a post-hoc explanation for debugging a DCQL model.

- (Kapturowski et al., 2023) used 80 million frames of experience for agent world champion level. MEME (Efficient Memory-based Exploration) to try to improve memory efficiency.

Deep Learning revolutionized the field of classical Reinforcement Learning. Managing state complexity took priority over resource efficiency in the early days (Mnih et al., 2015, 2016) and as models developed and became distributed, resource efficiency had to become the focal point. Research is being directed towards handcrafted meta-controller systems like Agent57 and its eventual successor MEME (Kapturowski et al., 2023), this seems to be going in the wrong direction to generalized agents, a consistent aim of much of their prior research.

Although Experience Replay is not well understood and many believe it to be flawed (S. Zhang & Sutton, 2017) with A3C by (Mnih et al., 2016) being a popular alternative, A3C requires more resources, so researchers revisited Experience Replay. They increased its size (Bruin et al., 2018), increased the amount of experience being sampled (Fedus et al., 2020), and developed selection criteria for experience (Schaul et al., 2016; Ramicic & Bonarini, 2017; S. Zhang & Sutton, 2017; Sovrano et al., 2021). These efforts have been shown to improve model performance. However, increasing the Experience Replay capacity size will burden resources to process the additional transitions. The minimum Experience Replay size allowed in DQL is unknown, but using Deep SHAP explainability while reducing it can provide insight into the behavior of the algorithm, training dynamics, or other relevant aspects, highlighting any unexpected observations or patterns to spark further interest or investigation.

Explainability can help (Vilone & Longo, 2021b, 2021a) and custom Explainers exist for RL environments, to aid in the explainability of sampling and to understand the black box nature of Artificial Neural Networks, but existing research does not tackle the optimization of Deep Q-Learning Experience Replay by using SHAP. This could improve the explainability of Experience Replay by reducing its capacity in several simulations of varying complexity. This gap has led to the following research question:

> "For simulations of varying complexity, what is the smallest Experience Replay buffer size allowed in Deep Q-Learning and why?"

In order to tackle the research question, empirical work is set out, as described in the next chapter.

3 Design & Methodology

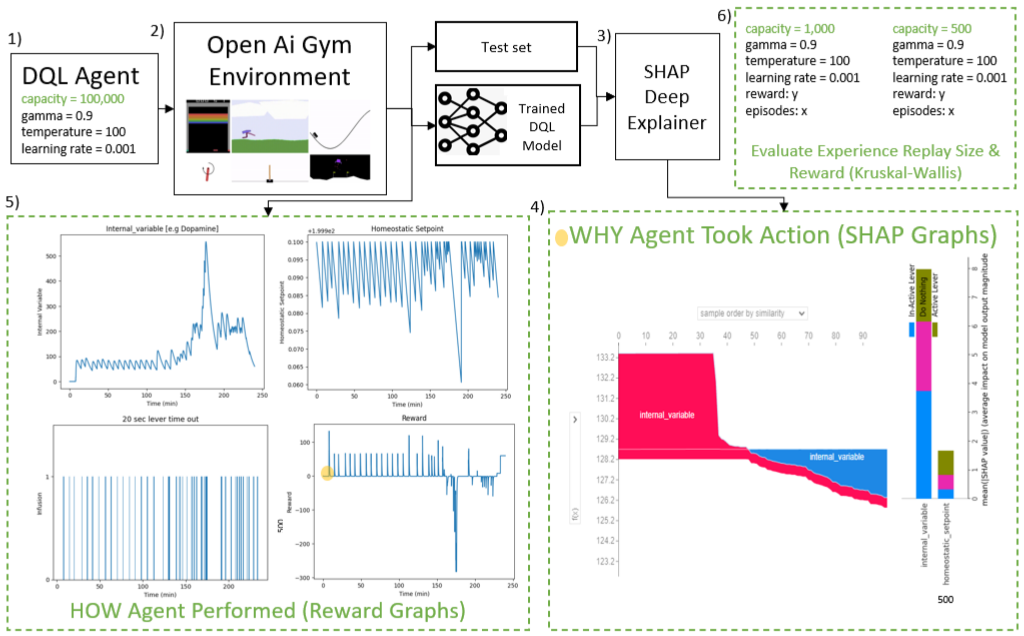

Figure 6: Experiment Process Design: 1) Create DQL Agent and set Experience Replay size, 2) Run the simulation to create test dataset, train Deep Q-learning model, and graph cumulative reward. 3) Run SHAP Deep Explainer to create SHAPly values. 4) Use SHAP Graphs to explain why actions were taken in particular situations. 5) Reduce Experience Replay to find a size where the reward does not decrease for total simulation time. 6) Evaluate that there is a statistically significant difference in means or variances of reduced Experience Replay groups when compared against the reward received.

In this chapter, the alternative hypothesis is defined, and Deep Q-Learning and Deep Convolutional Q-Learning model architecture is described along with their respective hyperparameters. An overview of the process for testing the Hypothesis is given (see Figure 17), along with information about Shapiro-Wilk testing, Kruskal-Wallis testing, and Dunn's post hoc testing. Finally, four experiments are outlined that will be used to test the alternative hypothesis. In Figure 17 a test sample is taken from the randomised Experience replay batch sampling that the Q-approximator neural network uses. This unseen sample, along with the trained model is passed to a SHAP Deep Explainer, where SHAP values are generated and visualized to help understand 'WHY' the Agent took the action in that given state.

3.1 Hypothesis

The alternative hypothesis (H_1) tests for:

> A difference ($p < 0.05$) in reward scores when Experience Replay is reduced.

3.2 Model Design

Two Deep Q-learning models are described below. These are needed to handle the different complex state spaces such as working with pixels as described previously in Sections 3.2.1 and 2.3.

3.2.1 Deep Q-Learning Architecture

The DQL Agent contains a neural network with an observation input that states are passed into, 30 neurons hidden, and a Q-value output layer that actions can be selected from. The neural network is approximating Q-values. It uses Adam's optimizer and an $\alpha = 0.001$ to calculate the loss error defined in Section 2.2.1 of the Temporal Difference defined in Section 2.1.6 with gamma $\gamma = 0.9$. To explore and exploit what actions to take the system uses the Softmax Policy with Temperature $T = 100\%$. 128 previous states are randomly sampled from Experience Replay plus 10% of experience is set aside and not shown to the system so as to be used later for the SHAP Deep Explainer.

3.2.2 Deep Convolutional Q-Learning Architecture

The DCQL Agent has three convolutional layers the first takes a single 80x80 pixel image as an input feature and outputs 32 feature maps 5x5 pixels in size. The second layer takes 32 feature map images and outputs feature maps as 3x3 pixels in size. The final convolutional layer takes 32 feature maps and outputs 64 feature maps 2x2 in size. These are passed into a fully connected neural network layer with a similar architecture defined in Section 3.2.1, except the number of

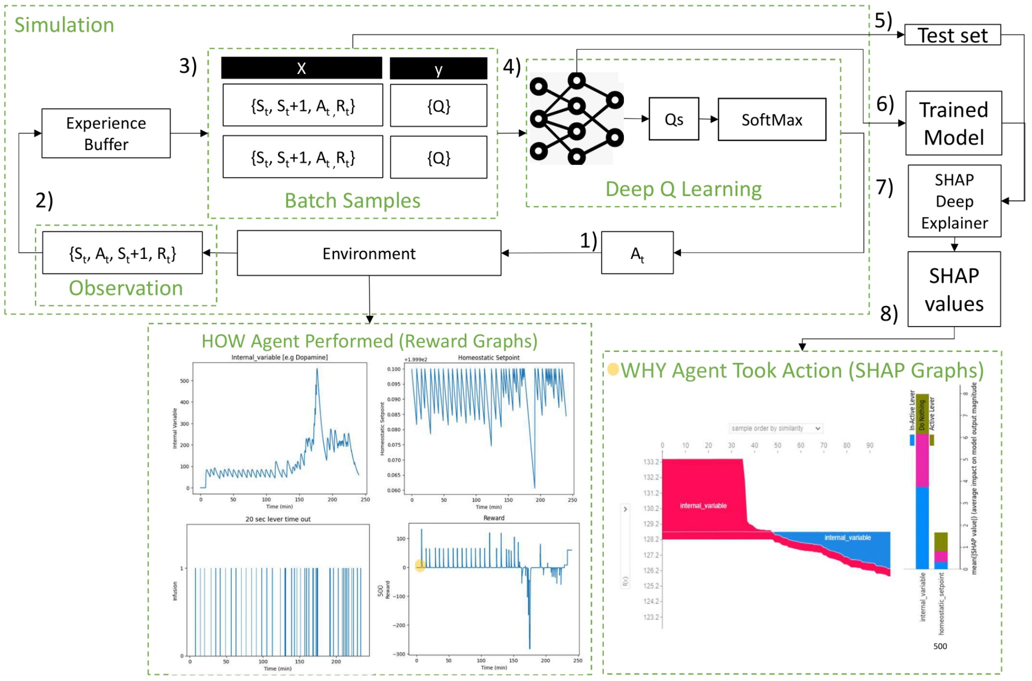

Figure 7: DQL Architecture: 1) Agent takes action. 2) Environment returns a reward + next state. 3) observations in experience replay are sampled. 4) The neural network learns q-values outputting the next best action with Softmax. 5) repeat until terminal state. Sample extracted from buffer. 6) model extracted. 7) Deep Explainer creates interpretable SHAP values.

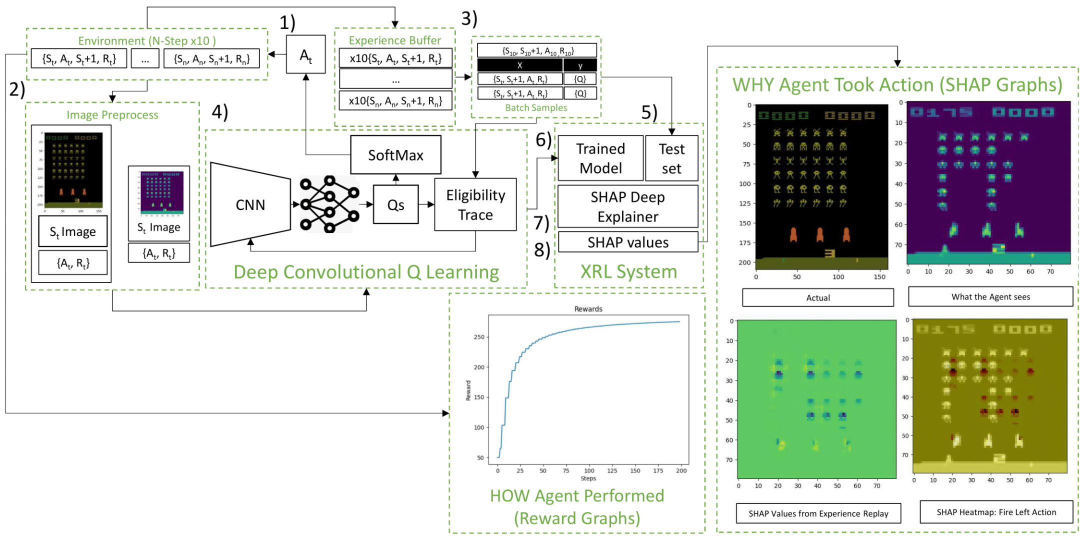

Figure 8: DCQL Architecture: 1) Agent takes action. 2) Environment returns a reward + next state. A state is 10 preprocessed images stored in an n-step history buffer. 3) observations in experience replay are sampled. 4) The neural network learns q-values outputting the next best action with Softmax. 5) repeat until terminal state. Sample extracted from buffer. 6) model extracted. 7) Deep Explainer creates interpretable SHAP values.

input neurons is dynamically determined by first creating a blank fake black and white image 80x80 pixels in size, max pooling the resulting layer through the three convolutional layers with a stride of 2 and flattening the layer into one dimension. The resulting quantity of input neurons is needed. 30 hidden neurons are used which are then passed to a second fully connected layer where the output is the number of actions the Agent can play. These outputs are the Q-values that are explored or exploited by the Softmax Policy to select an action. The DCQL Agent is trained using Eligibility Trace, defined in Section 2.3.3. It is similar to Asynchronous n-step Q-learning used by (Mnih et al., 2016), except Softmax is used instead of Epsilon Greedy as discussed in Section 2.1.7.

3.3 Hypothesis Testing

Experience Replay is set to 1×10^6 and reduced to 5×10^5, 1×10^5, 5×10^4, 1×10^4, 5×10^3, 1×10^3, and 5×10^2 transitions respectfully.

In order to test the hypothesis defined in Section 3.1 to determine if there are significant differences in reward received when Experience Replay is reduced, an Agent and the environment are initialized with an Experience Replay set to 1×10^6 transitions, then three steps occur. Firstly the environment is simulated for 200 episodes. Secondly, a reward by episode is stored and graphed to learn how the Agent performed. Finally, either a SHAP summary plot for state vector data or a SHAP heat map is generated for state image data to explain from Experience Replay samples why the Agent took an action in a given state. These previous three steps mentioned are then repeated until the Experience Replay size reaches the last size of 500 transitions.

When all simulations are complete a Kernel Density Estimate (KDE) plot of mean difference between different groups of Experience Replay sizes is created and statistical testing is conducted. Boxplots also consulted can be found in Section 7.2.

As described in Section 2.1.1, during the simulation, synthetic data is generated either mimicking experimental results or the laws of physics, depending on the simulation. The Agent will receive information about the environment's state and be allowed to make an action. The Agent decides what action to take and then takes it. The environment will provide a reward score and the next state. The reward amount is predefined (as discussed in Section 2.1.3. This reward data is continuous, collected in groups around what Experience Replay size was chosen during the simulation, and has equal sample sizes. Since the simulation is repeated with different Experience Replay sizes these groups are independent. ANOVA would be a suitable test here but the reward data may not be normally distributed and may contain outliers which violate assumptions required for ANOVA testing. A Shapiro-Wilk test can check if the reward data is in fact normally distributed (Shapiro & Wilk, 1965) but outliers will be difficult to remove. Instead, a non-parametric test like the Kruskal-Wallis test will be conducted with Dunn's test as a post hoc test to view how each group compares.

The Kruskal-Wallis test is based on the ranks of the data rather than the actual values. It ranks the combined data from all groups and calculates the test statistic, which similar to ANOVA, can measure the differences between the ranked group medians. The test statistic follows a chi-squared distribution with (k - 1) degrees of freedom, where k is the number of groups being compared. The null hypothesis of the Kruskal-Wallis test is that there are no differences in medians among Experience Replay size groups. The alternative hypothesis suggests that at least one group differs from the others (Kruskal & Wallis, 1952).

Dunn's post hoc test determines which Experience Replay groups have significantly different sizes, this together with KDE plots, boxplots and SHAP plots will help identify the groups with the smallest allowed capacity and answer the research question defined in Section 2.6.2.

3.4 Experiment Design

3.4.1 Experiment 1: Classic Controls CartPole Simulation

In this experiment, a cart balancing a pole is simulated. This environment corresponds to the version of the cart-pole problem described by (Barto, Sutton, & Anderson, 1983). A pole is attached by an unactuated joint to a cart, which moves along a frictionless track. The pendulum is placed upright on the cart and the goal is to balance the pole by applying forces in the left and right direction on

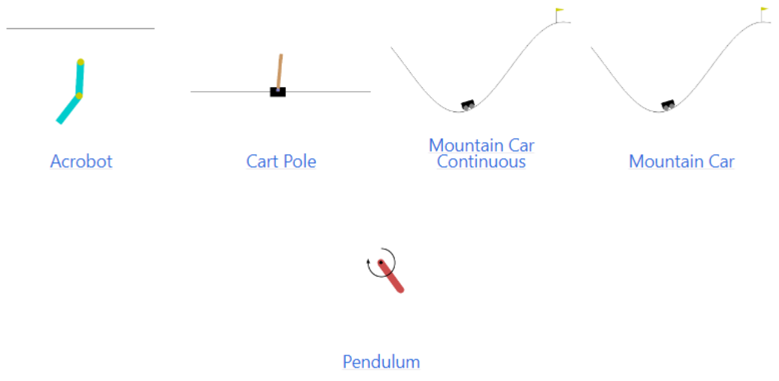

Figure 9: Classic Control Problems

the cart. A reward of 1 is given if the angle of the pole is less than 12 degrees and 0 otherwise. Actions available are 0 to push the cart to the left and 1 to push the cart to the right. A handcrafted Policy would move the car in the direction the pole is moving. This would require an understanding of mechanics and does not generalize well to other problems. The Agent instead learns the model trying to accumulate a reward for keeping the pole less than 12 degrees.

3.4.2 Experiment 2: Box2d Physics Lunar Lander Simulation

Figure 10: Box2d Physics Problems

In this experiment, a spacecraft needs to land at a targeted location on the surface of the moon. The further away from this location and not having two landing legs on the ground leads to a negative reward with landing on the target location a positive reward of +100 and -100 for crashing. Leg contact is worth 10. The main engine thrust is given -0.3. This is a rocket trajectory optimization problem where it is optimal, according to Pontryagin's maximum principle, to have engines on full or off.

3.4.3 Experiment 3: Atari Simulations Space Invaders & More

Figure 11: Atari Game Problems

In this experiment, 20 games from the classic Atari 2600 are used where each has a specific goal and reward structure. For example in Space Invaders a reward is given for every enemy shot down with a bonus reward for shooting the mother ship, in breakout a paddle and ball are used to remove bricks and in Montezuma Revenge, a reward is given for planning by navigating a castle collecting keys to unlock doors, stealing treasure and avoiding enemies.

3.4.4 Experiment 4: Custom Addiction Simulation

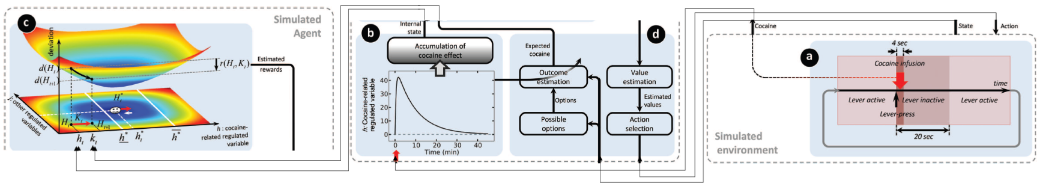

Figure 12: Homeostatic Reinforcement Learning System (Addiction) Problems

In this experiment a lab mouse can either rest or pull one of two levers, one of which self-administers cocaine over 4 seconds, followed by a 20-second time out during which the lever is inactive. A brain internal variable (h) then increases and falls gradually as the cocaine degrades in the rat's system. Internal variable (h) is similar to dopamine. The effect of continuous cocaine use collects over time. The rewarding value of the outcome (K) is the decrease in distance between the drive indicated by $d(H_t)$ from an internal state H_t and a homeostatic setpoint H^*. Every hit of cocaine causes a slow adaption that shifts the homeostatic setup forward. Absence of cocaine results in slow recovery to the initial setpoint. The Agent must learn to predict the drive-reduction rewarding values of its choices. This simulation is run for 3,600 episodes, replicating 4 seconds per episode up to 4 hours (Keramati et al., 2017). The reward is recorded and plotted on a line chart on the y-axis while the simulation time or steps taken is plotted on the x-axis, explaining how the Agent performed. Source code for this custom-built simulator can be found in Section 7.4 of the Appendices.

3.5 Summary

The alternative hypothesis (H_1) was defined as:

> A difference ($p < 0.05$) in reward scores when Experience Replay is reduced.

Two Deep Q-Learning Models were defined with their corresponding hyperparameters, this is to handle both simple vector state information and complex image pixel state information.

Hypothesis Testing was discussed where ideally ANOVA would be a suitable tool, some of the underlying assumptions about normality that ANOVA requires may be violated. A Shapiro-Wilk test is used later in Section 4 to check this and if violated a Kruskal-Wallis test is carried out instead. A Dunn's post hoc test is then carried out instead of a Tukey test.

Finally, four experiments were described where up to 23 simulations of varying complexity will be used to test the two Deep Q-Learning Models.

All of these will help answer the research question:

> "For simulations of varying complexity, what is the smallest Experience Replay buffer size allowed in Deep Q-Learning and why?"

3.5.1 Strengths of Design

Each experiment increases the complexity of the problem that the Agent must optimize for. The Custom Addiction Simulation in particular is an interesting choice as it is the safest way to test minimum experience size changes when exposed to the Sim2real gap as discussed in Section 2.4 previously.

3.5.2 Limitations of Design

Some simulations may be too complex to solve due to delay or timing of rewards prebuilt models from libraries such as Keras may be better suited for training, to control other hyperparameter selections such as n-step levels. Also as discussed in Section 2.4 previously, the Custom Addiction Simulator tries to tackle Sim2real by building upon the simulator designed by (Keramati et al., 2017) which mimicked experimental results of cocaine-infused lab rats. Calibration and evaluation of any simulation are important factors in overcoming the Sim2real gap and building confidence in results and claims. These issues will be considered in the next Chapter when results are evaluated and discussed.

4 Results, Evaluation & Discussion

In this chapter, the results from experiments are given in a summary table that shows the minimum Experience Replay allowed. Results of hypothesis testing are provided along with KDE plots, boxplots of reward scores for different sizes of Experience Replay in Sections 7.1, 7.2 and 7.4 of the Appendices. CartPole, LunarLander, SpaceInvadors, and Addiction are given focus in the following sections. To aid in debugging and interpretability so as to find the minimum Experience Replay for simulations of varying complexity, SHAP heat maps are presented as an additional tool to explain why the minimum size is suitable or not. Finally, the strengths and limitations of the findings are discussed with possible ideas for improvement.

4.1 Minimum Experience Replay Size Summary

The following Figure 13 summarises the results of four experiments, defined previously in Section 3.4, that placed a DQL and DCQL Agent inside 23 different simulations of varying complexity. Experience Replay was set to a 1×10^6 (Mnih et al., 2015; S. Zhang & Sutton, 2017; Fedus et al., 2020), then reduced.

4.2 Evaluation of HOW & WHY

A kernel density estimate (KDE) plot is used to visualize rewards and Experience Replay sizes, See Figure 22 in the Appendices section for all plots. The y-axis of the KDE plot indicates reward. The highest reward is desirable. The x-axis indicates different Experience Replay sizes. The furthest to the right on the x-axis indicates 1 million transitions and to the left 500 transitions. The shape of these plots is univariate. For each of the plots, a high kernel density indicates an Experience Replay size that can be set to the given Reward level that is most likely to occur here. Peaks also correspond to meaningful clusters, that is in some instances there may be two or three sizes to choose from with the densest more predictable than the other. SHAP explainability plots can help give the WHY so as to select among these. Also, outliers or unusual observations that lie in low-density regions can also be identified so as not to choose them as minimum size.

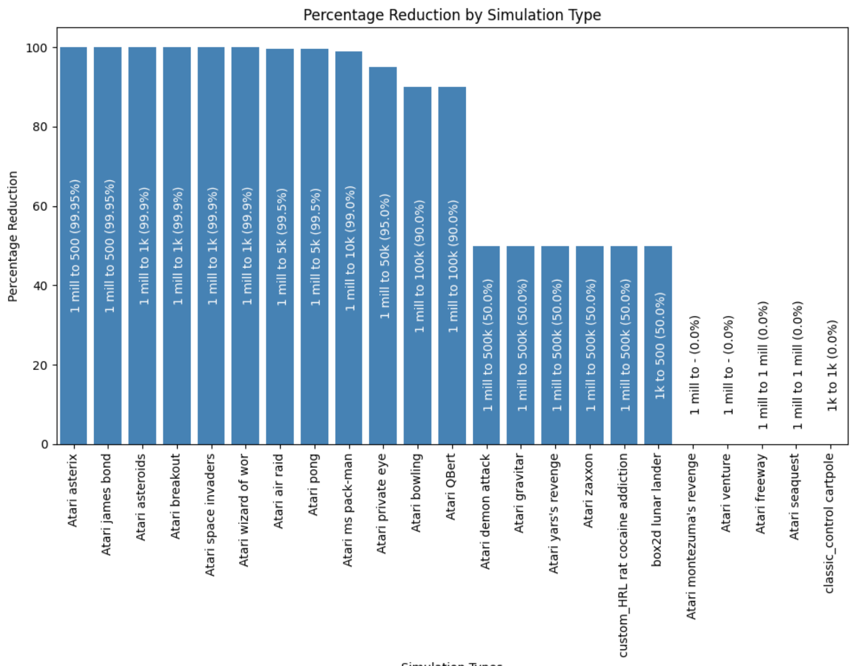

Figure 13: Minimum Experience Replay Size Allowed

CartPole, LunarLander, SpaceInvadors, and Addiction are given focus in the following sections. A Shapiro-Wilk test is run to confirm that ANOVA assumptions of normality are violated, if they are a Kruskal-Wallis non-parametric test is conducted with a Dunn's Post Hoc Test to compare the means of individual groups. The results of Dunn's test are compared with a KDE plot of the same distribution to understand HOW the agent performed after reducing Experience Replay then SHAP Heatmap Explainer is plotted to answer WHY the agent took an action given a state. Where possible the default SHAP summary and force plots were used but for real-time image sequence analysis a custom SHAP Heatmap Explainer inspired by SHAP's Deep Explainer Image Plotter for image classification explanations, is created. Source code for which can be found in Section 7.4 of the Appendices. SHAP is used to help confirm if the Agent's performance on a minimum setting is in fact working, as in it has learned the underlying concepts of the simulation or outliers are skewing the result.

4.2.1 Results 1: Classic Controls CartPole Simulation

Based on a Shapiro-Wilk test (Test Statistic=0.7308, p<0.001), the sample being tested does not follow a normal distribution. The low p-value indicates strong evidence against the null hypothesis of normality, meaning ANOVA cannot be used. Kruskal-Wallis test shows there are significant differences (Test Statistic=22,311.57 and p<0.001) between the medians of the Experience Replay size groups being compared. The extremely low p-value indicates strong evidence against the null hypothesis, supporting the alternative hypothesis that at least one group's median is different from the others.

When viewing the Dunn's Test pairwise Table 2 for cartpole it can be seen that the capacity size can be set between 1,000 and 10,000 as all other sizes have p-values further from them. The Default for this simulation is 1,000 so increasing is not desired. KDE Plot and SHAP Heatmap may help explain why no change can be made to the size of Experience Replay for CartPole.

Table 2: Dunn's Test for CartPole

	1×10^6	5×10^5	1×10^5	5×10^4	1×10^4	5×10^3	1×10^3	5×10^2
1×10^6	1.000	0.000	0.000	0.000	<**0.001**	0.109	0.000	0.000
5×10^5	0.000	1	<**0.001**	0.292	0.000	0.000	<**0.001**	<**0.001**
1×10^5	0.000	<**0.001**	1	<**0.001**	0.000	0.000	0.000	0.000
5×10^4	0.000	0.292	<**0.001**	1.000	0.000	0.000	<**0.001**	0.000
1×10^4	<**0.001**	0.000	0.000	0.000	1.000	<**0.001**	<**0.001**	<**0.001**
5×10^3	0.109	0.000	0.000	0.000	<**0.001**	1.000	0.000	0.000
1×10^3	0.000	<**0.001**	0	<**0.001**	<**0.001**	0.000	1.000	<**0.001**
5×10^2	0.000	<**0.001**	0.000	0.000	<**0.001**	0.000	<**0.001**	1.000

The KDE plot in Figure 14, indicates that 1,000 transitions are the most suitable for achieving the highest reward. The SHAP plot in the next section can help explain why.

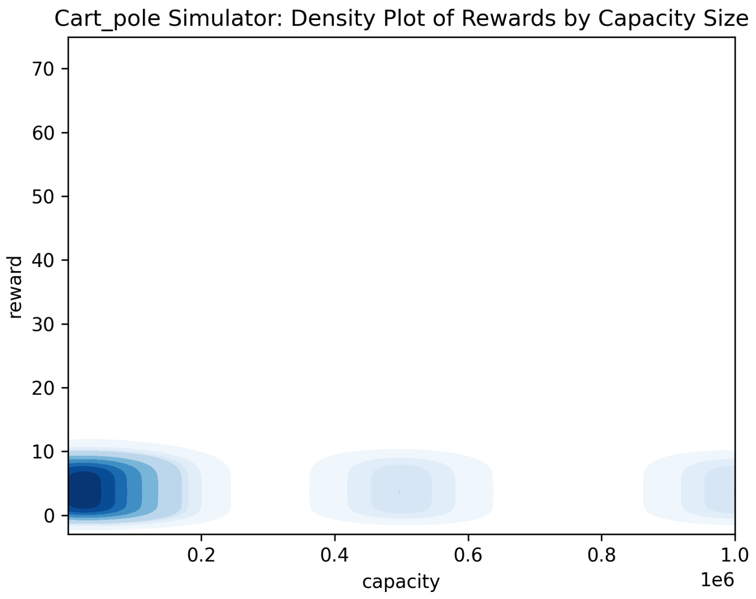

Figure 14: KDE Plot for Cartpole

The summary and forced plot in Figure 15 shows the most important features and the magnitude of their impact on the model. Red indicates a positive impact and blue a negative impact. Since the model is predicting the next Q-values vs the distance between the target or best Q-value (see Section 2.1.6) that in this state presented to the model for the first time, angular velocity for the pole and velocity of the cart and pole were more likely to cause the Agent to move the cart right than to the left. To a lesser extent observing the pole's angle and the cart position also forced the model to choose the right action. Since pole and cart velocity contribute the most to the model's decision-making to obtain high Q-values it supports that a decrease in Experience size would cause the Agent to lose context of these two input vectors and would be more biased to selecting to move in one direction (i.e. the right). The Experience Replay in this instance should not change from 1,000 transitions.

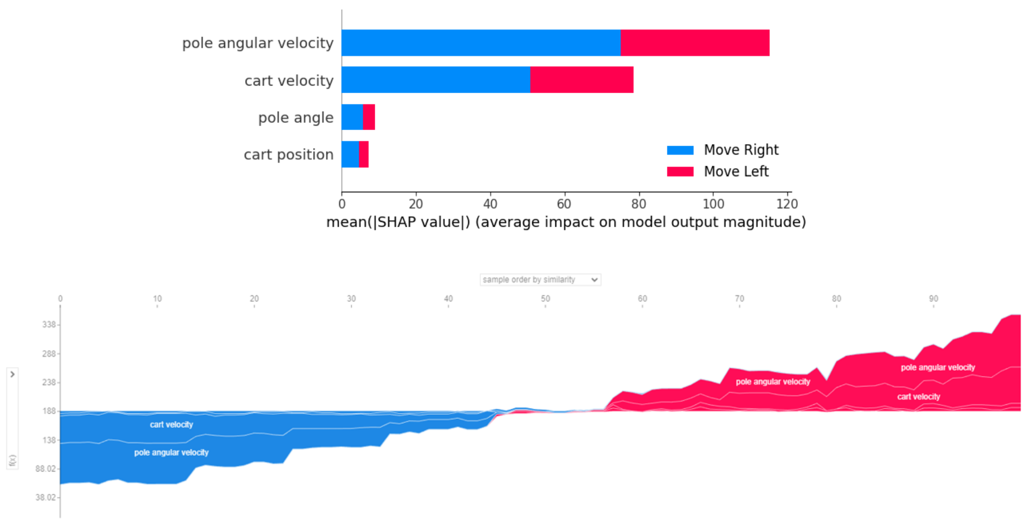

Figure 15: SHAP Heatmap Explainer for Cartpole: 1000 transitions

4.2.2 Results 2: Box2D Lunar Lander Simulation

The null hypothesis of normality from Shapiro-Wilk (Test Statistic=0.4664 and $p<0.001$) is rejected and we do not use ANOVA. A significant difference exists from Kruskal-Wallis test (Test Statistic=6789.99 and $p<0.001$) between the medians of the Experience Replay size groups being compared. A low p-value also indicates evidence against the null hypothesis, at least one group's median is different from the others.

The Dunn's Test pairwise Table 3 for Lunar Lander indicates that Experience Replay size can be set to either 500, 1,000 or 10,000 as all other sizes have p-values father are from them.

Table 3: Dunn's Test for Lunar Lander

	1×10^6	5×10^5	1×10^5	5×10^4	1×10^4	5×10^3	1×10^3	5×10^2
1×10^6	1.000	**<0.001**	0.000	**<0.001**	0.000	**<0.001**	0.000	**<0.001**
5×10^5	**<0.001**	1.000	**<0.001**	0.570	0.000	0.117	**<0.001**	**<0.001**
1×10^5	0.000	**<0.001**	1.000	**<0.001**	**<0.001**	**<0.001**	**<0.001**	0.000
5×10^4	**<0.001**	0.570	**<0.001**	1.000	0.000	0.317	**<0.001**	**<0.001**
1×10^4	0.000	0.000	**<0.001**	0.000	1.000	0.000	**<0.001**	0.000
5×10^3	**<0.001**	0.117	**<0.001**	0.317	0.000	1.000	**<0.001**	**<0.001**
1×10^3	0.000	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	1.000	**<0.001**
5×10^2	**<0.001**	**<0.001**	0.000	**<0.001**	0.000	**<0.001**	**<0.001**	1.000

Negative outliers in the KDE plot in Figure 16, indicate that 5000 transitions are not sufficient for achieving a high reward. However comparing 500 to 5,000 and then to 1,000 in Table 3, it shows 500 mean is vastly different to them. A choice between 500, 1,000, and 10,000 for Experience Replay size is permissible. 500 region appears darker indicating that it is more volatile in receiving negative rewards but more likely to settle at zero. A SHAP Heatmap is plotted next to aid in confirming if 500 is acceptable.

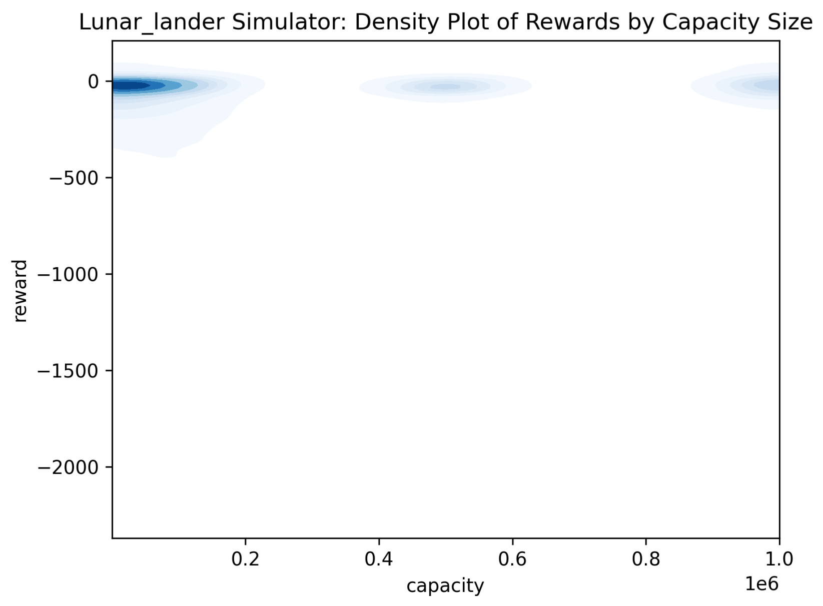

Figure 16: KDE Plot for Lunar Lander

The summary and forced plot show the most important features and the magnitude of their positive (red) or negative (blue) impact on the model. The model predicts the temporal difference of Q-values (see Section 2.1.6). The model seems to have fit around the x linear velocity with the angle contributing a small amount and the landing pad smaller still. Remembering back in Section 3.4.2 it is optimal to have engines on full or off. Crashing is worth -100. A lower Experience Replay will cause the craft to crash more and so overfit on one action but a negative reward is less than a capacity of 1000 or more. This might indicate that 500 is in fact suitable but more training time should be given to verify.

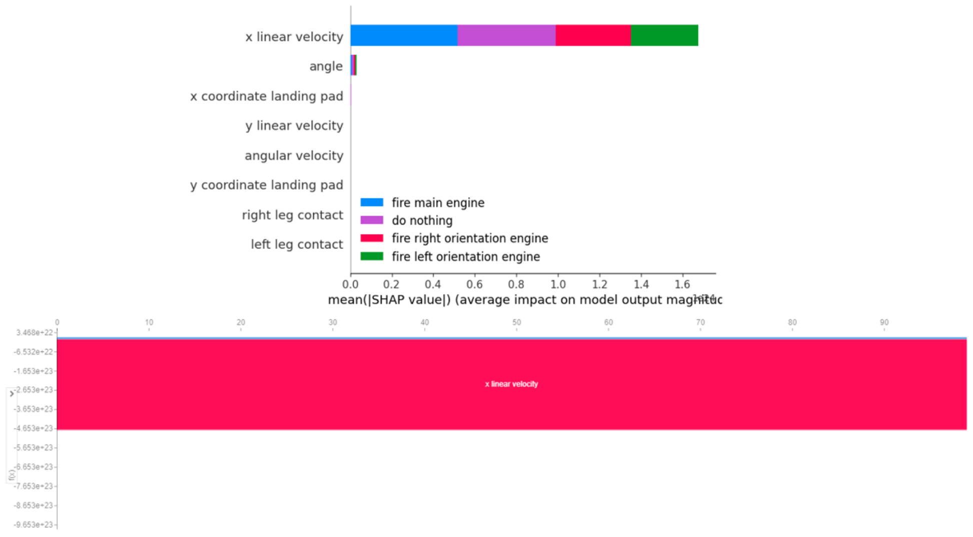

Figure 17: SHAP Heatmap Explainer for Lunar Lander: 500 transitions

4.2.3 Results 3: Atari Simulations Space Invaders

Results of Shapiro-Wilk test (Test Statistic=0.9578, p-value<0.001) are extremely small. This indicates that data does not follow a normal distribution and confirms that a non-parametric statistical test should be used instead of ANOVA. We reject the null hypothesis of the Kruskal-Wallis test (Test Statistic=777.57 and p<0.001) that the groups of Experience Replay sizes have equal medians.

The Dunn's Post Hoc Test pairwise Table 4 below for Space Invaders indicates an Experience Replay of 1,000 in size is vastly different to other groups but it and 500 are more so for 5,000. Both 500 and 1000 are strong candidates for being the minimum size. A KDE Plot and SHAP heatmap can help determine which.

Table 4: Dunn's Test for Space Invaders

	1×10^6	5×10^5	1×10^5	5×10^4	1×10^4	5×10^3	1×10^3	5×10^2
1×10^6	1.000	0.739	**<0.001**	0.013	**<0.001**	**<0.001**	**<0.001**	**<0.001**
5×10^5	0.739	1.000	**<0.001**	0.031	**<0.001**	**<0.001**	**<0.001**	**<0.001**
1×10^5	**<0.001**	**<0.001**	1.000	0.065	0.638	**<0.001**	**<0.001**	**<0.001**
5×10^4	0.013	0.031	0.065	1.000	0.170	**<0.001**	**<0.001**	**<0.001**
1×10^4	**<0.001**	**<0.001**	0.638	0.170	1.000	**<0.001**	**<0.001**	**<0.001**
5×10^3	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	1.000	**<0.001**	**<0.001**
1×10^3	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	1.000	**<0.001**
5×10^2	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	1.000

The KDE plot in Figure 14, indicates that the 1,000 transitions setpoint is the most suitable for achieving the highest reward. However, the KDE plot indicates outliers which could be driving up the high score. 500 is a more likely candidate but is negatively affected by outliers whereas 1,000 is not. The highest score is desired so a Shap heatmap will highlight if the Agent is making the right decisions with such low Experience Replay.

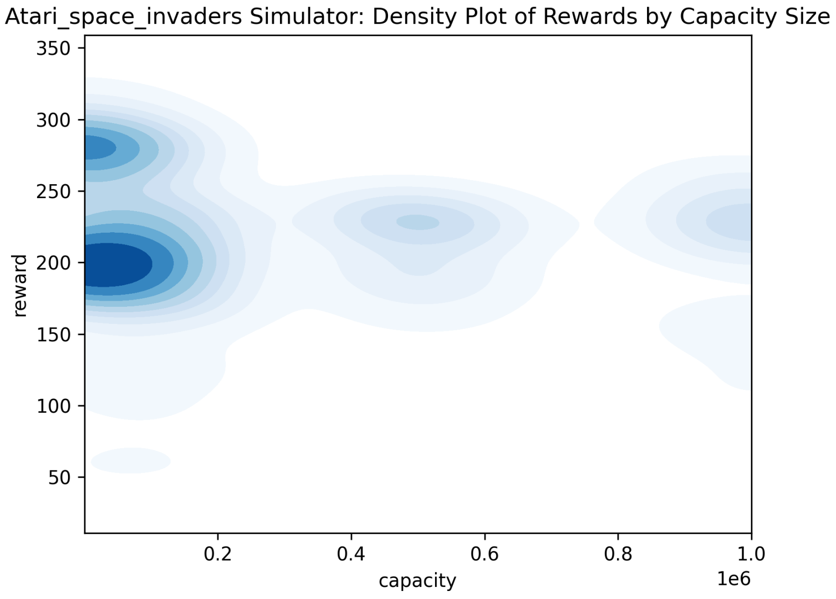

Figure 18: KDE Plot for Space Invaders

The Heatmap plot in Figure 19, shows the most important features and the magnitude of their impact on the model. In this case, the color map 'hot' is used. Bright areas have a positive impact on choosing an action and dark have a negative impact. When overlaid on what the agent sees, after preprocessing the original input image (discussed in Section 2.3), in Figure 19, the agent has identified the enemies as an important part of receiving the reward. It has also identified the trajectory of the enemies (moving left), the ones on the far right of the image are glowing bright yellow. When comparing this to the predicted Q-values we can see that Left-Fire action has the highest value. We can also see that the Agent is on the far left of the screen. Enemies on the far left are darker indicating that these are negatively influencing the decision to Right-Fire or Go-Left. Since SoftMax is used and Fire or Go-Right are also quite high, there is a high probability that these actions will be explored, to create a line of fire while enemies pass through or move right to gain a better firing position. The Agent seems to have grasped the basic concepts about the environment, more training may be required to improve its overall score but 1,000 as a minimum Experience Replay size is sufficient.

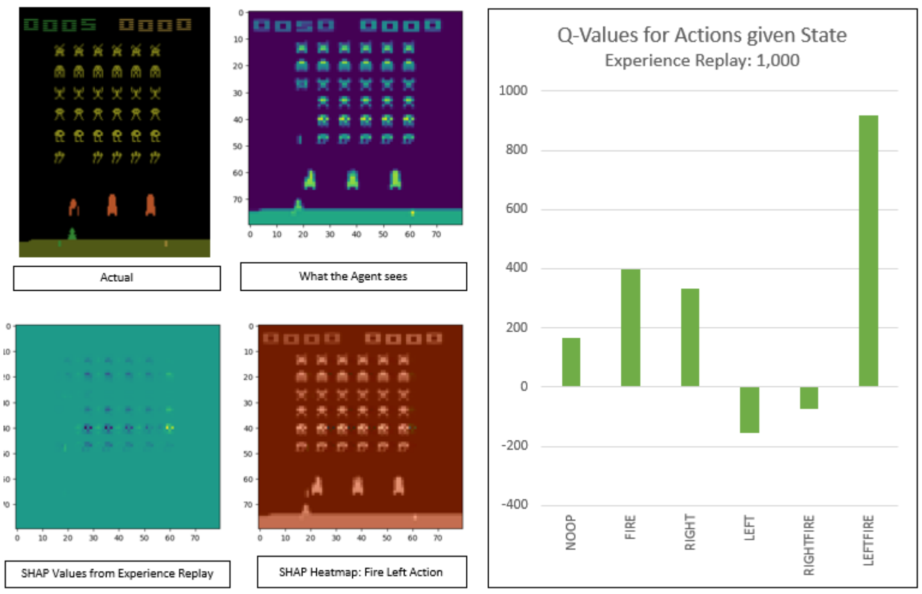

Figure 19: SHAP Heatmap Explainer for Space Invaders: 1,000 transitions

4.2.4 Results 4: Custom Addiction Simulation

Again we reject the null hypothesis of the Shapiro-Wilk test (Test Statistic=0.73 and $p<0.001$) that the data is normally distributed and choose the Kruskal-Wallis test instead of ANOVA. Kruskal-Wallis test (Test Statistic=22311.57, p-value: 0.0) indicates not all rewards are the same when Experience Replay size changes.

The Dunn's Post Hoc Test pairwise Table 5 for Addiction indicates that 500K is vastly different from 500 and 1k. Remembering from Section 3.4.4 each episode is 4 secs long and the agent is given 4 hours in the simulation or 3,600 episodes to be exposed to taking cocaine, which will give a huge positive reward then a continuous negative reward thereafter. The Agent needs more Experience Replay to understand how to maximize its reward given its simulated addiction.

Table 5: Dunn's Test for Addiction Sim

	1×10^6	5×10^5	1×10^5	5×10^4	1×10^4	5×10^3	1×10^3	5×10^2
1×10^6	1.000	0.000	0.000	0.000	**<0.001**	0.109	0.000	0.000
5×10^5	0.000	1.000	**<0.001**	0.292	0.000	0.000	**<0.001**	**<0.001**
1×10^5	0.000	**<0.001**	1.000	**<0.001**	0.000	0.000	0.000	0.000
5×10^4	0.000	0.292	**<0.001**	1.000	0.000	0.000	**<0.001**	0.000
1×10^4	**<0.001**	0.000	0.000	0.000	1.000	**<0.001**	**<0.001**	**<0.001**
5×10^3	0.109	0.000	0.000	0.000	**<0.001**	1.000	0.000	0.000
1×10^3	0.000	**<0.001**	0.000	**<0.001**	**<0.001**	0.000	1.000	**<0.001**
5×10^2	0.000	**<0.001**	0.000	0.000	**<0.001**	0.000	**<0.001**	1.000

The KDE plot in Figure 20, indicates that there are four distinct choices for Experience Replay size: 500, 5k, 500k, and 1 million respectively. A low Experience Replay reward mainly oscillates around zero or 100 and a high score is desired so 500k does look more suitable. A SHAP Heatmap together with domain-specific plots on addiction based on (Keramati et al., 2017), will help determine if 500k is suitable or not.

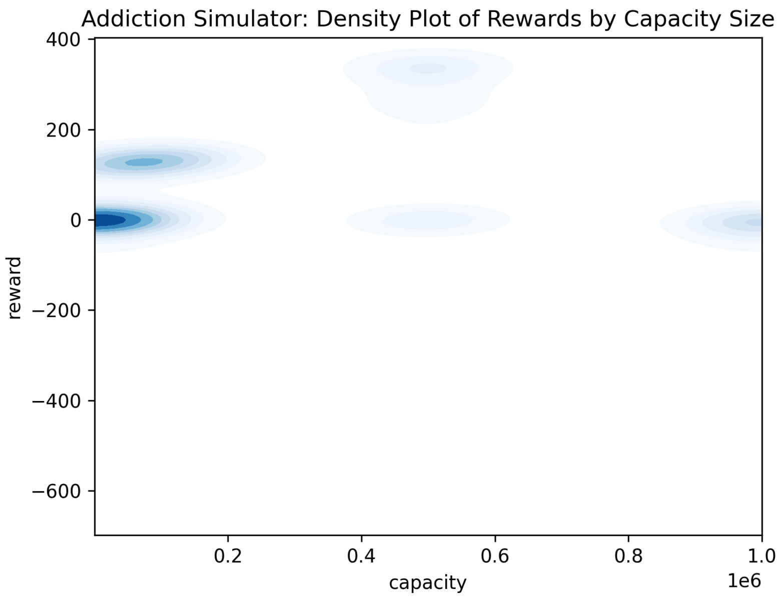

Figure 20: KDE Plot for Addiction

The SHAP forced plot in Figure 21 shows the most important features and the magnitude of their impact on the model. Red indicates a positive impact and blue a negative impact. In this case, the Homeostatic Setpoint is positively impacts the model's ability to get the reward. The summary plot above it indicates that changes in the Homeostatic setpoint impact the model more than the internal variable dopamine when presented with a state sample from experience replay, that was sampled about 2 hours into training. The Agent has fitted to the active layer negatively impacting reward, do nothing not affecting reward, and inactive positively affecting reward. Again, remembering Section 3.4.4, the Agent must learn to predict the drive-reduction rewarding values of its choices. This is the delta reduction of moving close to the Homeostatic setpoint at 200. Cocaine shifts the setpoint which causes the agent to chase it causing a repeat use of cocaine. This can be seen from the custom domain graphs where cocaine caused a large spike in dopamine but also shifted the setpoint to a new 'normal' and now requires cocaine to be taken at key intervals. 500k Experience Replay is sufficient to capture this domain complexity to maximize reward.

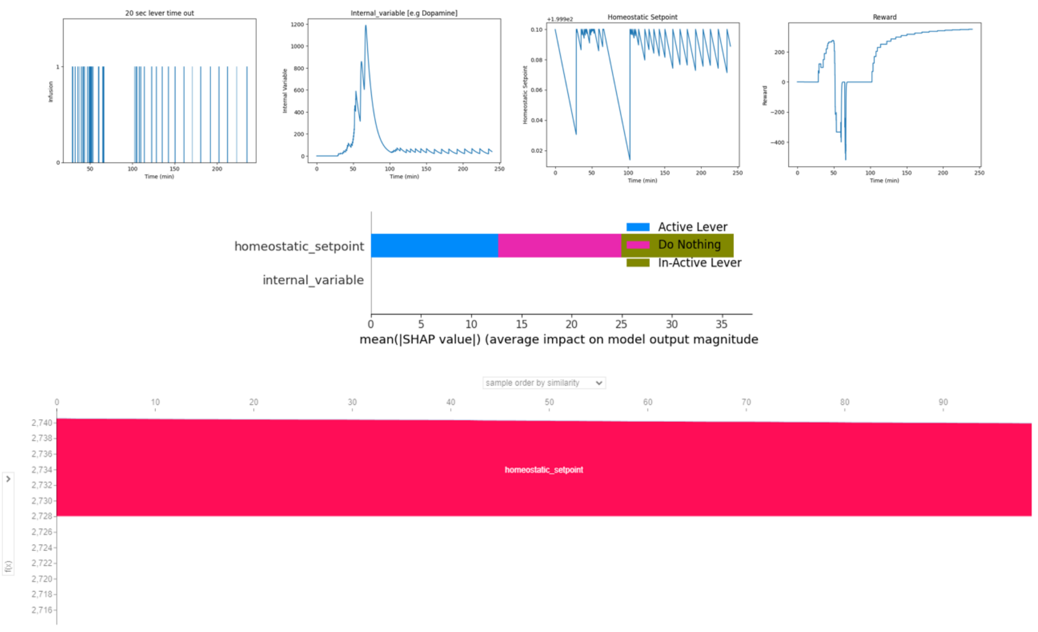

Figure 21: SHAP Heatmap Explainer and Domain Explainers for Addiction: 500k transitions

4.3 Discussion

Four experiments previously outlined in Section 3.4 were conducted where a DQL and DCQL Agents were placed in 23 simulations to answer the research question defined in 2.6.2. Table 13 was created that summarized for simulations of varying complexity, the smallest Experience Replay size allowed. Shap heatmaps were used to answer why they were allowed.

In order to accept or reject the research hypothesis that there was a difference ($p < 0.05$) in reward scores when Experience Replay is reduced, a Kruskal-Wallis test was conducted. This was in part due to a Shapiro-Wilk test informing that the reward data was not normally distributed violating assumptions required to use ANOVA. The results of each Kruskal-Wallis test show $p<0.001$. This supported the rejection of the null hypothesis and the acceptance of the alternative hypothesis. To obtain a deeper understanding of the difference in mean of different groups of experience replay sizes a Dunn's post hoc test was carried out. This test, together with a KDE Plot and SHAP heatmaps helped explain why the Agent's reward changed when Experience Replay was reduced and if the new minimum size is acceptable or not.

It was found that in 18 of 23 simulations, the Agent was tested in, Experience Replay could be reduced over 40% smaller than 1×10^6 transitions, defined in Section 2.2.2. Some simulations proved too challenging to get a suitable result or had to be kept high in order to receive the highest reward. The Atari game Montezuma's Revenge and Venture fell into the former while Freeway, Seaquest, and CartPole fell into the latter. Either high pre-planning is required for delayed rewards or timing sequences of actions was key to being successful, indicating that the n-step hyperparameter of 10 may have needed to change. CartPole was interesting, it could not change from its default value of 1,000 transitions. Explainability using SHAP heatmaps helped understand why, by reducing from 1,000 to 500 transitions the model would bias to only one direction to avoid crashing, to the right in this instance. As Experience Replay was increased this bias was reduced.

These results have helped answer the research question. In Deep Q-Learning, for simulations of varying complexity, a 40% from $1\times^6$ transitions for Experience

Replay capacity size is allowed because the Agent is still able to understand key concepts about many simulations from Experience Replay despite it being small.

4.3.1 Strengths of Findings

SHAP heat maps proved to be an excellent addition to existing statistical and domain expert tools, to improve interpretability and debugging of Deep Reinforcement models. SHAP helped to explain why a reduction in Experience Replay size was suitable or, in the case of CartPole, not suitable.

4.3.2 Limitations of Findings

When trying to determine a minimum size for Experience Replay between 500, 1,000 and 10,000 lack of training made it difficult to determine from the plots if 500 was in fact suitable. 500 had more negative outliers indicating that the Lunar Lander was crashing more at lower levels of Experience Replay but settled around a score of zero eventually. More training time should be given to verify a level of 500.

Some simulations could not be solved due to the simplicity of the Deep Learning Models and the complexity of delayed or timing of rewards. Prebuilt models from libraries such as Keras may be better suited for training in these experiments, to better control other hyperparameter selections such as n-step levels. These will be mentioned in the future work Section 5.5 of the next chapter.

5 Conclusion

5.1 Research Overview

Deep Reinforcement Learning (DRL) can optimize control and decision-making processes that are complex. However, it lacks explainability, limiting its widespread use in regulated environments, where rising cost, safety, and ethical concerns exist. As models generalize better with the addition of techniques such as Deep Learning and Experience Replay their internal workings become more complex which will only exacerbate these issues. Shapley Additive exPlanations (SHAP) are a popular tool to explain model predictions. The aim of this dissertation was to create an XRL-based system that produced SHAP heat maps to explain how input samples from the experience replay buffer affect the actions taken by a DQL Agent. These SHAP heat maps were further used to investigate the impact of reducing Experience Replay size on an Agent's reward received in simulations of varying complexity.

5.2 Problem Definition

Deep Learning models have become more distributed and resource efficiency has become the focal point. Current research is being directed towards handcrafted meta-controller systems like Agent57 and its eventual successor MEME (Kapturowski et al., 2023), this seemed to be going in the wrong direction away from generalized agents, a consistent aim of much of their prior research.

Experience Replay is not well understood, many believe it is flawed (S. Zhang & Sutton, 2017) and want to replace it with A3C by (Mnih et al., 2016) which requires more resources. Researchers who revisited Experience Replay focused on increasing its size (Bruin et al., 2018), the amount of experience being sampling (Fedus et al., 2020) and developed selection criteria for experience (Schaul et al., 2016; Ramicic & Bonarini, 2017; S. Zhang & Sutton, 2017; Sovrano et al., 2021). These efforts showed an improvement of model performance. However, increasing the Experience Replay size burdens resources required to process the additional transitions. To date, as far as this report has surveyed, no study has looked at minimum Experience Replay sizes allowed, nor could explain empirically why 90% of total transitions is a good rule of thumb (Bruin et al., 2018).

These problems led to the following research question being asked: *"For simulations of varying complexity, what is the smallest Experience Replay buffer size allowed in Deep Q-Learning and why?"*.

5.3 Design/Experimentation, Evaluation & Results

To test the research question, an alternative hypothesis was defined: *"There is a difference ($p < 0.05$) in reward scores when experience replay buffer capacity is reduced."*. Four Experiments were designed and conducted where a DQL and DCQL were put into 23 simulations of varying complexity while Experience Replay was reduced. KDE plots were created for rewards by Experience Replay Size, Kruskal-Wallis testing was carried out instead of ANOVA after confirming with a Shapiro-Wilk test that the data violated ANOVA's assumptions. Dunn's post hoc test, determined which Experience Replay groups had significantly different sizes, together with the boxplot and SHAP plots will help identify the groups with the smallest allowed capacity and answer the research question. In Deep Q-Learning. for simulations of varying complexity, a 40% reduction in 1×10^6 transitions for Experience Replay capacity size is allowed, because the Agent is still able to understand key concepts about many simulations from Experience Replay despite it being small.

5.4 Contribution & Impact

The contribution to the body of knowledge is an xRL optimizing method, that can be used in addition to traditional methods for tuning the Experience Replay size hyperparameter. This creative and visual method in 18 of 23 simulations tested, aided in the achievement of a savings of 40% or more in the reduction of Experience Replay size. This is less than 1×10^6 transitions.

5.5 Future Work & Recommendations

This research can be progressed in many interesting directions: Deep Q-Learning and Deep Convolutional Q-Learning could be tested in other complex simulations such as Minecraft's MineRL or in PettingZoo as a multi-agent reinforcement learning (MARL) problem to see the effect of reducing Experience Replay.

Other custom environments could be created and calibrated to test Experience Replay reduction to tackle the Sim2real gap. These could include manufacturing environments, inventory management, etc.

Different algorithms could be tested such as SARSA or Rainbow instead to help stabilize or remove the difficulty of solving harder environments like the Atari game Montezuma's Revenge. This would allow the minimum Experience Replay size to be found in these simulations.

Alternative selection strategies could be investigated, including the intriguing Explanation-Aware Experience Replay proposed by (Sovrano et al., 2021).

As Experience Replay is increased or A3C is used, resources will be burdened and distribution methods will only temporally solve inefficiencies of these underlying techniques. The magnitude of burdening resources should be better defined. Specifically quantifying the computational cost of running Deep Q-Learning models, including the kilowatt-hours of electrical energy used and the associated ton of carbon emitted to generate that energy. To gain societal acceptance in regulated industries such as manufacturing, finance, and medicine, an understanding of costs and their reduction will be important.

Image overlaying SHAP Heatmaps also allowed us to identify areas that are important to the model such as enemies, key items, or areas to navigate to. This may be of interest in the application of Transfer Learning where the Attention of a trained model is passed to assist in updating the weights of a smaller untrained model.

In conclusion, the proposed xRL-based system using SHAP values for Experience Replay can provide a more transparent, interpretable explanation of actions taken by a DQL Agent, which can aid in optimization for a better use of resources.

6 References

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-13*(5), 834-846. doi: 10.1109/TSMC.1983.6313077

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013, may). The arcade learning environment: An evaluation platform for general agents. *J. Artif. Int. Res.*, *47*(1), 253–279.

Bellman, R. (1957). *Dynamic programming.* Dover Publications.

Bhattacharya, A. (2022). *Applied machine learning explainability techniques: Make ml models explainable and trustworthy for practical applications using lime, shap, and more.* Packt Publishing. Retrieved from `https://books.google.ie/books?id=Kal3EAAAQBAJ`

Bilgin, E. (2020). *Mastering reinforcement learning with python: Build next-generation, self-learning models using reinforcement learning techniques and best practices.* Packt Publishing. Retrieved from `https://books.google.ie/books?id=s0MQEAAAQBAJ`

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *Openai gym.*

Bruin, T. D., Kober, J., Tuyls, K., & Babuška, R. (2018, 1). Experience selection in deep reinforcement learning for control. *J. Mach. Learn. Res.*, *19*, 347-402.

de Ponteves, H. (2019). *Ai crash course: A fun and hands-on introduction to reinforcement learning, deep learning, and artificial intelligence with python.* Packt Publishing. Retrieved from `https://books.google.ie/books?id=9p0aywEACAAJ`

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., & Dabney, W. (2020). Revisiting fundamentals of experience replay. JMLR.org.

Ghavamzadeh, M., Mannor, S., Pineau, J., & Tamar, A. (2015, nov). Bayesian reinforcement learning: A survey. *Found. Trends Mach. Learn.*, *8*(5–6), 359–483. Retrieved from `https://doi.org/10.1561/2200000049` doi: 10.1561/2200000049

Hayes, T. L., Krishnan, G. P., Bazhenov, M., Siegelmann, H. T., Sejnowski, T. J., & Kanan, C. (2021, 10). Replay in deep learning: Current approaches and missing biological elements. *Neural Computation*, *33*, 2908-2950. doi: 10.1162/neco_a_01433

Heuillet, A., Couthouis, F., & Díaz-Rodríguez, N. (2021). Explainability in deep reinforcement learning. *Knowledge-Based Systems*, *214*, 106685. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0950705120308145` doi: https://doi.org/10.1016/j.knosys.2020.106685

Kapturowski, S., Campos, V., Jiang, R., Rakicevic, N., van Hasselt, H., Blundell, C., & Badia, A. P. (2023). Human-level atari 200x faster. In *The eleventh international conference on learning representations, ICLR 2023, kigali, rwanda, may 1-5, 2023.* OpenReview.net. Retrieved from `https://openreview.net/pdf?id=JtC6yOHRoJJ`

Keramati, M., Durand, A., Girardeau, P., Gutkin, B., & Ahmed, S. H. (2017, 3). Cocaine addiction as a homeostatic reinforcement learning disorder. *Psychol. Rev.*, *124*, 130-153.

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, *47*(260), 583–621. Retrieved 2023-06-14, from `http://www.jstor.org/stable/2280779`

Kumar, S., Vishal, M., & Ravi, V. (2022). *Explainable reinforcement learning on financial stock trading using shap.*

Lanham, M. (2020). *Hands-on reinforcement learning for games: Implementing self-learning agents in games using artificial intelligence techniques.* Packt Publishing. Retrieved from `https://books.google.ie/books?id=zlvIDwAAQBAJ`

Li, C., Zheng, P., Yin, Y., Wang, B., & Wang, L. (2023). Deep reinforcement learning in smart manufacturing: A review and prospects. *CIRP Journal of Manufacturing Science and Technology*, *40*, 75-101. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1755581722001717` doi: https://doi.org/10.1016/j.cirpj.2022.11.003

Li, Y. (2019). Reinforcement learning applications. *CoRR*, *abs/1908.06973*. Retrieved from `http://arxiv.org/abs/1908.06973`

Liessner, R., Dohmen, J., & Wiering, M. (2021). Explainable reinforcement learning for longitudinal control. In A. P. Rocha, L. Steels, & J. van den Herik (Eds.), (p. 874-881). SciTePress. (Publisher Copyright: © 2021 by SCITEPRESS - Science and Technology Publications, Lda.; 13th International Conference on Agents and Artificial Intelligence, ICAART 2021 ; Conference date: 04-02-2021 Through 06-02-2021)

Lin, L.-J. (1992, 5). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, *8*, 293-321. doi: 10.1007/BF00992699

Longo, L., Goebel, R., Lécué, F., Kieseberg, P., & Holzinger, A. (2020). Explainable artificial intelligence: Concepts, applications, research challenges and visions. In A. Holzinger, P. Kieseberg, A. M. Tjoa, & E. R. Weippl (Eds.), *Machine learning and knowledge extraction - 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 international cross-domain conference, CD-MAKE 2020, dublin, ireland, august 25-28, 2020, proceedings* (Vol. 12279, pp. 1–16). Springer. Retrieved from `https://doi.org/10.1007/978-3-030-57321-8_1` doi: 10.1007/978-3-030-57321-8_1

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In (p. 4768-4777). Curran Associates Inc.

Miralles-Pechuán, L., Jiménez, F., Ponce, H., & Martinez-Villaseñor, L. (2020). A methodology based on deep q-learning/genetic algorithms for optimizing covid-19 pandemic government actions. In (p. 1135-1144). Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3340531.3412179` doi: 10.1145/3340531.3412179

Mishra, P. (2023). *Explainable ai recipes: Implement solutions to model explainability and interpretability with python.* Apress. Retrieved from `https://books.google.ie/books?id=pKdxzwEACAAJ`

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., … Kavukcuoglu, K. (2016, 11). Asynchronous methods for deep reinforcement learning. In M. F. Balcan & K. Q. Weinberger (Eds.), (Vol. 48, p. 1928-1937). PMLR. Retrieved from `https://proceedings.mlr.press/v48/mniha16.html`

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529-533. Retrieved from `https://doi.org/10.1038/nature14236` doi: 10.1038/nature14236

Ramicic, M., & Bonarini, A. (2017). Attention-based experience replay in deep q-learning. In (p. 476-481). Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3055635.3056621` doi: 10.1145/3055635.3056621

Ras, G., Xie, N., van Gerven, M., & Doran, D. (2022, 5). Explainable deep learning: A field guide for the uninitiated. *J. Artif. Int. Res.*, *73*. Retrieved from `https://doi.org/10.1613/jair.1.13200` doi: 10.1613/jair.1.13200

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. In Y. Bengio & Y. LeCun (Eds.), *4th international conference on learning representations, ICLR 2016, san juan, puerto rico, may 2-4, 2016, conference track proceedings.*

Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, *52*(3/4), 591–611. Retrieved 2023-06-14, from `http://www.jstor.org/stable/2333709`

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the 34th international conference on machine learning - volume 70* (p. 3145–3153). JMLR.org.

Sovrano, F., Raymond, A., & Prorok, A. (2021). Explanation-aware experience replay in rule-dense environments. *CoRR*, *abs/2109.14711*. Retrieved from https://arxiv.org/abs/2109.14711

Strubell, E., Ganesh, A., & McCallum, A. (2020, Apr.). Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(09), 13693-13696. Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/7123 doi: 10.1609/aaai.v34i09.7123

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (Second ed.). The MIT Press. Retrieved from http://incompleteideas.net/book/the-book-2nd.html

Thirupathi, A. N., Alhanai, T., & Ghassemi, M. M. (2022). A machine learning approach to detect early signs of startup success. Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3490354.3494374 doi: 10.1145/3490354.3494374

Thompson, N. C., Greenewald, K., Lee, K., & Manso, G. F. (2021). Deep learning's diminishing returns: The cost of improvement is becoming unsustainable. *IEEE Spectrum*, *58*(10), 50-55. doi: 10.1109/MSPEC.2021.9563954

Tokic, M., & Palm, G. (2011). Value-difference based exploration: Adaptive control between epsilon-greedy and softmax. In J. Bach & S. Edelkamp (Eds.), *Ki 2011: Advances in artificial intelligence* (pp. 335–346). Berlin, Heidelberg: Springer Berlin Heidelberg.

Vilone, G., & Longo, L. (2021a). Classification of explainable artificial intelligence methods through their output formats. *Machine Learning and Knowledge Extraction*, *3*(3), 615–661. Retrieved from https://www.mdpi.com/2504-4990/3/3/32 doi: 10.3390/make3030032

Vilone, G., & Longo, L. (2021b). A quantitative evaluation of global, rule-based explanations of post-hoc, model agnostic methods. *Frontiers in Artificial Intelligence*, *4*, 160. Retrieved from https://www.frontiersin.org/article/10.3389/frai.2021.717899 doi: 10.3389/frai.2021.717899

Vouros, G. A. (2022, 12). Explainable deep reinforcement learning: State of the art and challenges. *ACM Comput. Surv.*, *55*. Retrieved from `https://doi.org/10.1145/3527448` doi: 10.1145/3527448

White, D. J. (1993, Nov 01). A survey of applications of markov decision processes. *Journal of the Operational Research Society*, *44*(11), 1073-1096. Retrieved from `https://doi.org/10.1057/jors.1993.181` doi: 10.1057/jors.1993.181

Wu, G., Fang, W., Wang, J., Ge, P., Cao, J., Ping, Y., & Gou, P. (2022, Dec 17). Dyna-ppo reinforcement learning with gaussian process for the continuous action decision-making in autonomous driving. *Applied Intelligence*. Retrieved from `https://doi.org/10.1007/s10489-022-04354-x` doi: 10.1007/s10489-022-04354-x

Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., & Fujita, H. (2020). Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, *538*, 142-158. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0020025520304692` doi: https://doi.org/10.1016/j.ins.2020.05.066

Yu, C., Liu, J., Nemati, S., & Yin, G. (2021, nov). Reinforcement learning in healthcare: A survey. *ACM Comput. Surv.*, *55*(1). Retrieved from `https://doi.org/10.1145/3477600` doi: 10.1145/3477600

Zhang, K., Zhang, J., Xu, P.-D., Gao, T., & Gao, D. W. (2022). Explainable ai in deep reinforcement learning models for power system emergency control. *IEEE Transactions on Computational Social Systems*, *9*, 419-427. doi: 10.1109/TCSS.2021.3096824

Zhang, S., & Sutton, R. (2017, 12). A deeper look at experience replay.

7 Appendices

7.1 KDE Plots

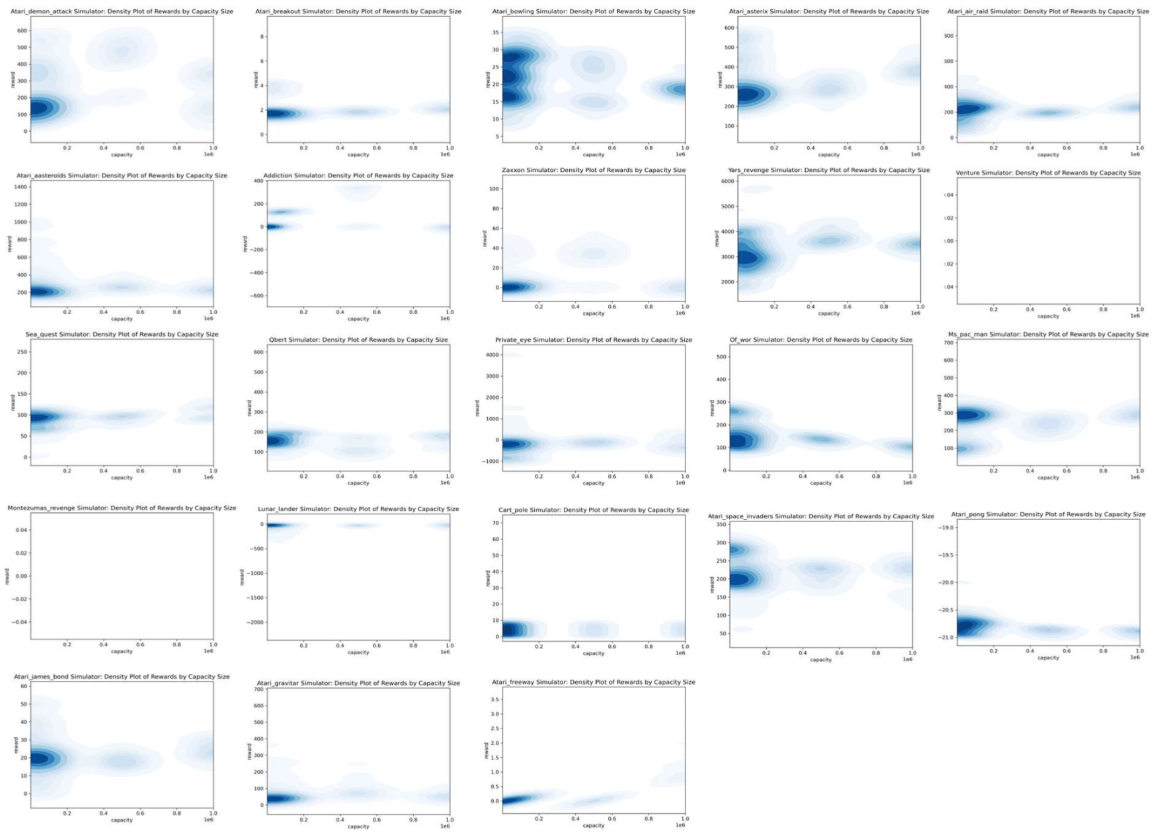

Figure 22: KDE Plot of Reward vs Experience Replay size

7.2 Box Plots

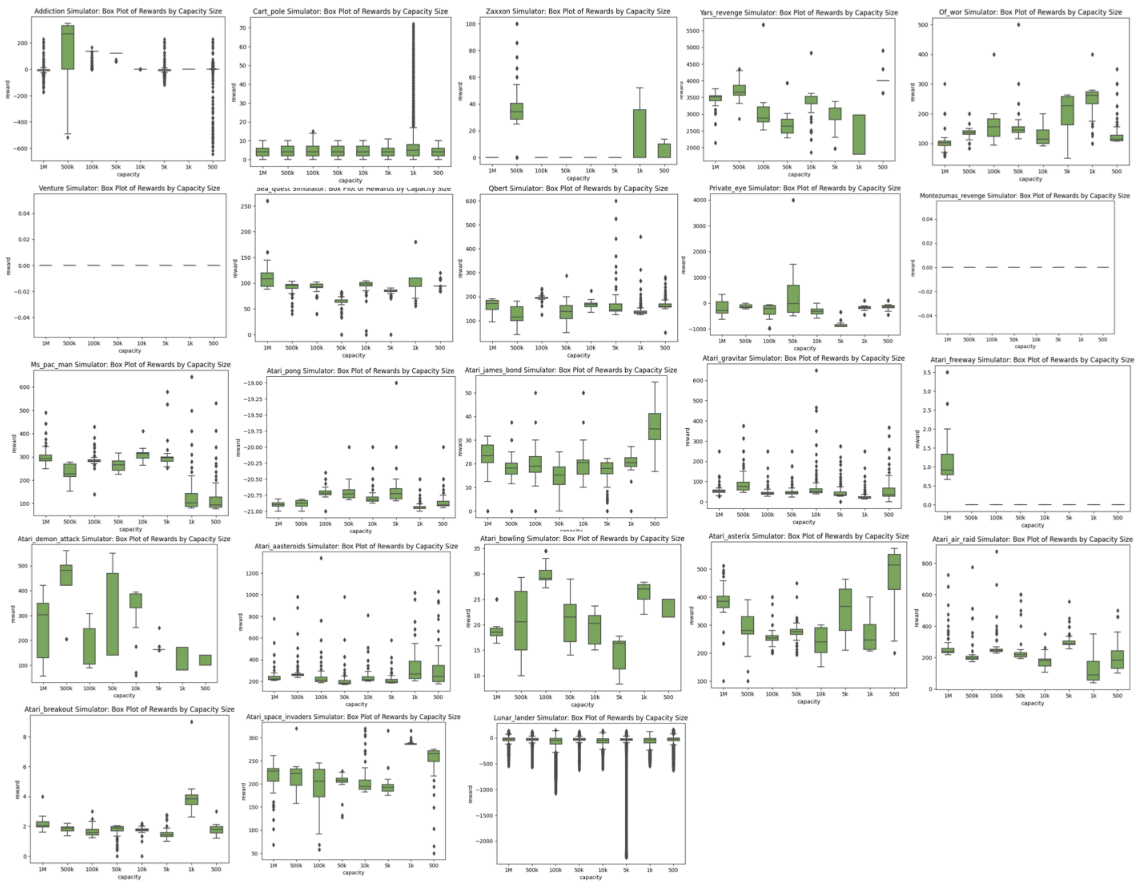

Figure 23: Box Plot of Reward vs Experience Replay size

7.3 Hardware & Software Specifications

Each experiment was run on an Asus GL502VMK Windows laptop. It had an Intel Core i7-7700HW CPU with 4 cores, 8 threads, and 2.8GHz clock speed, a 128GB SSD, 1TB HDD, 16 GBs of RAM, and a Nvidia GTX 1060 4GB graphics card. Episodes were restricted to 200, given hardware constraints. Most of the simulations require that pygame be installed so may not work in Google Colab. Models were built with PyTorch and run inside a Python 3.8 virtual environment. The custom addiction simulator built to run with gym requires installation, details can be found in GitHub repository.

7.4 Location of Source Code

Link to code, graphs, and tests are on GitHub at:

- **Simulation Results**:
  `https://github.com/rob-sullivan/tu060/tree/research/sims`
- **Hypothesis Testing Results**:
  `https://github.com/rob-sullivan/tu060/tree/research/tests`
- **Reward Datasets**:
  `https://github.com/rob-sullivan/tu060/tree/research/datasets`
- **Custom Addiction Simulator Source Code**:
  `https://github.com/rob-sullivan/tu060/blob/research/hrl_gym/hrl_gym/envs/hrl_env.py`