Reinforcement Learning for Tower Defence

Andrew Wallace
101210291
andrewwallace3@cmail.carleton.ca

Derrick Zhang
101232374
derrickzhang@cmail.carleton.ca

Mohammad Rehman
101220514
mohammadrehman@cmail.carleton.ca

Manal Hassan
101263813
manalhassa@cmail.carleton.ca

Abstract

The abstract paragraph should be indented ½ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Introduction

1.1 Project focus

Tower defence is a strategy game where the goal is to place defensive structures to obstruct and defeat waves of enemies from reaching a goal. This is done by placing and upgrading towers along the attacker's path. The towers are stationary objects that the agent places to attack enemies. These towers have a cost to create and upgrade, meaning the agent must gather resources to do so. In traditional tower defence games, the player starts with a set number of resources and can obtain more after successfully defending from a wave or defeating enemies. The defensive tower upgrades can vary in range and damage.

The objective of this project is to train an agent using reinforcement learning (RL) methods to learn an optimal policy for tower placement that maximizes survival and minimizes damage taken from enemies. Unlike traditional tower defence games with manual upgrades, our environment features towers that automatically evolve and strengthen based on the number of enemies they eliminate, creating a feedback loop between decisions and long-term strategy when it comes to positioning.

1.2 Importance of the problem

Exploring RL for tower defence is extremely valuable. Applying RL methods to high-dimensional state and action spaces, challenging reward structures, and balancing short- and long-term investment provides valuable insights into RL's capabilities in highly transferable domains.

Preprint. Under review.

1.3 MDP specifications

To Be added

2 Approaches

2.1 Prior work

Previous work by Maria Manolaki also focused on implementing reinforcement learning (RL) for tower defence for learning an optimal policy [1]. However, in their implementation, the enemy was the artificial intelligence agent. The agent's goal was to learn an optimal policy to win the game. This involves the enemy avoiding towers and finding the best path to the base. They implemented the A2C algorithm to provide the enemy with the intelligence needed to reach the base. Our implementation differs, focusing only on the player agent. It would be interesting to build on this by deploying a multi-agent interaction between a player and an enemy agent and analysing their interaction.

Another application of RL for tower defence was implemented by Timothée Blondiaux et al. for Plants vs. Zombies, a popular tower defence game consisting of plants as towers and zombies as enemies [2]. They implemented a wide array of RL methods, including A2C, Monte-Carlo policy gradient, Deep Q-Networks (DQN), and Double Deep Q-Networks (DDQN). They observed good results with the DDQN algorithm, beating difficult levels 65% of the time [2].

A more general approach was taken in a study by Abhinav Wasukar and Sumitra Jakhete, looking at the application of the DQN and PPO algorithms in various game environments, including tower defence. They found that PPO significantly outperformed DQN, achieving a much higher average return. However, DQN showed lower variance and was more consistent across runs [3].

One interesting (and indirect) study done by Baylor Wetzel studied Transfer Learning in a tower defence environment. Transfer learning is the process of using knowledge gained while solving one problem to solve a new, previously unencountered problem [4]. In this study, they collected data from human-reported solutions to a tower defence game. They used these solutions to create computational agents to model human strategies. Then, a domain expert was asked to tell apart the human-made tower placements from the computer-generated ones - they could not tell the difference [4]. This approach focuses on modelling human behaviour rather than achieving optimal performance.

A similar study by Augusto Dias et al. utilised game metadata as inputs to a Deep Q-Learning agent rather than video frames or pixel information [5]. They found that no other work reported developing autonomous agents to play tower defence games (2020) and that using metadata as input reduced computational complexity [5]. This is similar to our implementation, where the state is represented by the game metadata.

2.2 Our approaches

Our project differs from these works through an automatic tower level-up mechanism and a unique approach to action and state space reduction. In our environment, towers level up automatically based on the number of enemies they kill without explicit user input. This adds a layer of complexity for learning optimal solutions. That is, balancing strategic placement for tower level-ups and successful long-term defence.

The tower defence environment has a high-dimensional and continuous state, with a large discrete action space. This makes tabular-based learning methods infeasible. Conventional methods, such as Tabular Q-Learning or SARSA, cannot store Q-tables for an infinite space. So, we first chose SARSA with function approximation to handle the high-dimensional state space. Also, SARSA is an on-policy algorithm that provides sufficient coverage of the large action space. Note that linear function approximation struggles with the large state space, so a Q-network was used. Next, we looked at the Advantage Actor-Critic Algorithm (A2C), which handles continuous state spaces and discrete action spaces [6]. Actor-Critic algorithms utilise policy gradient to update the policy directly with learned parameters. This allows us to tune the policy to reduce variance, demonstrated by methods such as SARSA (e.g., reducing actor step size).

However, SARSA and A2C are on-policy algorithms, making convergence to a stable optimal policy difficult. We then explored Deep Q-Learning with Q-Networks to avoid the on-policy nature of SARSA, and utilized a replay buffer to reduce variance.

// TODO talk PPO.

3 Empirical studies

3.1 Q-learning

We implemented Q-learning using a simple 3-layer Q-network as our function approximator and a target network to stabilize updates. The Q-network allowed us to learn in a high dimensional state and action space where tabular Q-learning would be infeasible. The space complexity for tabular learning would be $O(|S|x|A|)$. Specifically, $O(11^{v(n)})$, where $v(n) = 49$ valid squares for $n = 10$ (10×10 grid).

The target network was implemented to reduce variance and stabilize learning across episodes. However, this approach did not yield good results. As seen in Figure 1, no convergence and instability in learning.

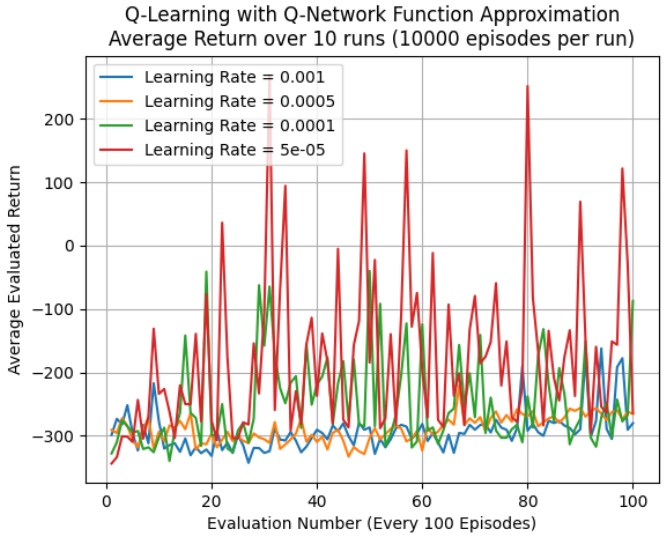

Figure 1: Q-learning performance with different learning rates.

This is due to the deadly triad combination of function approximation, bootstrapping, and off-policy learning. Even when attempting to stabilize bootstrapping updates, there was no consistency in learning. One interesting observation was that higher learning rates yielded more consistent, but lower returns. On the other hand, lower learning rates showed high variance and less consistency. With a lower learning rate, the immediate rewards propagate slowly through the Q-network, so stochasticity in the environment dominates. This demonstrates Q-learning's sensitivity to hyperparameters and overall instability.

3.1.1 SARSA

After implementing the SARSA algorithm with Q-network function approximation, we evaluated and observed the following results in Figure 2.

First, the environment has a delayed and sparse reward structure, causing SARSA's on-policy TD(0) update to struggle with delayed reward signals. Also, as wave difficulty increases, an action taken in one state may signal a vastly different reward in another state. For example, in an early wave when enemy health is lower, placing a tower in a cell that results in the elimination of the enemy will return a large positive return. However, in later waves, when enemy health is high, placing the

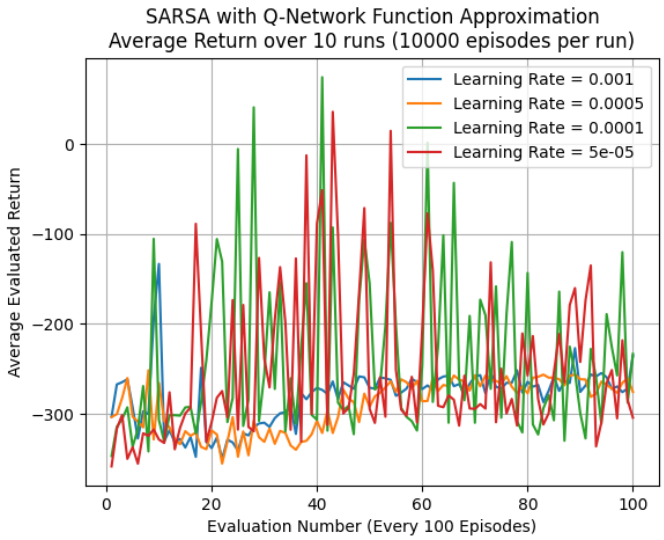

Figure 2: SARSA performance with different learning rates.

same tower in the same cell may not yield the same result. This inhibits SARSA learning, since it expects environment dynamics to be stationary to guarantee convergence. Furthermore, SARSA is an on-policy algorithm. It depends directly on actions taken. In early exploration, actions are random, which leads to poor estimates, noisy updates, and high variance.

Due to the on-policy nature of SARSA, we do see more coverage of the action space compared to off-policy methods such as Q-learning. Overall, SARSA performed poorly in learning an optimal policy. Reducing the state dimensionality and ensuring stationary environment dynamics are required for better performance. Other methods should be considered before implementing SARSA in a tower defense environment.

3.1.2 Advantage Actor-Critic (A2C)

After running the A2C algorithm, it was observed in Figure 3 to have rough convergence to an optimal policy.

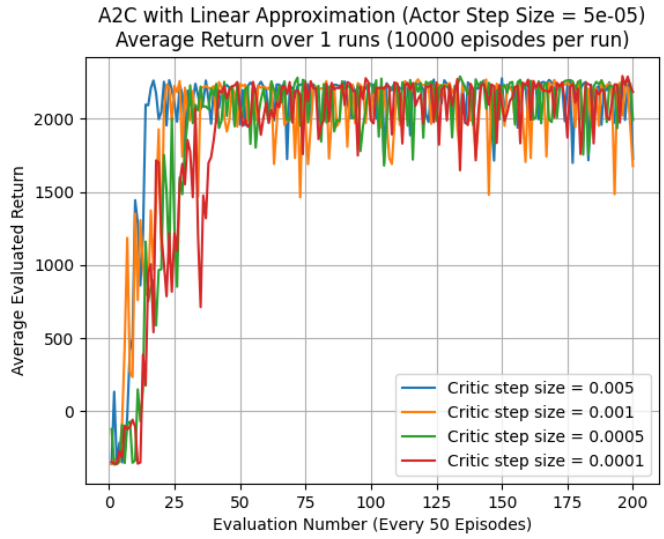

Figure 3: A2C performance with different learning rates.

In this reward structure, the estimated maximum value you could reach when interacting with the environment was 3700. In Figure 3, the algorithm performed well in achieving close to maximum reward values across runs. However, we do see high variance regardless of the critic's step size. This is due to the on-policy nature of the algorithm, where updates to the actor policy impact the policy for all other actions. Another reason is that A2C uses a SoftMax policy. The action is sampled from a distribution, so there's always a chance that the policy will deviate from the optimal action, creating variance in the average evaluated return.

We also noticed the training time for A2C was considerably longer compared to Q-learning and SARSA.

Overall, the A2C algorithm performed well with this environment, likely due to the critic's ability to make less accurate state-value estimates without drastically impacting the actor policy.

3.1.3 PPO

Proximal Policy Optimization is a type of policy gradient method, particularly helpful for complicated or large environments. Our tower defence environment has a very large discrete action space so this algorithm is suitable as it allows for efficient state exploration through stochastic sampling rather than `argmax` selection over Q-values [7]. We implemented this algorithm using the Stable Baseline library. In our experiments we tested two key parameters, learning rates and clip ranges. In the first experiment we tested 3 different clip ranges with a fixed learning rate of 0.0003. Our experiment was set up to run for 1000 episodes, evaluating every 15th episode, and a single run per clip. From this experiment we observed in Figure 4 how different clip ranges led to different learnings.

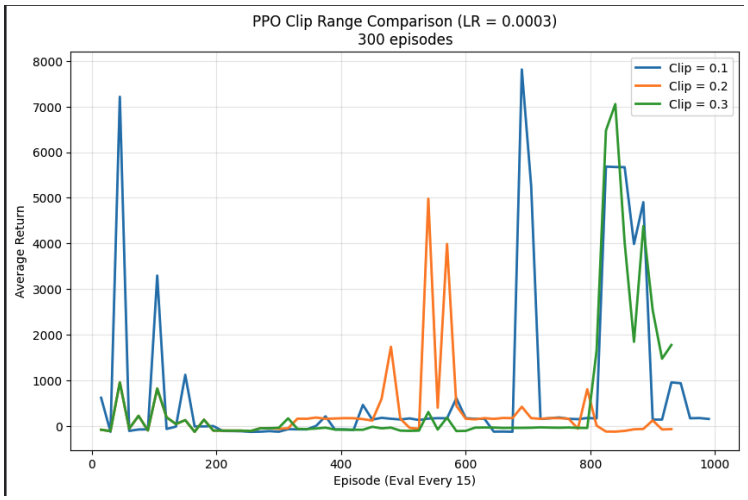

Figure 4: A2C performance with different learning rates.

With the smallest clip range of 0.1, we can see that it had the earliest return of about 7,200 with another huge peak at around 750th episode. Although it had a long period of near 0 return between the two spikes it achieved the highest single evaluation return. The next clip range of 0.2 showed a slower learning compared to 0.1. With this clip range we see that it had low peaks through learning episodes but had a major peak at around mid max episodes. Showing an overall more learning consistency than 0.1 clip. Lastly, clip 0.3 showed very slow learning performance in the beginning of the episodes as it hovered between 0 to less than 1000 return for the initial episodes. It has a major return peak near 800 episodes and showing a maintaining return between 1,500-3,000. This shows that 0.3 had very slow learning in the beginning but there is visible improvement in learning and high returns as there are more episodes. Overall, smaller clips show a faster initial return but less stable, and higher clips learn slowly but show improvement with more learning.

4 Conclusion

In conclusion, we explored various reinforcement learning (RL) methods to learn an optimal policy for tower defense. This included Q-learning, SARSA, Advantage Actor-Critic (A2C), Proximal Policy Optimization (PPO), and Deep Q-Networks (DQN).

We observed high variability and poor performance for the Q-learning and SARSA algorithms. This is due to the difficulty of bootstrapping in high-dimensional state spaces with sparse and delayed rewards. Policy gradient-based algorithms such as A2C and PPO yielded better results. This is due to their ability to learn the policy directly, instead of relying on bootstrapped Q-values. Also, these algorithms are designed for high-dimensional continuous state spaces while maintaining sufficient action space coverage [8]. However, they still struggle with the sparse reward system and require many episodes to converge, making training computationally expensive.

// TODO talk more about PPO and DQN?

In future work, more analysis and design of a strong feature extractor should be considered. This would help reduce the high-dimensional state complexity and improve algorithm performance. Next, other algorithms such as Double Deep Q-Networks (DDQN) and Asynchronous Advantage Actor-Critic (A3C) should be considered to stabilise learning and handle the high-dimensional state space.

Overall, we gained valuable insights into the application of RL in a tower defence environment.

5 Citations, figures, tables, references

These instructions apply to everyone.

5.1 Citations within the text

The `natbib` package will be loaded for you by default. Citations may be author/year or numeric, as long as you maintain internal consistency. As to the format of the references themselves, any style is acceptable as long as it is used consistently.

The documentation for `natbib` may be found at

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

Hasselmo, et al. (1995) investigated…

If you wish to load the `natbib` package with options, you may add the following before loading the `neurips_2023` package:

```
\PassOptionsToPackage{options}{natbib}
```

If `natbib` clashes with another package you load, you can add the optional argument `nonatbib` when loading the style file:

```
\usepackage[nonatbib]{neurips_2023}
```

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]," not "In our previous work [4]." If you cite your other papers that are not widely available (e.g., a journal paper under review), use anonymous author names in the citation, e.g., an author of the form "A. Anonymous" and include a copy of the anonymized paper in the supplementary material.

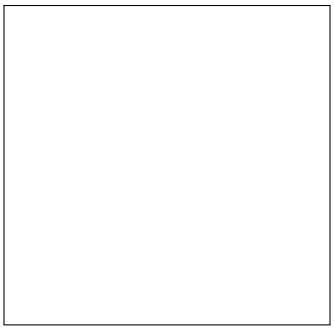

Figure 5: Sample figure caption.

Table 1: Sample table title

	Part	
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

5.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[1] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).

Note that footnotes are properly typeset *after* punctuation marks.[2]

5.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

5.4 Tables

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

5.5 Math

Note that display math in bare TeX commands will not create correct line numbers for submission. Please use LaTeX (or AMSTeX) commands for unnumbered display math. (You really shouldn't be using $$ anyway; see `https://tex.stackexchange.com/questions/503/why-is-preferable-to` and `https://tex.stackexchange.com/questions/40492/what-are-the-differences-between-align-equation-and-displaymath` for more information.)

[1] Sample of the first footnote.

[2] As in this example.

5.6 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

6 Preparing PDF files

Please prepare submission files with paper size "US Letter," and not, for example, "A4."

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You should directly generate PDF files using `pdflatex`.
- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- `xfig` "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
- The `\bbold` package almost always uses bitmap fonts. You should use the equivalent AMS Fonts:

  ```
  \usepackage{amsfonts}
  ```

  followed by, e.g., `\mathbb{R}`, `\mathbb{N}`, or `\mathbb{C}` for $\mathbb{R}$, $\mathbb{N}$ or $\mathbb{C}$. You can also use the following workaround for reals, natural and complex:

  ```
  \newcommand{\RR}{I\!\!R} %real numbers
  \newcommand{\Nat}{I\!\!N} %natural numbers
  \newcommand{\CC}{I\!\!\!\!C} %complex numbers
  ```

  Note that `amsfonts` is automatically loaded by the `amssymb` package.

If your file contains type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

6.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the `graphicx` package. Always specify the figure width as a multiple of the line width as in the example below:

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

See Section 4.4 in the graphics bundle documentation (`http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command when necessary.

Acknowledgments and Disclosure of Funding

Use unnumbered first level headings for the acknowledgments. All acknowledgments go at the end of the paper before the list of references. Moreover, you are required to declare funding (financial activities supporting the submitted work) and competing interests (related financial activities outside the submitted work). More information about this disclosure can be found at: `https://neurips.cc/Conferences/2023/PaperInformation/FundingDisclosure`.

Do **not** include this section in the anonymized submission, only in the final paper. You can use the `ack` environment provided in the style file to autmoatically hide this section in the anonymized submission.

7 Supplementary Material

Authors may wish to optionally include extra information (complete proofs, additional experiments and plots) in the appendix. All such materials should be part of the supplemental material (submitted separately) and should NOT be included in the main submission.

References

[1] M. Manolaki. Development of a tower defense game with reinforcement learning agents. Technical report, Online report, November 2020.

[2] T. Blondiaux, L. Frank, H. Gebran, and A. Perot. Plants vs. zombies: Reinforcement learning to a tower defense game, 2025. URL https://hanadyg.github.io/portfolio/report/INF581_report.pdf.

[3] A. Wasukar and S. Jakhete. Comparative study of reinforcement learning methods: Dqn vs ppo in a game, December 2024. URL https://pijet.org/papers/volume-2%20issue-1/Final%20Revised%20Paper_Pijet-07_Done.pdf.

[4] B. Wetzel. Transfer learning in spatial reasoning puzzles. In *Proceedings of IJCAI 2011*, 2011. URL https://www.ijcai.org/Proceedings/11/Papers/504.pdf.

[5] A. Dias, J. Foleiss, and R. P. Lopes. Reinforcement learning in tower defense. In *Communications in Computer and Information Science*, pages 127–139. 2022. doi: 10.1007/978-3-030-95305-8_10. URL https://doi.org/10.1007/978-3-030-95305-8_10.

[6] Actor-critic methods — mastering reinforcement learning, 2023. URL https://gibberblot.github.io/rl-notes/single-agent/actor-critic.html.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018. URL http://incompleteideas.net/book/RLbook2020.pdf.

[8] L. Weng. Policy gradient algorithms, April 2018. URL https://lilianweng.github.io/posts/2018-04-08-policy-gradient/.