



Efficiency Comparison Between Curriculum Reinforcement Learning & Reinforcement Learning Using ML-Agents

Ala Jafar
Marco Tabell Johnsson

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Bachelor of Science in Digital Game Development. The thesis is equivalent to 10 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:

Author(s):

Ala Jafar

E-mail: aljg15@student.bth.se

Marco Tabell Johnsson

E-mail: majh17@student.bth.se

University advisor:

Dr Yan Hu

Department of Computer Science

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background. Machine Learning is a subset of Artificial intelligence which is a popular approach and widely used in computer science. Reinforcement learning is a type of machine learning, and it's about taking action to maximize the reward. Curriculum reinforcement learning is a way to train, by breaking up complex tasks into separate parts that would ease the learning rate.

Objectives. The existence of Curriculum reinforcement learning brings forth the problem of not knowing if it is a more efficient approach then Reinforcement learning. By answering our research question, "*Is curriculum reinforcement learning more efficient than reinforcement learning?*", we will be able to solve the problem.

Methods. We answered the problem by using the methods Implementation & Experimentation. We implemented an environment using the Unity Engine and the Unity Team developed ML-Agents toolkit. The training was done using the TensorFlow script in the Windows command console. To evaluate the results we used the TensorFlow feature Tensorboard, which will collect the training data and produce graphs of them. What we evaluate is the learning efficiency which we judge by the improvement in mean reward over the training steps. We trained multiple separate times to provide more valid results.

Results. The results showed that when they had the same values in the environment, they ended up with similar results. Possibly because our environment had a simple design. The results did not have a significant difference according to the *t*-test we performed.

Conclusions. We did not manage to reject the null hypothesis. Because of this and the results, our hypothesis is unsupported. In our experimentation, Curriculum reinforcement learning was not more efficient than Reinforcement learning, possibly because it was not complex enough.

Keywords: Machine Learning, Reinforcement Learning

Acknowledgments

We would like to thank our supervisor Dr Yan Hu for helping us.

Contents

Abstract	i
Acknowledgments	iii
1 Introduction	1
1.1 Reinforcement Learning	2
1.2 ML-Agents	3
1.3 Aim & Objectives	3
1.4 The structure of the thesis	4
2 Related Work	5
3 Method	7
3.1 Implementation	7
3.2 Development	12
3.3 Evaluation	12
4 Results	15
5 Analysis And Discussion	19
6 Conclusions and Future Work	23
References	25

Machine Learning is a popular approach in computer science and widely used. Machine Learning is used for different tasks in industries such as; healthcare, military, and commercial. During the Google I/O'19[13] conference, Laurence Moroney in his 'Machine Learning Zero to Hero' presentation explained the popularity of Machine Learning, with a case of the Rock Paper Scissors game. In the game, you can use your hand to make a rock, a paper, or a scissor. And the computer would recognize your hand and be able to play with you. To code this we would have to record images from the camera and then start decoding the images. Then we would have to tell the differences between a rock, scissors, and a paper, the shape of the hand, the skin tone. This simple game would end up taking a lot of time and becoming very complicated to program. This becomes simpler and less time-consuming using Machine Learning. This is because we can supply the answers to what is Rock, Paper, or Scissors. The Machine Learning algorithm will then try to match its image input or camera feed to these answers, depending on which it is most similar to.

Machine Learning is a subset of artificial intelligence (AI). According to *Si, Jennie et al.* [11] Machine Learning is the study of methods for constructing and improving software systems by analyzing examples of their behaviour rather than by directly programming them. Like in the previously mentioned game of Rock Paper Scissors Machine Learning algorithms are used in cases where the precise specification for behaviour is unable to be provided while the desired outcome can be. Examples of this are robot control and navigation or speech recognition. Similarly, there are tasks that a person might not know how to do, but they can still evaluate the performance. An example of this is master-level chess. Machine Learning can also be fitting for tasks that end up changing over time or with different users, this is because a programmer might not be able to know, ahead of time, the required behaviour of the program. Examples of this method of Machine Learning can be witnessed in browsing behaviour prediction on the internet. Machine Learning algorithms have also been used for data mining, finding patterns in data. Machine Learning builds a mathematical model based on the training data. The training data is the experience produced after it has been supplied with information on the task. Machine Learning consists of three main types; supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is often used as an introduction to machine learning. The design of supervised learning algorithms is to learn by example. Meaning that when training with supervised learning, the data is provided for the input and desired output. The algorithm will then look for patterns between the

provided input and output data. When it is done training the supervised learning algorithm can take new input and predict the correct output. Unsupervised learning in contrast to supervised learning does not get a corresponding output to its input. If it doesn't know what to do with the input how does unsupervised learning evaluate its performance? It can't measure its accuracy since it does not have the desired output. It will instead observe the data and look for patterns. Unsupervised learning will extract useful information from the input. It maps the input to an output based on what it is learning from the structure of the data.

1.1 Reinforcement Learning

The last main type of Machine Learning is Reinforcement Learning (RL). RL is utilized in situations where the desired behaviour is not available, but it is possible to score the behaviour. For example a poor reception on a mobile phone during a call. The person with the phone might move it around to try and find a better signal or just give up. There is no predefined destination that has a better connection so the person would have to search for it or give up if an acceptable signal cannot be found. This RL problem can be viewed as optimizing an unknown reward function. The system takes in a location that will, in turn, give a reward based on the unknown function. The goal that RL has is finding the input that gives the highest possible reward. RL does not get training examples in addition to not knowing the reward function. It does, however, get to decide the input, and it does observe the resulting reward of that input. The reward could be stochastic or deterministic. There are two main differences between RL and supervised learning. RL does not have a fixed distribution of input, and the goal is to find the best input rather than minimizing the loss overall inputs. If the reward is stochastic, the goal is instead to maximize the expected reward with regards to the randomness in the unknown function.

The learning in RL comes from long-term memory, it will remember which input produced the highest reward so far. This means that RL works with search and long-term memory, where the results of the search are stored so that future searching takes less effort. The term RL comes from the idea that if an event is followed by a positive response or improvement in the situation, it will increase the likelihood of that event occurring again. This idea comes from the field of psychology, but it is not used within that field, opting instead for different terms. Since RL does not receive training directions or examples directly, it instead has to try different approaches. It will then observe the different results from the approaches, to find better behaviour. This process can be described with many different terms, such as; trial-and-error, generate-and-test, variation and selection, blind variation and selection, and selectional. The need for different approaches lead to a conflict between exploitation of what has been learned, and exploration to learn more. RL needs to maintain a balance between these conflicts.

A different approach to RL is Curriculum Reinforcement Learning (CRL). The education of humans is organized, based on systems and curriculums to introduce concepts at different times, to ease the learning process by exploiting previously learned concepts. CRL works similarly to guide the learning. According to *Bengio et al.* [14] The basic idea of CRL is to start small, learn easier aspects or a task or easier sub-

tasks, to then gradually increase the difficulty. CRL also allows the increase to be based on different attributes, such as the reward. Allowing it to change the difficulty level if the reward or mean reward is high enough. It is also possible to decide a minimum amount of lessons so that the difficulty doesn't increase too early because of a lucky instance.

With the existence of CRL, one can wonder if it is more efficient than RL. Designing a curriculum could be a possible way to save time on training or they could be equally efficient in learning. Since curriculum are used in human education to make the learning process more efficient as well as easier. It might also apply to AI learning, which is our motivation for picking this gap for our research. In this project, we will compare CRL with RL to see which of these is more efficient in learning.

1.2 ML-Agents

ML-Agents toolkit [8] is a package developed by the Unity team for training Machine Learning agents. The package includes a Python API that communicates with the Unity engine and allows for a simulation when training. The ML-Agent package includes documentation and step by step examples on how to set up the API. By default, the API uses Proximal Policy Optimization (PPO) [12], an RL technique that is implemented in TensorFlow [1][2] which runs in Python. PPO uses the Agent's observation to map the best action it can take using a neural network. ML-Agents toolkit contains example templates of environments that are already trained with Machine Learning algorithms. Like 3D Balance Ball, where the goal is for the agent to balance the ball on its head. Push Block, a platform environment where the Agent needs to push a block to the goal. Tennis, A two-player game where the agent controls a racket and its goal is to hit a ball over the net. The example templates include documentation that teaches the basics of API and step by step instructions on how to its implemented. TensorFlow has a feature called Tensorboard where you can observe the training statistics.

1.3 Aim & Objectives

We will be comparing instances of learning performed with RL and CRL to try and discover any significant differences. We used the Unity Engine and the ML-Agents toolkit to create an environment in which an Agent will try to get to a goal while avoiding enemies and hindrances. The question we aim to answer is; "Is curriculum reinforcement learning more efficient than reinforcement learning?"

The hypotheses we had for this project were; H0: "There is no significant difference between curriculum reinforcement learning and reinforcement learning" H1: "Curriculum reinforcement learning is more efficient."

According to Guy Bruce (Evidence-Based Educational Methods, 2004) [10] learning efficiency is judged by performance improvement and learning time. His work

is relating not to Machine Learning, but relating to human education, but we can use his definition for Machine Learning learning efficiency. We will measure how efficient RL is in comparison to CRL by training a few instances of each. Then we will observe the statistics of the training with Tensorboard.

This thesis aims to evaluate our research question if curriculum reinforcement learning is more efficient than reinforcement learning.

1.4 The structure of the thesis

This thesis is split into different parts. *Related Work* gives an overview, introduces and describes some related work on AI trained with RL and we will mention how they are related to this work. In *Method* we go through the methods we used for the thesis and how we evaluate the results. In *Results*, we will present the results of our experimentation. *Analysis and Discussion* is where we analyse and discuss the results of the experimentation. *Conclusion and Future Work* is where we end with the conclusion from our work and what similar work could be done in the future.

There is a magnitude of previous work within AI, Machine Learning, and even within RL specifically. OpenAI and DeepMind have a lot of work involving RL used to train Agents to play and master different games. Which relates to our experimentation through experimenting on RL using a game environment. Training competing agents at the same time produce an arms race effect which is similar to CRL.

OpenAI Hide-and-Seek [9] involved two teams of two Agents competing in a game of hide-and-seek. One team was tasked with hiding, while the other team were the seekers. The seeking Agents had an angle in front of them which was their line of sight and would chase the hiding Agents if they spotted them. The environment had objects that the Agents were able to move if they were in front of the Agent, and the Agents were also able to lock the objects, preventing them from being moved. The tactics that were employed kept evolving depending on what the other team did. Some examples include the seeking team learning to use the ramp object when the hiding team blocked the entrances and the hiding team taking the ramp before blocking in response. There are also examples of the Agents breaking the game or choosing unimagined tactics, like using the ramp to launch themselves, throwing the ramp outside the map through the wall, or surfing on a block by standing on the edge facing inward. While we didn't see the exploitation of this kind in our experimentation, we did notice and counteract a behaviour that undermined the learning, which was when our Agent moved along the outer wall until it got to the goal, often taking multiple laps. The competitive nature of this also created a type of automatic curriculum, making it similar to using CRL.

OpenAI Five (Dota 2) [4] is an AI that was trained with RL to play the game Dota 2. It learned using a separate neural network for each different playable character and no human experience to build from. It managed to develop strategies that human players utilize and eventually managed to beat human players that spend time training at the game. This was done using the PPO algorithm developed by OpenAI, which was implemented in ML-Agents and subsequently was used in our Experimentation. Additionally, the way that OpenAI Five avoided strategy collapse was to play some games against its previous self. (80% current self, 20% old self) This together with the competitive nature simulates CRL.

DeepMind AlphaZero (Go, Chess, & Shogi) [3] this is based on the same AlphaGo Zero algorithm. It was developed to be able to learn and master the games; Chess, Shogi (Japanese chess), and Go. It got recognition from the chess community who identified its chess play-style as ground-breaking. AlphaZero was able to beat hand-crafted engines for shogi and chess, as well as beating their previous AlphaGo and AlphaGo Zero. This general approach to RL is similar to our experimentation which doesn't aim towards one specific target but the general comparison between RL and CRL.

DeepMind AlphaStar (Starcraft 2) [5] is an AI that learned how to play the game Starcraft 2. It managed to reach the top league, being ranked above 99.8% of active players. It used the same constraints that human players have, by viewing the game through a camera and having a limit on the frequency of actions. This AI started with supervised learning and went on to use RL to improve. They used more general-purpose techniques. We similarly used general purpose techniques in our experimentation to compare RL and CRL.

DeepMind FTW (Quake 3 Arena) [6] is an AI that was trained to play capture the flag, one of the first game-modes that became popular in first-person shooter games. The agents were trained as a population, learning by playing together. They are trained for teamwork and were judged as more cooperative than human players in a test tournament with a mix of human and AI players. The agents were unaware of the rules of the game and discovered them through play, similarly to how in our experiment, the Agent only knew its own forward vector and velocity and could detect other entities through cast rays.

There is general work in attacking RL. This was done by Adam Gleave et al. [7] which they demonstrated in three zero-sum games between the adversarial AI and the self-play RL taught AI. The adversarial policies managed to reliably win against the RL AI and they found that the policies were more successful in high-dimensional environments. The policies also induce different activations in the RL AI than when the RL AI played against normal opponents. The general approach is similar to our approach to comparing RL and CRL.

There is a lot of work involving Machine Learning as well as RL. Competitive games are a common area in which AI is trained to perform, which produces a similar situation to CRL where the challenge gets progressively more difficult. None of the work we've seen compares the learning efficiency between RL and CRL, or RL, and the pseudo-CRL situation of competitive games.

To answer our research question we used the research methods Implementation and Experimentation. Using the Unity Engine we will implement the environment. For the experimentation we trained the Agent using TensorFlow that is running a Python script in the Windows command console. We use the TensorFlow feature Tensorboard to observe the recorded statistics of the training session. We trained the Agent three times with RL and four times with CRL. We did CRL four times because the following two did not recreate the result of the first one, so we trained it a fourth time. This gave us three similar CRL results.

3.1 Implementation

For the experiment, we created a 3D environment. The Environment was designed purposely in a circular way as shown in figure 3.1. We created an area script that manages the environment. The purpose of the area script was to generate the environment at the start of each episode and clear it on episode end. The script generated the props inside in an area at a random position between a specified angle and range from the center. The props in the experiment consisted of a Bird, Pig, Nest, and Pillar, shown in figure 3.2.

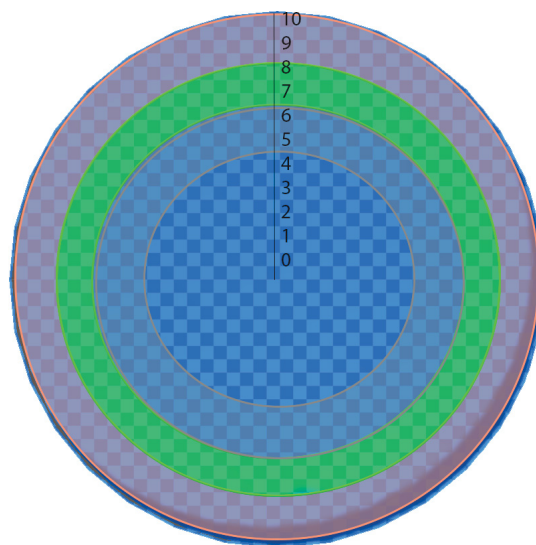


Figure 3.1: Shows a top-down view of the 3D environment. The coloured areas are where props spawn.

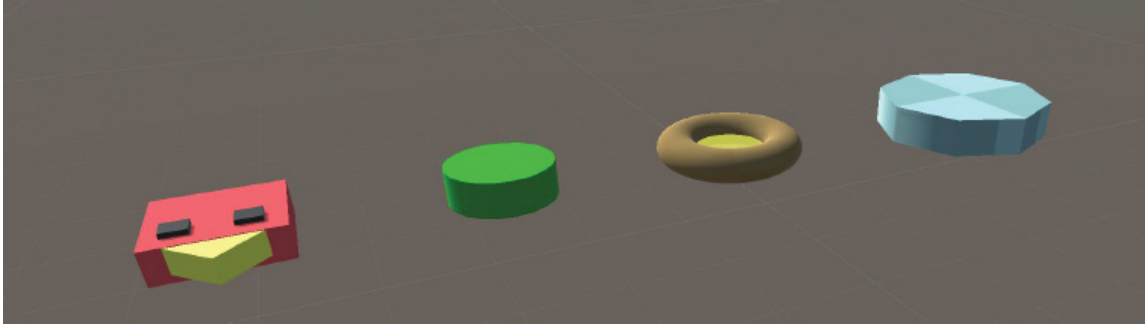


Figure 3.2: From the left to right, Bird Agent, Pig, Nest, and Pillar.

The Bird is the Agent that we were training. The Agent could move forward, backward, and was able to rotate to the left and the right. To make an informed decision of the state of the environment. The Agent collected observations of its forward vector, and velocity. It also cast one ray forward and five rays spread over 100° to each side. The rays will collide with props in the environment and they will detect and distinguish between the different types of props by their unique tags.

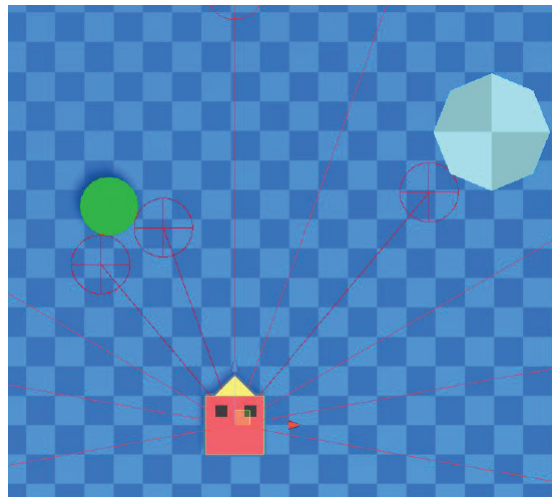


Figure 3.3: The ray sends out a sphere that uses collision for detection.

The goal for the Agent is to successfully navigate through the environment to the Nest. A penalty was given to the Agent every step, making it learn to avoid standing still or performing unnecessary movements. The Pigs are enemies that the Agent needs to avoid. When the Agent collides with a Pig it gets a significant penalty. The Pigs move towards a random position which makes it hard to predict what direction it will take next. If the Pig collides with a pillar or wall it will change and move towards the opposite direction. When the Agent reaches the Nest it gets rewarded and completes the task, ending the current episode. The Pillar is an obstacle that limits the movement area of the Agent and gives a small penalty on collision. This penalty discourages the Agent from colliding with the Pillars. The walls of the environment also cause the episode to end when the Agent collides with them. The reason why we did this is that when training we noticed a technique the Agent abused was simply

moving to and along the walls. This was an issue because it became too easy when it avoided the obstacles. The episode length we used was based on the ML-Agents examples. The episode length was 5000 steps, after that the episode ended. We had 16 instances of this environment running in parallel during the training, shown in figure 3.4, to increase the amount of learning experiences.

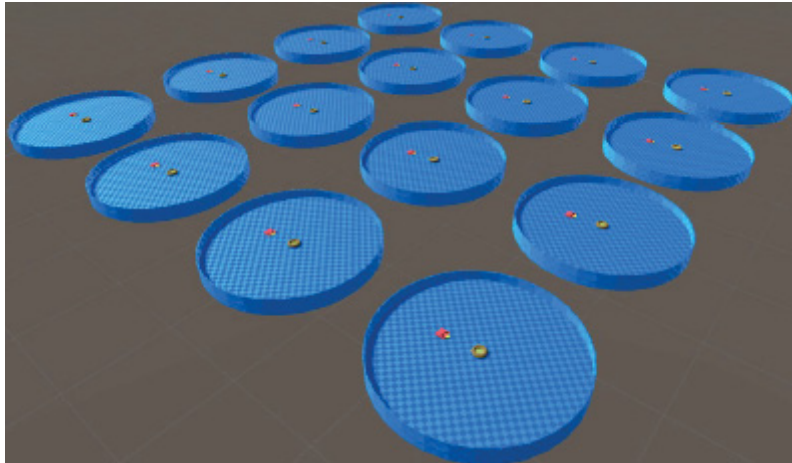


Figure 3.4: The parallel environment setup we used in our experimentation.

The figures 3.5 to 3.9 show the script values and component values used during the training.

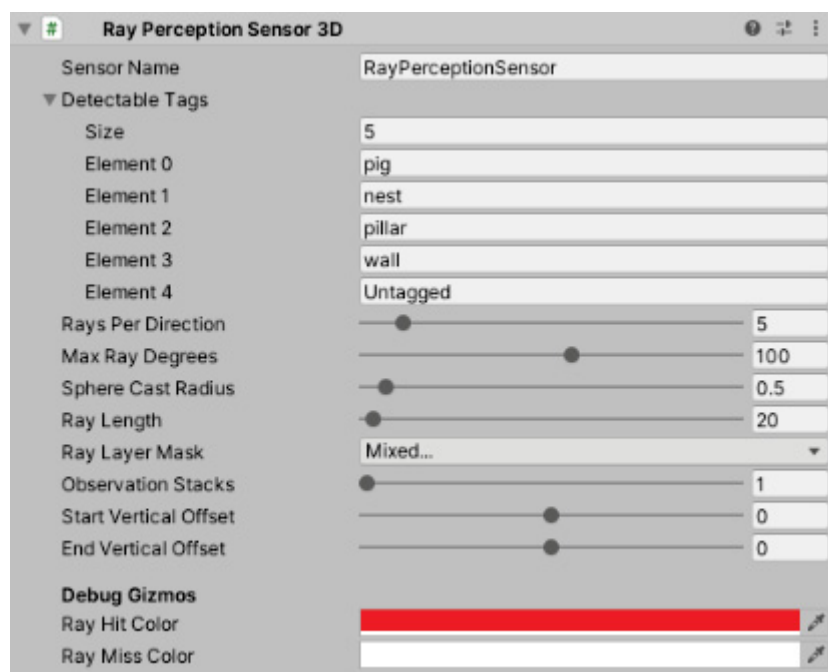


Figure 3.5: The 3D Ray Perception Sensor Component for the Agent.

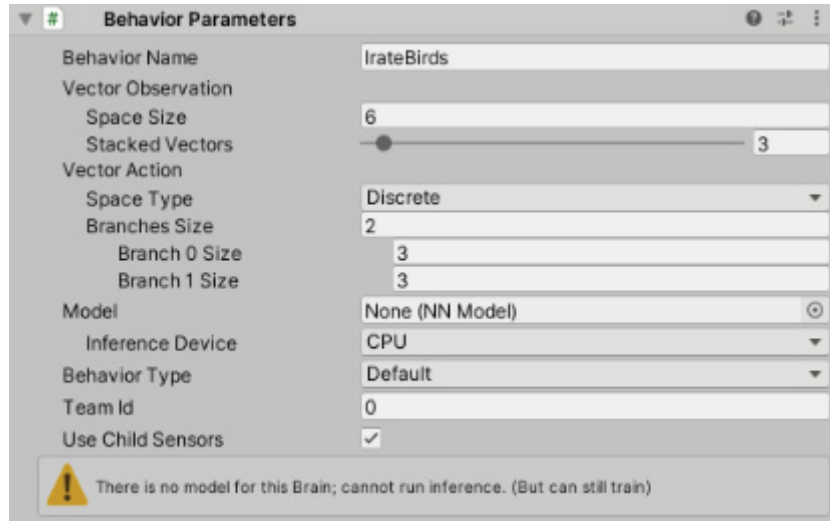


Figure 3.6: The Behaviour Parameter component for the Agent.

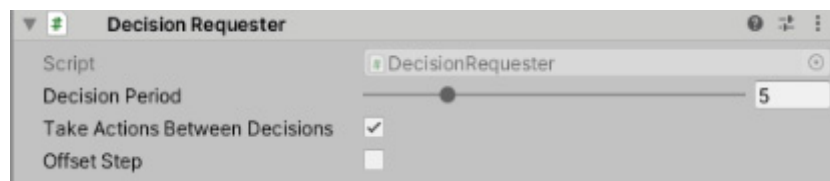


Figure 3.7: The Decision Requester Component for the Agent.

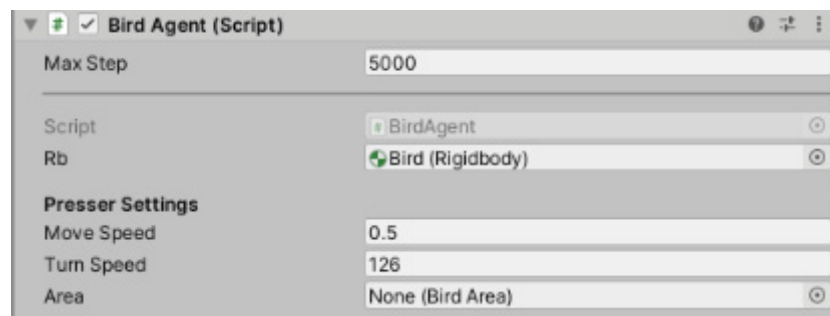


Figure 3.8: The script for the Agent.

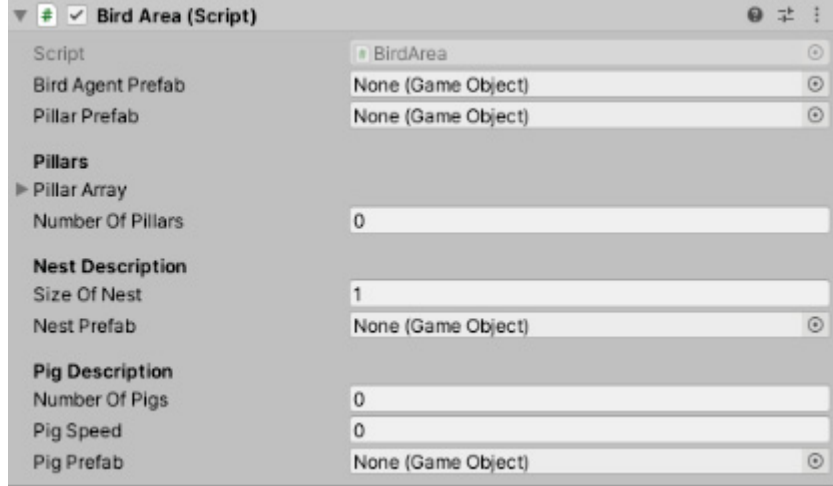


Figure 3.9: The script for the Area.

Figure 3.5 is of the 3D Ray Perception Sensor Component which was used by our Agent. It sends out spheres along rays that provide detection. It detects an object with the five listed tags; pig, nest, pillar, wall, and Untagged. (Untagged is the default unspecified tag for Unity objects) In the figure we can see the different values that dictate how the rays are set up, with the number of rays on the sides, degrees they are spread away from the middle, and the detection sphere radius. Figure 3.6 is of the Behaviour Parameters Component, it is the brain of the Agent. It is in here we can define what it can perceive and the actions it can take. What Figure 3.7 shows is the Decision Requester Component. It is used to allow the Agent to make decisions on regular intervals instead of at specific moments. Our setup had the Agent make a decision every five steps. Figure 3.8 is the Bird Agent Script Component, here is where we implement the functions for movement and rewards. We set the movement speed, turn speed, and the maximum steps for an episode. Figure 3.9 is the Script Component for the Bird Area. This script controls the environment. On episode starts it creates the props and on episode end it destroys them. We can observe the active attributes during training.

Lesson	Start	1	2	3	4	5	6	7
Threshold		0.2	0.45	0.55	0.7	0.75	0.8	0.8
# of Pigs	1.0	1.0	2.0	2.0	3.0	3.0	4.0	4.0
# of Pillars	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
Pig speed	0.0	0.5	0.5	1.0	1.0	1.5	1.5	2.0
Nest size	1.7	1.6	1.5	1.4	1.3	1.2	1.1	1.0

Table 3.1: The Curriculum behaviour

We found the best way to compare CRL with RL was that the last lesson in CRL should have the same setup in the environment as the RL. CRL lessons setup is seen in Table 3.1. By increasing the difficulty of the environment gradually we can ease the Agent into the task. Similar to how mathematics is taught in schools, starting with arithmetic then algebra, before starting with calculus.

3.2 Development

Just by looking at the mean cumulative reward and the standard deviation (std) of the reward on the Windows command console that is running the Machine Learning will not help us much. A good feature that comes with ML-Agent is that you can watch the Agent train real-time in Unity. We use this to discover any bad behaviour like the Agent moving along the wall or getting stuck between the wall and pillar. This could make it too easy or lead to std being high. We found out that std became high when we had too many Pigs or the Pigs were too fast. We did some preliminary training with RL to get an idea of what the different lesson thresholds should be for CRL. We based the length of the training on both previous tests and wanting a longer length to be able to properly discover any difference in the learning rate.

3.3 Evaluation

The ML-Agents toolkit has Python packages that run in the Windows command console. The Python package allows for training with Machine Learning and will also interact directly with the Unity environment. While training the Agent the metrics we will observe, in the console, is the mean reward and std, as shown in figure 11. The mean reward refers to the mean cumulative reward over all the Agent training in parallel. The std is a measure of the spread around the mean reward. The size of the value indicates the amount of variation in reward, the Agents received while training. High value means a large number of variations and a low value indicates the opposite. What we will be measuring is the improvement of the mean reward.

```
BirdCRLforSS: IrateBirds: Step: 630000. Time Elapsed: 900.375 s Mean Reward: 0.789. Std of Reward: 0.558. Training.
BirdCRLforSS: IrateBirds: Step: 640000. Time Elapsed: 915.410 s Mean Reward: 0.765. Std of Reward: 0.575. Training.
BirdCRLforSS: IrateBirds: Step: 650000. Time Elapsed: 930.921 s Mean Reward: 0.802. Std of Reward: 0.503. Training.
BirdCRLforSS: IrateBirds: Step: 660000. Time Elapsed: 943.895 s Mean Reward: 0.802. Std of Reward: 0.531. Training.
] IrateBirds lesson changed. Now in lesson 6: nrOf_pigs -> 4.0, nrOf_pillars -> 3.0, pig_speed -> 1.5, nest_size -> 1.1
BirdCRLforSS: IrateBirds: Step: 670000. Time Elapsed: 959.682 s Mean Reward: 0.840. Std of Reward: 0.469. Training.
BirdCRLforSS: IrateBirds: Step: 680000. Time Elapsed: 974.489 s Mean Reward: 0.617. Std of Reward: 0.702. Training.
BirdCRLforSS: IrateBirds: Step: 690000. Time Elapsed: 988.941 s Mean Reward: 0.668. Std of Reward: 0.665. Training.
BirdCRLforSS: IrateBirds: Step: 700000. Time Elapsed: 1003.591 s Mean Reward: 0.615. Std of Reward: 0.711. Training.
```

Figure 3.10: An example of the Windows command console during training.

We will be looking in the Windows command console that is running the training for the mean reward and how it improves by approaching the maximum reward, being 1 in our experiment. The std should be as low as possible so the Agents that are learning in parallel don't vary too much in reward. After the training is done we will look at TensorFlow for an overview of the training and evaluate the RL and CRL. We trained the Agent multiple times in both learning methods to obtain more stable data. Since the CRL starts with easier lessons we cannot properly compare the mean reward before they reach the same challenge as RL. When the CRL has reached the last lesson, which has the same attributes as the RL, we can properly compare their rate of learning. For example, if one mean reward goes from 0.4 to 0.8 over 1 million steps, while the other takes half as many steps, the second one would

be more efficient in learning. To validate the results we will be performing multiple instances of RL and CRL.

The results presented here are from the experimentation. We conducted seven instances of training, three with RL and four with CRL. We will first show the results from the RL training, then the CRL training, before showing a comparison between all seven training instances.

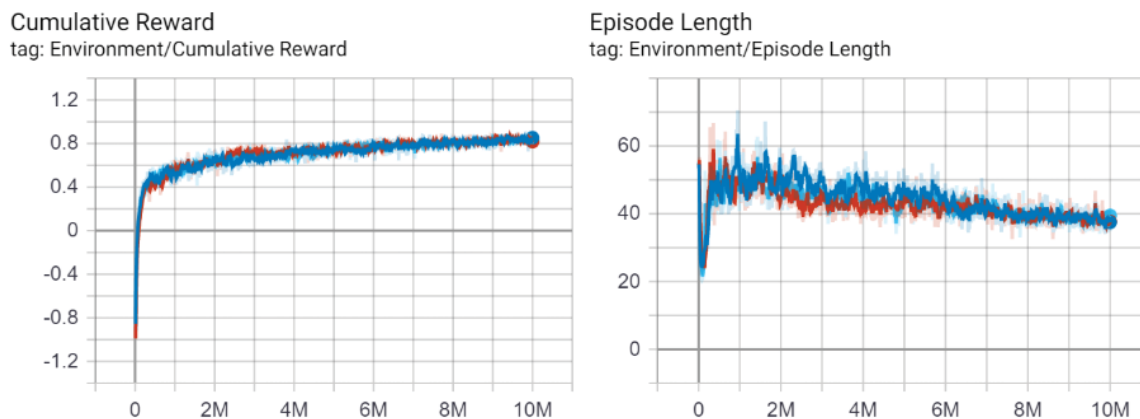


Figure 4.1: Graphs of three separate RL training statistics.

The result we got after running the three separate RL instances can be seen in figure 4.1. The left-hand graph in the figure shows the Cumulative Reward, which shows the mean reward throughout training. In the graph, we can see how the Agent improves over the length of ten million training steps. The right-hand graph in figure 4.1 is the Episode Length graph. It shows the mean episode length throughout training. It fluctuates a lot in the beginning and stabilizes toward the end.

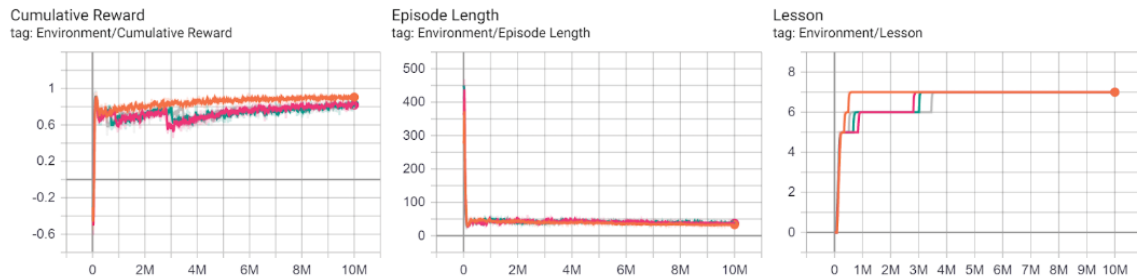


Figure 4.2: Graphs of four separate CRL training Statistics.

Figure 4.2 shows the results of the four CRL training instances. The left-hand graph shows the Cumulative Reward. The mean cumulative reward rises in the beginning before leveling out. The middle graph is of the Episode Length. It starts with long episode length and then descends into shorter length episodes. The right-hand graph is of the Lesson and shows the duration spent on a lesson. It shows the duration of the lesson in training steps. In the figure we can see the orange session being an exceptional instance of training. With higher mean cumulative reward and a shorter time spent in the lessons outside of the last.

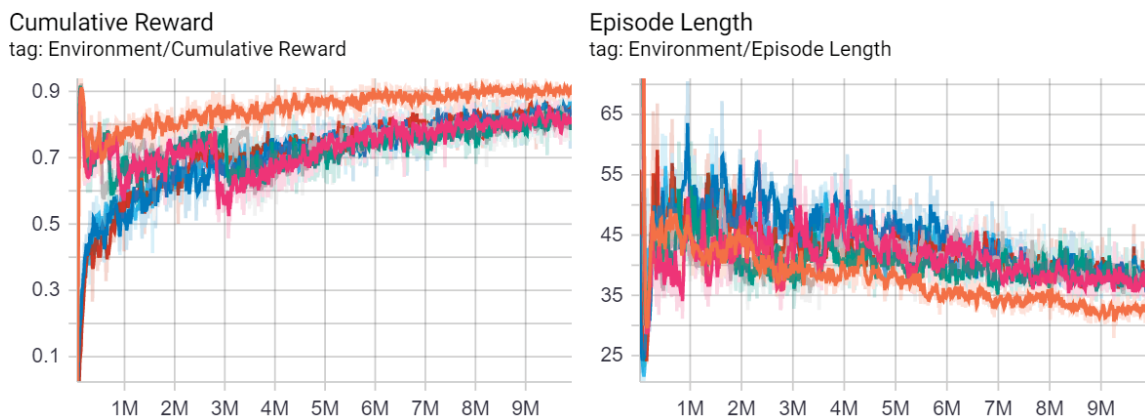


Figure 4.3: Graphs of the seven training instances.

What is shown in figure 4.3 is the results of all seven instances of training. The left-hand graph is the Cumulative Reward for both RL and CRL. In the graph it can be seen how CRL's mean reward rises above the RL early in the training. Before it drops off and aligns with RL. With the exception of the orange CRL which stays higher in mean reward than the six other instances of training. The right-hand graph shows the Episode Length of the seven instances. Both RL and CRL start out fluctuating and later become steady over time. The orange is still an exception, in that it stops fluctuating earlier in the training and ends up with a shorter episode length than the other six.

Name	Colour	Value	Training duration
CRL 1	Orange	0.9124	5h 11m 57s
CRL 2	Magenta	0.7947	4h 38m 53s
CRL 3	Green	0.8511	4h 44m 28s
CRL 4	Gray	0.8151	4h 27m 53s
RL 1	Deep blue	0.8897	4h 43m 40s
RL 2	Red	0.8174	4h 24m 34s
RL 3	Light blue	0.8294	4h 40m 48s

Table 4.1: Showing the end results of the training.

In table 2, we can see the end result of the RL and CRL training instances after ten million steps. From left to right we can see the name, colour, end value, and relative training duration. The order of four CRL and three RL was not necessarily the order we trained them in. But the number in the names does relate to their training order in relation to each other. (CRL 1 before CRL 2 for example) The colours were assigned by Tensorboard. The end value was the specific mean cumulative reward at the ten-millionth step. The relative training duration was how long the training took in real-time. The orange CRL instance stays an exception with a higher-end value and a longer training duration, even with the same amount of steps.

Chapter 5

Analysis And Discussion

We will analyze the results and discuss the possible reasons for those results. We will also discuss what the results mean in relation to our hypotheses and our research question.

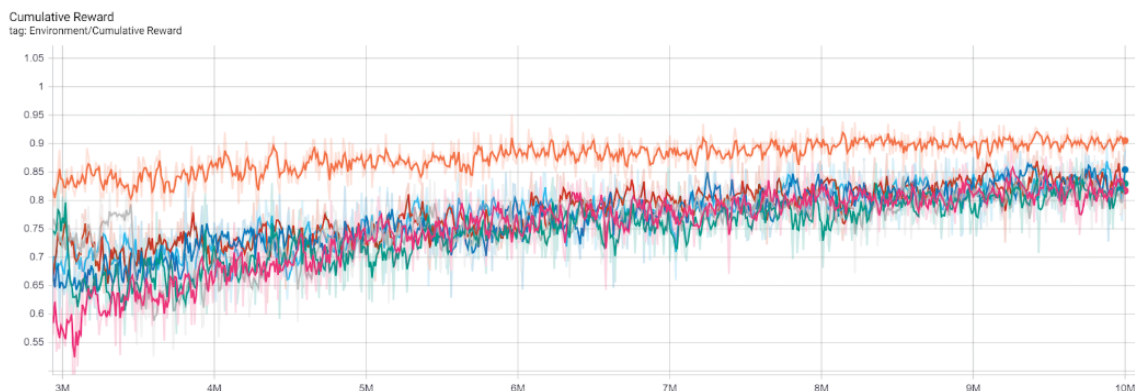


Figure 5.1: Graph of CRL and RL Cumulative Reward when CRL entered their last lesson

In figure 5.1 we can see the results of the seven training instances when the CRL instances have entered the last lesson. This lesson is when the environment in CRL has the same attributes as the RL environment. In this we can see that (with exception to the orange CRL) the mean reward of RL and CRL has aligned. The rise of mean cumulative reward at the beginning of CRL that was shown in figure 4.1 is due to the ease of the earlier lessons. This also explains the dropping that happens later, which appears to happen when they enter the more difficult lessons. The RL instead has a steady rise in mean cumulative reward since the environment stays the same and it becomes more adept at handling it. At the beginning of the training CRL appears to drop down from a high episode length, the reason for this is because the start only has one enemy which is immobile. The episode then reaches its maximum steps for an episode and ends. Since RL starts out with enemies that are moving it will be more likely to end an episode early due to collisions.

The research question we asked was “Is curriculum reinforcement learning more efficient than reinforcement learning?”. Meaning that if CRL at any time during the training was learning faster than RL. Either by reaching the same plateau earlier or being ahead in mean reward consistently at the time in learning. To do this properly both had to have the same environment attributes. This state of CRL happens in

the last lesson. By looking at the Lesson graph in figure 4.2 we can see that all except one enter its last lesson at around 3 million steps. That is when the CRL mean cumulative reward ends up aligned with the RL mean cumulative reward. The exception is the orange-coloured CRL training session. It keeps a high cumulative mean reward over its training duration. We are unsure why this happened. What can be seen in figure 5.1 is that the orange CRL session has a higher mean reward than the other CRL during their last lesson. Table 4.1 shows the results at the end of the training. It shows the mean reward value at the last step and shows the duration it took to train them. It can be noted that the orange CRL training is almost half an hour longer than the rest. This does not reveal much about this session. The Lesson graph in figure 4.3 also shows that the orange CRL instance passed into the last lesson earlier than the other CRL. This could be due to the random element of the environment and that the Agent got “lucky” and passed the lesson right after the minimum required lessons had passed.

Based on the results we got, we could not disprove the null hypothesis, H_0 . This states that “There is no significant difference between curriculum reinforcement learning and reinforcement learning.” The figure 4.3 shows that after entering the last lesson the CRL did not have a significant difference from RL in mean reward. It also shows that the episode lengths did not differ much either. In figure 5.1 we get a closer look at the mean rewards from around 3 million steps. This gives a better view of how similar the mean rewards were between RL and CRL.

To reject the null hypothesis we need to see if there is a significant difference in results. The mean reward at the last step can be seen in table 4.1. With a one-tailed t -test we found that there was no significant difference in results, $t(7) = -0.0607$, $p = .4769$, with CRL ($M = 0.8433$, $SD = 0.001$) rising faster before aligning with RL. ($M = 0.8455$, $SD = 0.001$) Because $p < .05$ that means the null hypothesis could not be rejected.

Our other hypothesis was that “Curriculum reinforcement learning is more efficient.” This hypothesis is not supported by the results. The results showed that when the CRL reached the same difficulty in the environment as RL, they aligned in their mean cumulative reward. Meaning there was no significant difference in their learning efficiency. There was no significant difference in mean rewards at the last step or the training time. The null hypothesis was not disproven either. All of this meant that our hypothesis was wrong.

A possible explanation why CRL was not more efficient than RL could be that the environment we trained it in was too simple. There might be a learning efficiency difference with a more complex scenario. Another possible reason is a lack of mechanical complexity. If our environment had more complex game mechanics that might have required training each mechanic separately in CRL, there might have been a significant difference. Theoretically, if we made a different environment with hazards that could be deactivated by situational mechanics to progress in the level. In this instance we could use CRL to train individual mechanics. This might give higher efficiency in learning compared to RL which would train in the level with all features from the start.

Chapter 6

Conclusions and Future Work

This thesis aimed to discover if CRL is more efficient than RL. We implemented an environment in the Unity Engine using the ML-Agents toolkit to perform our experiments. These included training Agents in both CRL and RL. To evaluate the experiments we looked at the mean cumulative reward over the training steps to see if CRL was more learning efficient. The results we got after training seven instances of experimentation, four, CRL and three RL. What these showed was that at the last lesson CRL aligned with the results of the RL. Analyzing the results showed that there was no significant difference in the learning efficiency between CRL and RL because the result of our t-test did not show significance. This could be because our implementation was too simple for a difference to appear. Since there was no significant difference our null hypothesis was not disproven, and the results together with this did not prove our hypothesis. Through the hypotheses and the results, we got the answer to our research question.

The answer to the research question “Is curriculum reinforcement learning more efficient than reinforcement learning?”, in our case, it was no, CRL was not more efficient. We did not manage to disprove the null hypothesis.

Our experiment consisted of testing one Agent in one task with both CRL and RL. This narrow focus leaves room for future work comparing CRL and RL. One type of future work that would be possible is conducting similar experiments on Agents performing a variety of tasks. This would allow the experimenter to observe the differences or similarities between the results of the different tasks. Another type of future work is experiments on more complicated tasks to see if any differences become significant. It would also be possible to have more complex game mechanics. In which RL trains on the whole while CRL trains with individual mechanics before training with all of them. As the experiments we conducted had only one Agent that was learning during the training. This means that future work involving multiple agents is possible. These multiple Agents could conduct the tasks in cooperation or competing against each other. Experiments in which the Agents compete against each other you could have CRL trained Agents compete against RL trained Agents. In such an experiment the difference in quality from the training could be assessed and analyzed. The length of the training is another area in which one could conduct future work. This could involve experimenting with CRL and RL to see when the two approaches align in their results, misalign, or if they align at all. There is also the possibility of testing CRL as a means to set up different levels of difficulty in AI

opponents that would compete against human players.

References

- [1] Tensorflow: A system for large-scale machine learning. 11 2016.
- [2] Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 11 2016.
- [3] Mastering chess and shogi by self-play with a general reinforcement learning algorithm. 12 2017.
- [4] Dota 2 with large scale deep reinforcement learning. 12 2019.
- [5] Grandmaster level in starcraft ii using multi-agent reinforcement learning. pages 350–354, 10 2019.
- [6] Human-level performance in 3d multiplayer games with population-based reinforcement learning. 364, 5 2019.
- [7] Cody Wild Neel Kant Sergey Levine Stuart Russell. Adam Gleave, Michael Dennis. Adversarial policies: Attacking deep reinforcement learning. 2 2020.
- [8] Esh Vckay Yuan Gao Hunter Henry Marwan Mattar Danny Lange. Arthur Juliani, Vincent-Pierre Berges. Unity: A general platform for intelligent agents. 9 2018.
- [9] Todor Markov Yi Wu Glenn Powell Bob McGrew Igor Mordatch. Bowen Baker, Ingmar Kanitscheider. *Emergent Tool Use From Multi-Agent Autocurricula*. 2 2020.
- [10] Guy S. Bruce. Evidence-Based Educational Methods, chapter Chapter 15 - Learning Efficiency Goes to College. Educational Psychology. Elsevier Academic Press, 2004.
- [11] Warren B. Powell Donald C. Wunsch Thomas G. Dietterich Jennie Si, Andrew G. Barto. Reinforcement Learning and its Relationship to Supervised Learning, chapter Chapter 2. John Wiley & Sons Inc.
- [12] Prafulla Dhariwal Alec Radford Oleg Klimov. John Schulman, Filip Wolski. Proximal policy optimization algorithms. 8 2017.
- [13] Karmel Allison Laurence Moroney. *Machine Learning Zero to Hero(Google I/O'19)*.

- [14] Ronan Collobert Jason Weston Yoshua Bengio, Jérôme Louradour. *Curriculum Learning*. Proceedings of the 26th International Conference on Machine Learning, 2009.

