“THE ROLE OF GAME DIFFICULTY IN TRAINING DEEP RL AI FOR CLASSIC ARCADE GAMES”

Project Proposal

Course: Artificial Intelligence Masters

Student name: Kotsos Vasilis

Student Number:190033787

Content Page

Introduction 2

Critical Context 3

Methods and tools for analysis and evaluation 3

The use of Deep RL in Arcade Games 4

Proposed experiment set up and evaluation 6

Objectives and Testable Outcomes 8

Project Deliverables 9

Project management and work plan 10

Risks and mitigation strategies. 11

References 12

Figures:

Figure 1: Pong 4

Figure 2. DQN Architecture V Mnih *et al. Nature* 518, 529-533 (2015) 5

Figure 3: Atari Breakout ..... **Error! Bookmark not defined.**

Figure 4: Atari Boxing 6

Figure 5: Experiment procedure 7

Figure 6: Curriculum learning example 7

Tables:

Table 1: Baseline DQN Neural Network parameters 5

Table 2: Project objectives and testable outcomes 9

Table 3: Risks and mitigation strategies 11

Introduction

The explosion of deep learning over the past decade has resulted in several advancements in areas that previously computers performed very poorly in. Computer vision, text processing and speech recognition are a few areas that have been experiencing a great deal of advancement. The common denominator behind all these advancements is Deep Learning, and more importantly the Neural Networks that power it. Reinforcement Learning is a subfield that has seen rapid due to the advent of Deep Learning. The seminal paper published in Nature by Deepmind[1], demonstrated how Deep NNs can be used to achieve human level control on simple Atari games. Deep Reinforcement Learning is now a very hot research area with many open topics to explore. The topic that this proposal will focus on, is the role that game difficulty plays in training deep RL algorithms for arcade games. Although remarkable progress has been made in "solving" these games, the role that game difficulty plays in the training process is an underexplored topic. From a bird's eye view, the objective of the project will be to train agents on varying difficulty levels and compare them against each other in a test environment of low difficulty. This set up will allow for the thorough examination of the intuitive idea that agents which have been trained on harder difficulty, should exhibit better performance than the ones trained on lower difficulty when they are tested under easy difficulty.

Aims of the Project

1. Investigate the role of game difficulty in training AI for classic arcade games.
2. Experiment with different training strategies and schedules and explore whether the use of harder difficulty settings during training can prove to be beneficial.
3. Explore whether agents trained on high difficulty settings appear to have an edge, compared to their counterparts trained under easier difficulty settings.

To fulfil the aim of this project, the following objectives have been set:

1. Identify suitable Deep RL learning algorithms for Classic Arcade Games
2. Identify suitable training strategies for the learning algorithms.
3. Design and develop at least 2 learning algorithms that can exhibit good performance in at least 3 games.
4. Compare and contrast RL algorithms in terms of performance and discuss the best results.
5. Compare and contrast different training procedures and investigate what impact the use of high difficulty setting has if any.

Critical Context

It is of paramount importance that the project idea is surrounded by a well-founded critical context. The context is detailed below.

The Arcade learning environment used by the DQN papers [1][2] was released in 2013 with the purpose of providing a platform for evaluating AI technology using a standardized approach [3]. Prior to the release of this paper reproducibility of RL experiments was a big issue, since no set of standard environments existed. This version however lacked the option of difficulty levels and as such, all games included in the emulator came with their default difficulty setting. The arcade environment was later revisited and expanded upon in 2017[4]. This revision included the option of several difficulty settings for some of the original games. This project aims to take advantage of those additions. The authors of the paper state that including higher difficulty modes essentially adds new environments that nevertheless have similar underlying state representations to their easier counterparts. This is also a key point to this project, since the main goal is to explore whether an agent can learn a policy on a high difficulty setting that can generalize on easier settings and if not, try to justify the results. A hurdle that might be tackled is the concept of overfitting [5]. Put simply in RL terms, overfitting is exhibited when the agent has learned to behave in a very specific manner in one setting (high difficulty setting) while performing very poorly under a different setting (same game but easier difficulty). Depending on the nature of the games used and whether there is a 2-player component, the concept of Multi-agent learning will be tackled, and thus appropriate training and testing methods will be employed [6]. Finally, the approach of curriculum learning could be used to explore whether a better policy can be found by exposing the agent to different difficulty levels during the training process[[7]. Curriculum learning mimics the human experience, by slowly exposing the agent to more difficult situations the agent should exhibit constant learning and improvement by building on past experiences, just as humans in real life. For the purposes of this project, the experiments will focus on advanced versions of DQN algorithms, as detailed in [8]. Rainbow DQN is a combination of useful add-ons to a simple DQN algorithm. More technical detail is provided in the next section.

Methods and tools for analysis and evaluation

This section will describe in detail the software tools that will be in the project. It also highlights some of the RL DQN algorithms that are under consideration.

The project aims to evaluate how algorithms trained on higher difficulties perform on easier settings. To achieve this, several tools and techniques will be used. The learning algorithms will be of Deep-Q Network variety and will ideally be able to achieve a good score across all games. To achieve this, the project will mainly focus on some variety of DQN, approaching Rainbow DQN [8]. The learning algorithms do not have to achieve state-of the art results, but performance should be sufficiently high. The Rainbow DQN algorithm expands upon the simpler DQN version by adding a few tricks such as a Dueling Network and prioritized experience replay, it has demonstrated major improvements in performance when compared to simpler DQN versions, specifically when applied to arcade games. The code will be written in python, more specifically

the Pytorch deep learning library will be used to develop the RL algorithms. To develop the more advanced version of the DQN algorithms, reinforcement learning libraries that are built on top of Pytorch could be used, such as SLM Lab and PTAN (Pytorch Agent Net). The potential use of Tensorflow and its high level RL libraries will also be explored. The ideal software platform to use as an Atari game environment emulator is the OpenAI Gym library[10]. The OpenAI Gym is a toolkit that supports the development of RL algorithms by providing several environments that facilitate agent training. The toolkit includes many different types of environments, such as Toy text games and Classic control problems. This project will focus on the Atari game emulator environments that the toolkit provides.

The use of Deep RL in Arcade Games

This section offers details on how RL algorithms work in simple Arcade game environments. It also motives the use of Deep Neural Networks.

Simple algorithms used in RL, such as Q-learning, employ the use of tables to store information about states transitions and rewards these methods are called Tabular methods. Tabular Q-learning and the use of tables is not suited for playing any kind of arcade game, since the state space is prohibitively large to store in memory. As a quick reference, the Atari platform has a resolution of 210x160 pixels, and each pixel can take one of 128 colors [3]. The number of screens that are possible is 128^{33600}. This is the main reason why Deep RL algorithms are used to approximate the Q learning function. Instead of looking at a Q table to find which action is optimal for a specific state, a function can be used to map a state to an action. This learned function takes in states as input and outputs values for each available action. The action with the highest value can be though of as the optimal action to take in that state. A single state in a game can be represented as a single frame, for practical and theoretical reasons however, several frames are combined to form a complete state. This simple method allows the games to be solved as Markov Decision Processes (MDPs), which is a prerequisite for RL problems. Thich means that states can be distinguished from one another and that all information that is needed to take an optimal action in each state, is part of the state representation. For example, in Pong (Figure 1: Pong) a single frame cannot be used as a state, since it does not contain any information about the direction of the ball. More information must be used from subsequent frames to capture the information that is required for the agent to perform optimally (whether to move the paddle up or down). In Arcade games usually 4 sequential frames are combined to make up the state. While there may exist other longer-term dependencies in the environment, this method works well for simple arcade games. The features used to describe each state are extracted from batches of 4 images using Convolutional Neural Networks (CNNs). The use of fully connected networks is this setting is infeasible since as mentioned above a single frame is made up of 210x160 = 33600 pixels. If for arguments sake a fully connected input layer was to be used with 512 input units, the number of

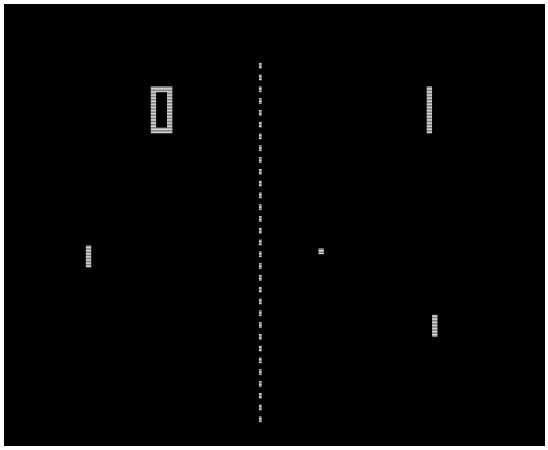


Figure 1: Pong

weights to be tuned would be 33600 x 512= 17,203,200, just for the first layer, so to keep the number of weights within reasonable bounds CNNs are used.

A good starting point for this project would be to emulate the Neural Network architecture presented in the original Deepmind paper (Figure 2. DQN Architecture V Mnih *et al. Nature* 518, 529-533 (2015)). The network used was a CNN, trained with stochastic gradient descent. It also employed the use of an experience replay buffer to help with creating training data that is independent and identically distributed. This is a requirement that must be met for the use of gradient descent. This version also used a target network to make the training more stable. Essentially a target network is a copy of the main network that is used to evaluate the maximum Q value of the next state. The target network is synchronized with the main network periodically. Finally, a simple e-greedy exploration method was used with a linear decay rate for e. using this method, the neural network will perform a random action or chose to exploit what it has learned. The agent at the start of training will chose to behave randomly most of the time (high e), while it will choose to exploit what it has learned close to the end of training (small e). The network architecture used in the paper is captured in the table (Table 1: Baseline DQN Neural)

Number of Layers	Optimization method	Activation function	Loss function
2 Conv + 2 Fully connected	Stochastic Gradient Descent (SGD)	RELU	Squared error (Regression)

Table 1: Baseline DQN Neural Network parameters

Built on top of this scheme, more advanced approaches can be pursued, such as using an n-step DQN which is simple unrolling of the Bellman equation which replaces one-step transition sequences with longer transitions of n-steps [8]. For a different exploration approach, noisy networks can be used. Noise is added to the fully connected layers of the DQN and overtime the noise is adjusted through back propagation. The noise added could be Independent or Factorized Gaussian noise. Finally, a significant improvement over a naïve sampling method from the replay buffer is the prioritized method. This method assigns priority to samples in the buffer according to which samples are deemed to be the most useful in learning a good policy [8]. These are some of the advancements that can be used to achieve high performance in playing Arcade Games.

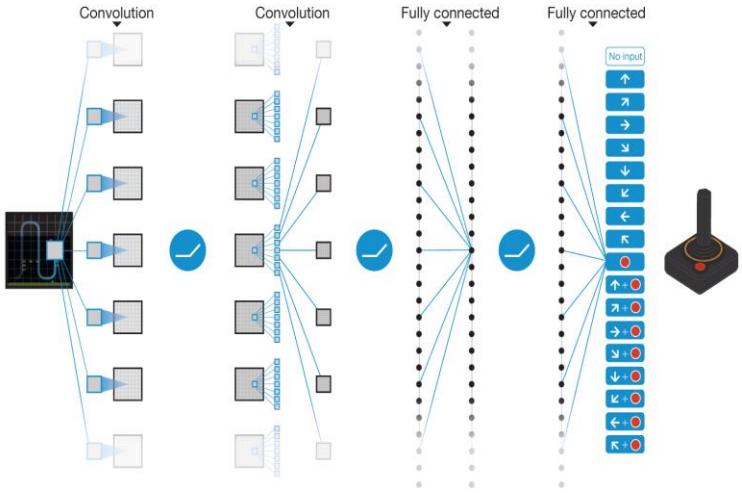


Figure 2. DQN Architecture V Mnih *et al. Nature* 518, 529-533

Proposed experiment set up and evaluation

This next section highlights how experiments could be set up, as well as how the agents could be evaluated.

Creating learning algorithms that perform well, even in simple Atari games, can be a difficult process. Careful consideration will be given on how the training and testing strategies must be structured, in order to present how game difficulty affects agent performance and more specifically whether or not an agent trained on higher difficulty can outperform an agent trained on easier difficulty, when they are tested on the 'easy' difficulty setting. The following strategies are being considered depending on the nature of the game used, and whether it is a 2-player game. Two games were chosen to illustrate the different training strategy options, as well as evaluation methods.

Game 1. Atari Breakout

Breakout (Figure 3: Atari Breakout) is a game about maximizing your score by moving the paddle at the bottom of the screen. In this simple game, the agent has control over the paddle can move it left and right. The objective is to direct the ball using the paddle to the blocks at the top of the screen. The player is awarded points when the ball hits the blocks. The design strategy for this experiment could be as follows:

For phase 1, two RL algorithms of the same architecture will be trained separately. The first one will be trained on the 'easy' setting of the game, and the second will be trained a harder setting. Both algorithms will be trained until they demonstrate convergence. A threshold score can be set to measure convergence, if the agents can attain the score, then they are thought to have converged. For the second phase the trained algorithms will be tested on the same game, with the difficulty setting set to 'easy' mode. By analyzing the behavior of the second agent we can find out whether it can generalize to the 'easy' setting and ultimately understand whether the training carried out on the harder setting proved to be beneficial. This method is depicted at (Figure 5: Experiment procedure)

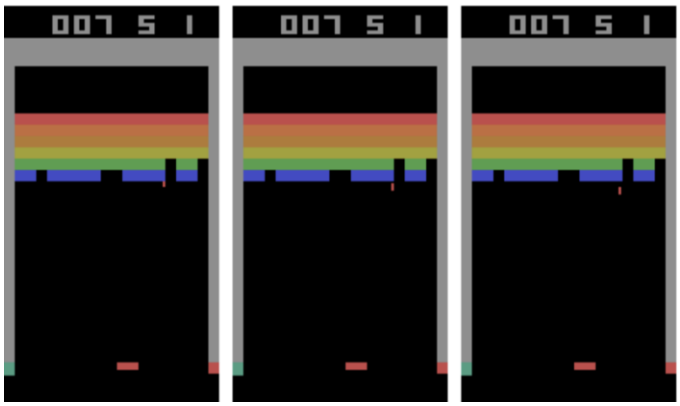
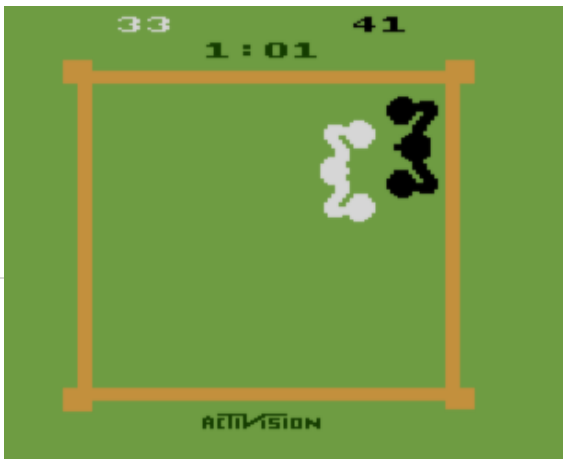


Figure 3: Atari Breakout

Game 2. Atari Boxing

Figure 4: Atari Boxing

The second game can present a slightly different evaluation methodology as it is a 2-player game. The objective in Boxing (Figure 4: Atari Boxing) is to punch the opponent as many times as possible, while avoiding their punches. The game ends if a player lands 100 punches or the timer has run out. In this



game each successful punch can be interpreted as a reward.

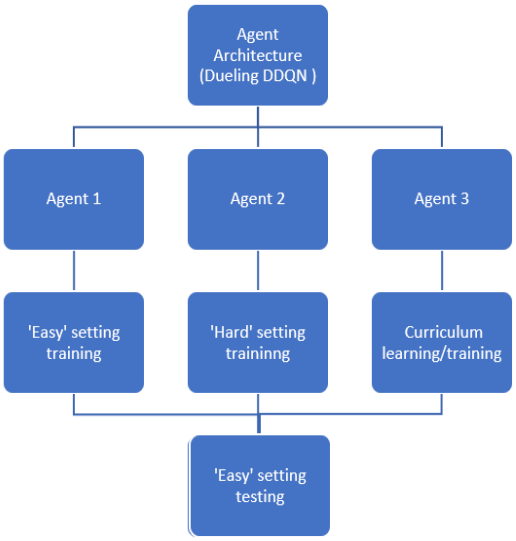


Figure 5: Experiment procedure

If the game completes without a player achieving 100 punches, the player with the highest score is declared the winner. Ties are also possible in this game. The evaluation methodology for a 2-player game of this nature could be as follows. Two agents can be trained in the same manner as the first game. The first agent would be trained on 'easy' difficulty and the second on 'hard' difficulty. As a test the agents would be put in an environment where they had to fight the in-game AI in the easiest difficulty and compare the results. Moreover, the trained agents can be pitted against each other to see which one has leaned a better policy and can generalize in a slightly different setting [6]. The agents would be allowed to play 10 rounds/games against each other and the agent with the most wins would be the winner (Figure 5: Experiment procedure). Finally, an approach that is wholly different can be taken to train an agent, known as curriculum learning [7]. This training approach would be slightly more involved as it would require training the agent on alternated opponents switching between in-game AI that is easy and in-game AI that is hard. The simplest approach that can be taken would be to increase the difficulty slowly and ideally present the agent a hard setting after it has mastered the easy one. This simple approach however has a weakness. As the agent is trained on harder settings, it may forget how to during easy ones. This could manifest as what is known as "catastrophic forgetting". To avoid such a scenario the training strategy might have to present the agent with different opponents, at different stages of the training. If the agent were to be trained on a different opponent after every few games, it could potentially lessen the chances of the agent overfitting and sticking to a policy that is good for a specific opponent, thus, demonstrating better results overall compared to other agents (Figure 6: Curriculum learning example). Overfitting may or may not present a significant challenge and can be tackled in many ways, for example if an agent trained on 'hard' difficulty overfits to that setting, it may exhibit very poor performance when placed on 'easy' difficulty, by changing the training parameters such as decay rate schedule and the architecture of the neural network, the effects of overfitting can be minimized. The report will discuss whether overfitting was encountered, and which steps were taken to minimize its impact. There exist many different schemes to curriculum learning, the goal of this project will be to find a suitable scheme that can offer results comparable to other training strategies

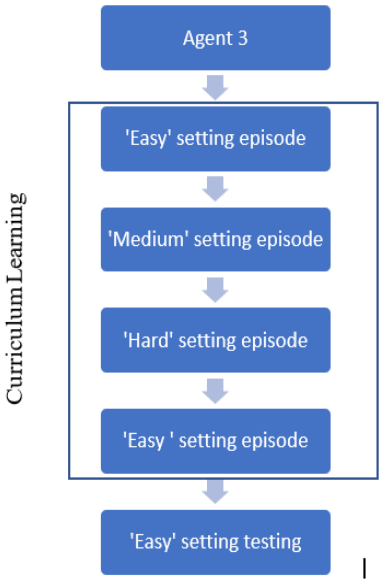


Figure 6: Curriculum learning example

Objectives and Testable Outcomes

This next section explores the objectives in more depth and sets out criteria that must be met for the successfully completing them.

1. It has been established that not all DQN architectures are beneficial for all games. For example, while a Dueling Double Deep Q network might perform extremely well in one game, it may not perform any better or perhaps perform worse than a vanilla DQN on a different game. As such suitable algorithms will be the ones that meet a performance standard across all 3 chosen games. Suitable algorithms will be one that can achieve better results than a vanilla DQN. The testing performance under the easy setting will be the performance metric that will show whether an agent is suitable, as it is possible that an agent can perform not so well during training, but perform very well during testing.

2. The training procedures that will be covered on this project will have to take advantage of the harder setting available for each game. These training strategies can be as simple as only training an agent on hard or easy settings and could be as advanced as employing a curriculum learning strategy. Since curriculum learning has many different training strategies, a strategy that is comparable in terms of results, to the simpler ones, must be found.

3. The results of the project will include two advanced RL algorithms that can perform well in at least 3 different games. This does not include a simple DQN, which will be used only as a baseline and a point of comparison.

4. In order to compare and contrast the algorithms, a detailed ablation study will be structured in the same vein as[8], where depending on the game discussed the agents will be compared on reward metrics or how many rounds they won. These comparisons will determine which agent exhibits optimal behavior in the easy setting. This optimal agent will have a specific DQN architecture and will have followed a specific training strategy. It is important to point out, that a truly 'optimal' policy for any game will not be learned, however the policy that outperforms all other policies will be evident.

5. After the optimal agent has been found, the impact of the training strategy will become clear, along with whether training an agent on hard settings presents an advantage. This of course, as previously stated, may be different depending on the game. For example, a Dueling Double Deep Q network when trained on a hard setting may exhibit better performance on Atari Breakout, but at the same time it may underperform in Atari Boxing, where the best agent might be a network of the same architecture, that was trained on easy settings. If training an agent on high difficulty results in better performance than training the same agent on easy settings, that would mean the high difficulty training setting was beneficial.

Objective	Testable Outcome
Identify suitable Deep RL learning algorithms for Classic Arcade Games	List of RL algorithms that can achieve at least as good performance as a simple DQN

Identify suitable training strategies for the learning algorithms.	List of 3 training strategies that will be used to train the RL algorithms. One of which will employ curriculum learning
Design and develop at least 2 learning algorithms that can exhibit good performance in at least 3 games.	List of RL algorithms that perform well on 3 chosen game. Excluding the vanilla DQN
Compare and contrast RL algorithms in terms of performance and discuss the best results.	Detailed ablation study that provides comparisons broken down by game, training strategy and DQN architecture.

Table 2: Project objectives and testable outcomes

The question to be tackled in this project has no clear answer yet. The outcome of this project is unknown and while it makes intuitive sense that agents which have learned how to behave in a harder setting should outperform the agents that were trained on an easier settings, the outcome remains to be seen. There are several outcome scenarios, the results may in fact go against human intuition and present an outcome where the agent performs badly in the 'easy' setting. This might not be the case for all games though and may not be true for all algorithm architecture, so it is of critical importance that the answer to be given makes those distinctions when they should be made in the form of a detailed ablation study. The reproducibility of these results is also of paramount importance so that can be accepted as reliable, this may be a sticking point that proves to be very difficult to achieve since there is a lot of randomness involved in the kinds of approaches that will be taken (Noisy networks, e-greedy policy, etc..) Multiple runs of the same code should demonstrate trusted results, by using statistics such as mean and standard deviation for reward measurements.

Project Deliverables

The project will offer the following deliverables upon its conclusion:

- A detailed report on the results of the testing carried out. The nature of the report will of course be quantitative, while also including certain remarks and comments about the performance of each agent and training approach.
- The full training and testing code of the agents in the form of .py files. Also, user documentation will be included in how to run the entirety of the code.
- Trained agents in the form of pickle files, that can be tested out of the box, without requiring any further training or tuning.
- A short video presentation summarizing the results achieved and showing agent behavior in the environment.
- A detailed project report that among other things will provide an answer to the central question that the project poses.

The project does not raise any ethical concerns, there will be no data collection from a sample population or any other kind of involvement of individuals. No legal issues or professional issues should arise from undertaking this project, all third-party contribution will be given appropriate credit when used. An ethics form is provided in the Appendix section of this document.

Project management and work plan

Below the Gantt Chart is given. It presents the several milestones that need to be met for the completion of the project, a detailed timeline is presented.

Gantt Chart	Begin Date	End Date	September	October	November	December
1. Environment setup and configuration	01/09/2020	15/09/2020				
2. Literature Survey research and Background Research						
2.a Algorithm RL survey	01/09/2020	15/09/2020				
2.b Curriculum learning approaches survey	01/09/2020	22/09/2020				
2.c Survey the games environments available and decide on 3 of them	08/09/2020	22/09/2020	*			
3. Small write-up in report						
3.a. Work on report introduction	22/09/2020	29/09/2020				
3.b. Work on report literature review	22/09/2020	29/09/2020				
4. System implementation						
4.a. RL Algorithms implementation - Easy & Hard training strategies	29/09/2020	13/10/2020				
4.b. RL algorithm testings, debugging and fixes	29/09/2020	13/10/2020				
4.c Curriculum learning strategy implementation	13/10/2020	27/10/2020		*		
4.d Work on "Methods used" part of the report	27/10/2020	3/11/2020				
5. System testing and debugging						
5.a. Review all implemented algorithms and implement testing procedures	03/11/2020	17/11/2020			*	
6. Experiments and report write up						
6.a. Gathering of evaluation statistics of agents	17/11/2020	01/12/2020				
6.b. Work on experiment evalaution part of report	17/11/2020	08/12/2020			*	
6.c. Review of results	08/12/2020	15/12/2020				
7. Complete report (Sections such as reflections, future work, etc...)	15/12/2020	21/12/2020				

The project aims to develop several pieces of software. As such, the approach of test-driven development will be followed [11]. Short testing periods will be followed up by quick coding sessions. This method will minimize the occurrence of bugs, and other technical mis happenings. The project will start with the environment set-up of the workstation, at the same time research will be carried out on the types of RL algorithms that can be used, and the different types of curriculum learning. After carrying out the research the beginning sections of the report can be written, more specifically the introduction and literature review (Tasks 3.a, 3.b). A supervisor meeting is proposed close to the end of the month to discuss the progress that has been made on the research [20/09/2020]. The second part of the project starts with the implementation of RL algorithms in the chosen game environments (Tasks 4.a-4.c). The implementation of the algorithms will run in parallel with testing and debugging (this is the approach of test-driven development). During this period, the bulk of the development will take place. The training strategies will be implemented starting with the ‘easy’ and ‘hard’ training strategies with curriculum learning following soon after. Overall, 4 weeks are dedicated to developing the core of the software, with one more week added for reviewing the code and implementing the testing procedures (Task 5.a). The second meeting is proposed at the end of the second month, around

[20/10/2020] so that the progress in the implementation can be discussed. Moreover, another meeting is proposed during the review of the implementation, around [15/11/2020].

The final part of the project consists of running the experiments several times and gathering statistics (Task 6.a). The most important part of the report will also be written during this period (experiment results and evaluation) (Task 6.b). A meeting is proposed around [8/12/2020], to discuss the report final parts and the project outcomes overall. The project ends with a final review (Task 6.c) and with the completion of the project report (Task 7), and the submission [21/12/2020]

Risks and mitigation strategies.

The project comes with risks that will have to be mitigated as much as possible. The risks below are mostly technically related. They are presented in the form of a table.

Risk Description	Likelihood of Risk	Impact on the Project	Mitigation Action
Getting environment set-up	Medium	High	Start as early as possible on the environment set-up. A virtual machine option will be explored to install a more stable version of AI gym.
Scope creep due to the software coding requirements	High	High	Setting a hard deadline for final software iteration after which the final part of report write-up will take place.
Hardware failures	Low	High	Move experiments to online environments or University premises if the need arises.
Data loss/ Results loss	Low	High	Purchase and use an external SSD as backup. Periodical automatic backups can be put in place.
Implementation of a learning algorithm in pure Pytorch.	Medium	Small	High level RL libraries can be used to implement RL algorithms.

Table 3: Risks and mitigation strategies

Some of the most important and impactful risks have been documented above. The mitigation strategy has also been set out

References

[1]Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540), pp.529-533.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*

[3] Bellemare, M.G., Naddaf, Y., Veness, J. and Bowling, M., 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47, pp.253-279.

[4] Machado, M.C, Bellemare, M.G., Talvitie, E., Veness, J., Hausknecht, M. and Bowling, M., 2018. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research*, 61, pp.523-562.

[5] Zhang, C., Vinyals, O., Munos, R. and Bengio, S., 2018. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*.

[6] Bhonker N., Rozenberg S., Hubara I., 2016. Playing SNES in the Retro Learning Environment. *ARXIV* arXiv e-prints, *arXiv:1611.02205*

[7] Narvekar S., Stone P., 2019. Learning Curriculum Policies for Reinforcement Learning, *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems* (pp. 25-33). International Foundation for Autonomous Agents and Multiagent Systems

[8] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. and Silver, D., 2018, April. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[9] Silver D., Antonoglou I., Schaul T., Quan T., 2016. Prioritized Experience Replay, *ICLR ARXIV* arXiv e-prints, *arXiv:1511.05952*

[10] Gym.openai.com. 2020. Gym: A Toolkit For Developing And Comparing Reinforcement Learning Algorithms. *Gym.openai*. [online] Available from: https://gym.openai.com/docs/ [Accessed 5 May 2020].

[11] Martin, R.C., (2011). *The clean coder: a code of conduct for professional programmers*. Pearson Education.

APPENDIX

PART A: Ethics Checklist.

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)? *e.g. because you are recruiting current NHS patients or staff?* *If you are unsure try - https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/*	**NO**
1.2	Will you recruit participants who fall under the auspices of the Mental Capacity Act? *Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - http://www.scie.org.uk/research/ethics-committee/*	**NO**
1.3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation? *Such research needs to be authorised by the ethics approval system of the National Offender Management Service.*	**NO**
A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent? *For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.*	**NO**
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	**NO**
2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	**NO**
2.4	Does your project involve participants disclosing information about special category or sensitive subjects? *For example, but not limited to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal*	**NO**

	offences and proceedings	
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? *Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/*	**NO**
2.6	Does your research involve invasive or intrusive procedures? *These may include, but are not limited to, electrical stimulation, heat, cold or bruising.*	**NO**
2.7	Does your research involve animals?	**NO**
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	**NO**
A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**		*Delete as appropriate*
3.1	Does your research involve participants who are under the age of 18?	**NO**
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	**NO**
3.3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	**NO**
3.4	Does your research involve intentional deception of participants?	**NO**
3.5	Does your research involve participants taking part without their informed consent?	**NO**
3.5	Is the risk posed to participants greater than that in normal working life?	**NO**
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	**NO**
A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK.		*Delete as appropriate*

If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form. **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		
4	Does your project involve human participants or their identifiable personal data? *For example, as interviewees, respondents to a survey or participants in testing.*	**NO**