

Computer Games Development

Project Report

Year IV

Richard Fleming

C00241212

27/04/2022

Contents

[Acknowledgements 2](#_Toc102045372)

[Project Abstract 2](#_Toc102045373)

[Project Introduction and Research Question 2](#_Toc102045374)

[Literature Review 3](#_Toc102045375)

[Evaluation and Discussion 5](#_Toc102045376)

[Project Milestones 6](#_Toc102045377)

[Major Technical Achievements 7](#_Toc102045378)

[Project Review 8](#_Toc102045379)

[Conclusions 8](#_Toc102045380)

[Future Work 9](#_Toc102045381)

[References 10](#_Toc102045382)

# Acknowledgements

I would like to thank the following people who assisted in completing this project including;

Oisin Cawley, who acted as my supervisor, providing advice and helping keep track of my progress throughout this project.

I’d also like to thank my fellow students for providing me advice whenever I encountered a problem.

# Project Abstract

The purpose of my project is to investigate and compare the advantages and disadvantages of two different training mechanisms of Artificial Neural Networks (ANN) in the context of interacting with a game world: Backpropagation and Reinforcement Learning respectively.

An ANN is a computational model designed to mimic a human brain. It uses a collection of nodes interconnecting using various patterns to create a facsimile of the neurons in the human brain. In order to do this, we train the ANN. There are various methods to do so. We could use supervised learning where we provide the ANN with training pairs, Unsupervised where we don’t, and reinforcement learning where the ANN learns from the feedback it gets from the environment.

During this investigation I will look at Backpropagation as a method of supervised Learning and Q-Learning as a method of Reinforcement Learning. I will be looking at the accuracy of the choices made by the ANN after each method, as well as the time and resources used during training. I will also be looking at how human like are the choices made by the ANN. This should give a comprehensive insight into which of the two methods would work best in a game environment.

# Project Introduction and Research Question

I’ve chosen to investigate and compare two training methods for artificial neural networks within the context of a game: Backpropagation and Reinforcement Learning respectively. What sparked this interest was the AI of the Alien in Alien: Isolation. Throughout the game, the developers managed to simulate the feeling of the Alien getting smarter and learning the players playstyle. In reality, this was achieved using 2 different AI and the Alien’s own behaviour tree which was unlocked as the player made their way through the game. In practice this could result in a player who would often hide in lockers having to change their tactics in order to combat the Alien.

I then learned about ANNs and how they actually can learn. Then that made me begin to think about the uses of ANNs in games development in general. For example, one could train an ANN with the purposes of playtesting, generating a model that mimics players in order to spot any bugs or problems a player could encounter. Alternatively, in a gameplay environment, what about creating an enemy that runs off of an ANN that learns and improves as the game goes on, either through the player’s input or the game environment itself. What kind of gameplay experiences could someone craft for the player with an enemy that adjusts to the players choice of playstyle, and what choices could the player make to navigate this. to that end, I chose to investigate two methods of training an ANN and weighting the pros and cons of the two methods to try and see which of the two would be better suited for interacting with a game environment.

# Literature Review

During the course of the project, I had read two works published regarding AI and ANNs: “Reinforcement Learning” 2nd Edition by Richard S. Sutton and Andrew G. Barto, as well as “Artificial Intelligence A Modern Approach” 3rd Edition by Stuart Russel and Peter Norvig.

“Reinforcement Learning”, as the name implies, went over Reinforcement Learning; it focuses on core online learning algorithms, with the more mathematical material set off in shaded boxes.

* Part I covers as much of reinforcement learning as possible without going beyond the tabular case for which exact solutions can be found. Many algorithms presented in this part are new to the second edition, including UCB, Expected SARSA, and Double Learning.
* Part II extends these ideas to function approximation, with new sections on such topics as artificial neural networks and the Fourier basis, and offers expanded treatment of off-policy learning and policy-gradient methods.
* Part III has new chapters on reinforcement learning's relationships to psychology and neuroscience, as well as an updated case-studies chapter including AlphaGo and AlphaGo Zero, Atari game playing, and IBM Watson's wagering strategy. The final chapter discusses the future societal impacts of reinforcement learning.

I found that Part 2 most relevant to my work, particularly the section on Nonlinear Function Approximation: Artificial Neural Networks. It starts off by showing and explaining the inner works of a feedforward ANN. It also goes on to compare Backpropagation to Reinforcement Learning. The authors spoke about how “The backpropagation algorithm can produce good results for shallow networks having 1 or 2 hidden layers, but it may not work well for deeper ANNs.” This in particular made me wonder about the limitations of the two methods, and helped me form the basis of the game I’d use for testing and experimentation.

Originally, I had envisioned making a simplified version of Megaman 3 and have the ANN actually play through a dumbed down version of a level from the game. But after reading this section, I decided it’s be best to reel the scope of the game as a whole to better compare the two methods as something too complex may result in unfair results. Likewise, as a result of bringing the scope back down, the use of a Deep Q-Learning method began to seem overly complex for the game as it was, so I opted to drop the Deep Learning aspect and use a simpler Q-Learning method.

“Artificial Intelligence A Modern Approach” On the other hand took a more general look at AI as a whole, having 6 sections, each covering a different topic:

* Part I introduced AI in general as well as the concept of Intelligent Agents interacting with an environment.
* Part II introduced problem solving, the idea of having AI or Intelligent Agents solving problems in their environments, including ones encountered in a game environment.
* Part III Spoke about knowledge, reasoning and planning, in which we design agents that can form representations of a complex world, use a process of inference to derive new representations about the world, and use these new representations to deduce what to do.
* Part IV then expanded on this by introducing uncertainty or nondeterminism to the AI, and how they make decisions when unexpected or unplanned situations crop up.
* Part V is of particular interest to me as it introduces the concept of AI learning and improving their behaviour, but we’ll circle back to this.
* Finally, Part VI talks about Communicating, perceiving and acting; how the Ai could communicate to a person, or perceive information.

As I mentioned, Part V was of particular importance to my work, as it discussed how we can design AI to learn. The subsection on reinforcement learning greatly influenced my work on the project as it dived into the concepts passive or active learning, as well as applications of reinforcement learning. In passive learning, the agent’s policy, a function that returns a feasible action for a problem, is fixed. This means when a certain state is met, it will always execute a certain action. On the other hand, an active agent must decide what actions to take. An example of active learning is Q-Learning.

This section even goes on to explain more about Q-Learning, discussing how, as it pays no attention to a policy, it can be more flexible than some other algorithms such as the SARSA(State-Action-Reward-State-Action) algorithm. From what I gathered, the freedom with which the Q-Learning algorithm approaches exploration, I felt it could lead to more interesting emergent gameplay as it tries to use the best Q value in any situation. As a result, this work helped solidify my choice on using the Q-Learning algorithm as a means of Reinforcement Learning.

# Evaluation and Discussion

During my investigation, I ran and trained the ANN Models through a randomly generated obstacle course where the ANN would choose the appropriate action based on the incoming obstacle. This was all done with the goal of tracking and identifying a number of key factors regarding the use of each training method:

1. Time and resources needed to train.
2. Accuracy of choices.
3. Similarity to human players.

In the case of point one, Backpropagation requires input and data beforehand to train the model off of, while Reinforcement Learning requires a great number of cycles to learn from the environment, and depending on the method of learning, may require large numbers of Q tables using normal Q-Learning methods.

The second point is more in regards to how long the ANN models can survive and make the correct choices in each situation. When given as much of an equal chance as possible, which of the two methods allows the ANN to progress further.

Finally, the last point is more subjective, regarding how closely does the ANN resemble a human player as they progress through the game.

From my experiments using backpropagation and a Q-Learning method, I found that while the backpropagation method provides consistent and reliable outputs and data, it also was subject to bias; since all of its input data came from me, it would develop the same habits as I do and tackle the obstacles in a similar manner.

On the other hand, the Q-Learning method allowed for emergent gameplay to occur as the ANN tackled the obstacles in its own way, often developing its own habits, but at the same time could provide unreliable data, and was prone to developing bad habits that would hinder it.

In regards to time and resources, backpropagation required little time and effort to provide training data for on account of the small number of inputs and situations in my game. the result was a csv containing the position and sizes of the obstacles encountered as well as my inputs in response, which I then used to generate a model in an h5 file. Reinforcement Learning meanwhile, as a result of choosing Q-Learning, I had to create 16 reward tables based on each situation the ANN would find itself in, and a Q table in the form of a csv that I could load and use later.

I felt that the backpropagation method provided gameplay that closer resembled a human, but the Q-Learning method allowed for emergent gameplay and new solutions to problems.

# Project Milestones

**Dec 13, 2021** SRS Created. On Schedule.

**Dec 16, 2021** TDD v1 Created. On Schedule.

**Jan 21, 2022** Basic game world set up. On Schedule.

**Mar 4, 2022** Infinite Runner Functionality. Behind Schedule.

**Mar 24, 2022** Backpropagation and Input Capture. Behind Schedule.

**Apr 25, 2022** Q-Learning Implemented. Behind Schedule.

# Major Technical Achievements

My main achievements would be my implementation of backpropagation and Q-Learning. For my implementation of backpropagation, I first had to research into Tensorflow and most importantly the Keras API to interact with it. Using Keras, I had to then figure out how to generate my ANN model using my captured input data. This included dividing my training data into my input and output datasets, the creation of my input and output layers of nodes, the compilation and optimisation of my model, actually training the model using that data over a number of epochs/iterations, and then saving that final model to an external file for later use.

Regarding Q-Learning, this gave me the most trouble in the entire project. To start, one of the initial problems I ran into was how I’d keep track of the game world. During my research, most of the time the non-Deep Q-Learning method seemed to mainly be used in static game worlds, such as navigating a maze. But my game world would constantly be changing as the obstacles moved. Therefore, I had to figure out a way to keep track of every state the game could be in. To do this, I decided to split the game world into a grid: as the current obstacle moves across the screen, it would update to reflect which grid space it occupied, letting me set up reward tables for each state the game could be in.

So, I then experimented with the number of grid spaces I used, as too little would lead to inaccurate choices on the ANNs part, as it may think the obstacle is a lot closer than it might actually be in game. But on the other hand, too many spaces would mean I would have to develop massive amounts of reward tables to reflect each state the game could be in. Eventually I settled on 2 rows of 8 spaces, allowing me to keep track of whether or not the ANN jumped, as well as where the obstacles and goals were without going overboard. This resulted in the creation of 16 tables, reflecting the 16 different states the game could be in at any one time. As the obstacles would move through the game world, they would update the current table being used based on their position.

I then set up a transition table that would keep track of the ANNs current state and position in the grid, allowing it to interact with the reward tables. Once training started, it would take random actions for a time to fill out an initial Q-Table to use as it trained, and would update each frame depending on the actions taken. The ANN would then play through the game for a number of iterations before saving its Q-Table to an external CSV to be loaded and used later, providing the best option in each game state based off of which action had the highest Q-value; continue running or jump.

# Project Review

While working on this project I was forced to scale heavily back from my initial vision of the ANN playing through a level of Megaman, to an infinite runner wearing Megaman paint. Looking back, my most obvious fault was my time management. Often times throughout the months I’ve spent working on the project, I’d often devote so much time to other projects that this one would fall to the wayside. I could have experimented with more technology for developing the project, such as trying to see if PyCharm may have been the better option rather than Jupyter Notebook.

As well as that, one piece of advice I’d give to someone staring this project would be to not be afraid to ask questions. Often times if I ran into a problem, I’d get stubborn and try for far too long to solve it myself rather than turning to either my supervisor or fellow classmates for advice. If I had been more proactive about asking questions, much of the hardship I had during the project would have been avoided entirely. For example, if I had asked my supervisor for advice regarding keeping track of the world state for Q-Learning sooner, I could have achieved much more with the time I had, such as the inclusion of more game inputs than simply jumping. During that time, I could have created more reward and transition tables to allow for more specific movements, such as sliding under obstacles.

On that note though, I am still proud of what I managed to achieve. The use of reward tables to keep of world states in particular, as well as the transition table being used to interact with them are both something I’m proud of being able to implement. As well as that, I’m glad I learned to make use of Keras during the development of the backpropagation model, as it made the whole process much easier to handle and understand, as well as being much more readable.

# Conclusions

After my experiments, I found that backpropagation provided gameplay that was more stable and consistent, but biased as a result of being trained solely by my inputs. On the other hand, while reinforcement learning allowed for emergent gameplay and unexpected way of tackling obstacles, it could also run the risk of developing bad habits and unstable gameplay that could only be ironed out with time and more iterations.

To answer the question of which methods to use, I believe that the answer lies somewhere in between. Neither method truly proved better than the other, and I feel that each method is simply better suited to different situations. Backpropagation could be a powerful tool for playtesting, as it can allow for the recreation of bugs or glitches consistently using player input, while the emergent gameplay of reinforcement learning may lend itself to crafting unique experiences for players.

One prospect I find interesting would be combining the two methods. for example, when creating an enemy, backpropagation could provide a stable and steady base for reinforcement learning to venture off of without losing consistency. One thing is certain though; both methods can lead to new and interesting developments in games, and games development.

# Future Work

If someone were to expand on this project at a later date, one idea would be to expand upon the number of actions the ANN is able take. One example would be the inclusion of a slide action. Using my current implementation, this could be achieved by expanding the number of reward tables used for Q-Learning, allowing for the tracking of more specific movements. This would also pose more of a challenge to the backpropagation model, and as a result allow for a more in-depth comparison between the two training mechanisms.

Another choice would be to change my implementation of reinforcement learning from simple Q-Learning to Deep Q-Learning. If implemented correctly, this could heavily cut down on the number of reward tables used. Not to mention the greater flexibility such an addition would allow, letting one create a much deeper game to be used for comparison.

Alternatively, someone could just include another reinforcement learning algorithm entirely rather than a Q algorithm. For example, the use of the SARSA method mentioned by “Artificial Intelligence A Modern Approach” earlier could be a good place of comparison between reinforcement learning methods.

One last thing I could see being an interesting development of this topic would be a third option for comparison; a combination of both backpropagation and reinforcement learning. For instance, one could use backpropagation to serve as a more stable and consistent base for the Deep Q-Learning algorithm to work off of.

# References

Sutton, R. and Barto, A. (2020). *Reinforcement Learning.* 2nd Edition. Cambridge, Massachusetts: The MIT Press

Russel, S. and Norvig P. (2010). *Artificial Intelligence: A Modern Approach.* 3rd Edition. Upper Saddle River, New Jersey: Prentice Hall