Deep Reinforcement Learning for Tank-Based Warfare

Irish Senthilkumar (16342613)

School of Computer Science

National University of Ireland, Galway

Supervisors

Dr. Frank Glavin

In partial fulfillment of the requirements for the degree of

MSc in Computer Science (Artificial Intelligence)

August 23, 2021

DECLARATION I, Irish Senthilkumar, do hereby declare that this thesis entitled Deep Reinforcement Learning for Tank-Based Warfare is a bonafide record of research work done by me for the award of MSc in Computer Science (Artificial Intelligence) from National University of Ireland, Galway. It has not been previously submitted, in part or whole, to any university or institution for any degree, diploma, or other qualification.

Signature: _____

Abstract

Since the dawn of reinforcement learning, the development of intelligent agents which can play video games at a comparable level to professional human players has been a growing area of research. Current video games are much more challenging than simple Atari games of the past, as they contain high dimensional continuous environment and action spaces, which pose insurmountable challenges to traditional reinforcement learning approaches which rely on discretised low dimensional environment and action spaces. However, recent research has proved that the combination of deep neural networks with reinforcement learning can help overcome these issues. In this project, a challenging tank-based warfare game, World of Tanks, is recreated inside the Unity game engine. Cutting-edge techniques such as deep-Q learning and massively parallel training are used to create an intelligent agent which displays vastly superior performance to the finite-state machine approaches used currently.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last decade, there has been increasing interest surrounding the field of artificial intelligence, brought on by advances in computer technology and availability. Reinforcement learning is one of the most exciting fields of artificial intelligence, as it closely relates to the mechanisms which humans use to complete tasks. Video games is an exciting application of reinforcement learning, where reinforcement learning is combined with game theory to produce interesting gameplay tactics that may be similar or different to professional human tactics. This is also a task where the performance of the algorithm can be directly compared against a human player.

Recent advances in artificial intelligence allow for the combination of reinforcement learning with advanced techniques to develop advanced intelligent agents which boast exceptional performance in complex video games such as Go. World of Tanks is another complex video game that contains continuous state and action spaces, where an intelligent agent will need to use cutting-edge techniques such as deep-Q learning and convolutional neural networks. The different types of tanks and environments in World of Tanks pose a unique design challenge where agents following different policies compete against each other in team-based battles.

Chapter 2

Background

2.1 Introduction

Armoured fighting vehicles have a diverse history. They have been used across the world for hundreds of years in various forms, ranging from antiquated chariots and siege machines to modern armoured cars and artillery. In WW1, the British Army introduced a new class of armoured vehicle that would shape the future of warfare - the armoured tank. A tank is a large armoured vehicle with heavy firepower, and is designed to be used as a primary offensive weapon in front-line ground combat. Over time, the combat capabilities of tanks have evolved as advances in technology such as reactive armour and depleted uranium ammunition have brought about changes in both survivability and firepower.

2.2 Tank Warfare

2.2.1 Combat Characteristics

Armour and firepower are the two main factors which affect the combat capability of tanks. Tanks typically have a turret that can rotate 360 degrees and a gun with

fixed elevation and depression angles, as shown in Figure 2.1. The gun depression angle is important to combat capability since higher gun depression allows the use of hull-down tactics with undulations in the terrain. Hull-down tactics, as shown in Figure 2.2, reduce the profile of the tank since only the turret needs to be exposed, therefore increasing both survivability and stealth.

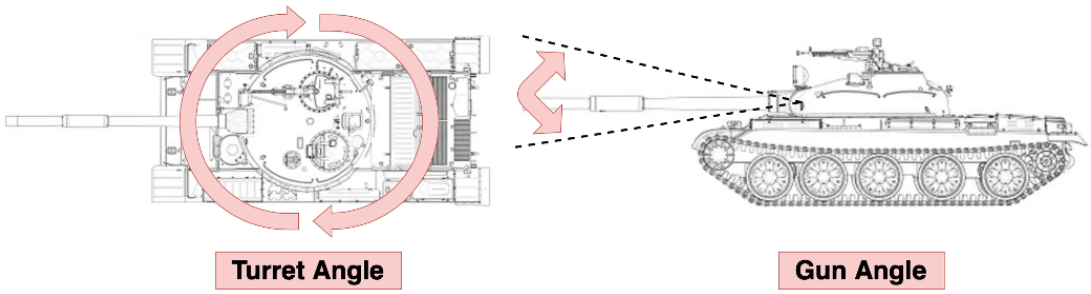

Figure 2.1: The turret rotation and gun angles are key factors that affect the combat capabilities of a tank (Malginov)

.

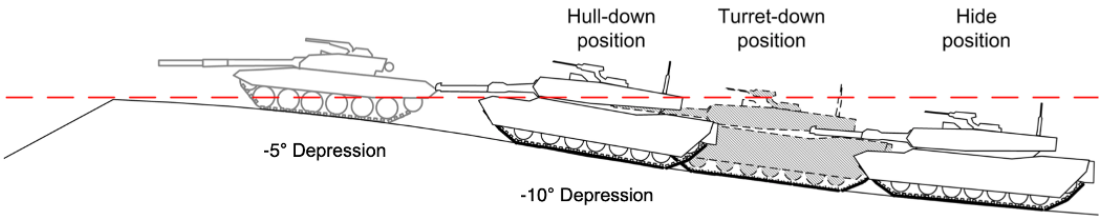

Figure 2.2: A tank with 10 degrees of gun depression can make greater use of hull down tactics than a tank with only 5 degrees of gun depression. Lower gun depression increases the exposure against enemy fire (Megapixie).

Armour is the single most important factor that affects survivability. Tanks typically have thick frontal hull and turret armour, moderate side armour, and weak rear armour. Naturally, thicker armour increases the resilience of a tank against enemy fire. The thickness of an armour plate depends on both its raw thickness and its angle. Angled armour is much more effective than flat armour since the shell will need to travel through more armour for penetration to occur,

as shown in Figure 2.3.

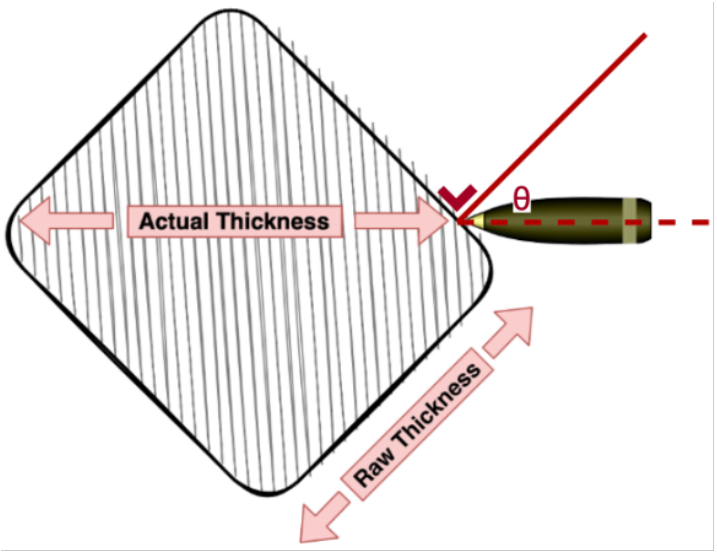

Figure 2.3: Increasing θ increases the armour angle and therefore the relative thickness of the armour plate. Angled armour is more resilient against incoming shells.

Angled armour is typically placed on the front of the tank, and is therefore only effective when shells impact the tank directly from the front. Enemies can exploit this weakness to fire perpendicularly at armoured plates, therefore significantly reducing the effectiveness of angled armour. Figure 2.4 illustrates the increase and decrease in relative armour thickness from two different shell impact angles. When the tank is facing the shell frontally, the right upper glacis plate (highlighted in black) is angled away from the shell and therefore has thick armour, while the lower glacis plate (highlighted in blue) is angled towards the shell and therefore has moderate armour. When the tank is facing diagonally, the right upper glacis plate suddenly becomes thin as the plate is angled directly towards the shell, while the lower glacis plate is angled significantly away from the shell, which will make it impenetrable from a diagonal attack.

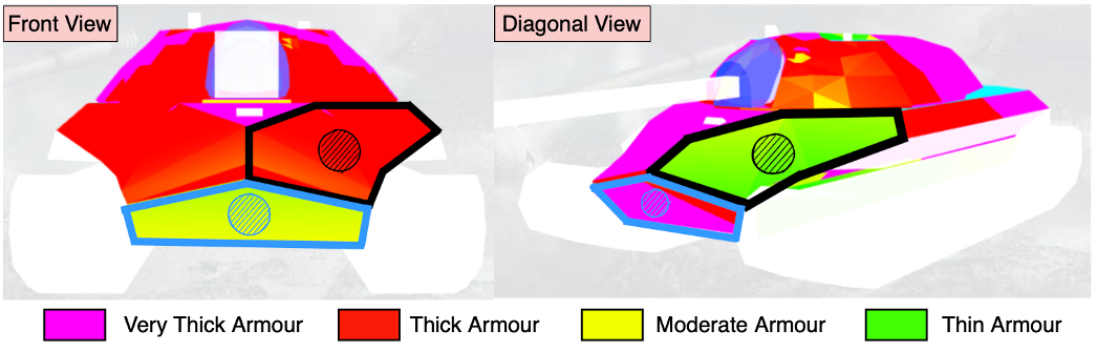

Figure 2.4: Different armour plates have different relative thicknesses when attacking from different angles (Adapted from TanksGG).

A daunting challenge in this project is to create a mechanism which enables the tank agents to manoeuvre themselves into advantageous positions which can exploit weak armour plates on an enemy tank, while simultaneously adjusting their own hull and turret angles to maximise the thickness of their armour from the enemy's perspective. This mechanism can be learned through reinforcement learning.

2.3 World of Tanks

World of Tanks (Wargaming) is a massively multiplayer online game where players fight against each other using hundreds of different Second World War and Cold War era tanks. Tank battles take place in a 15 vs 15 format on a diverse set of maps including deserts, mountains and cities. Every tank has its own strengths and weaknesses, and the player must make use of various tactics to maximise their score. World of Tanks has a steep learning curve, and it can take months for human players to fully grasp the mechanics of the game.

World of Tanks was immensely popular from 2012 through to 2016, but its player population has been steadily declining over the last few years. Therefore, bots were introduced onto servers with low populations, but these bots have

noticeably poor performance and can identified easily due to their tendency to repeatedly make critical errors in gameplay. An advanced bot which can utilise complex game mechanics effectively will improve the player experience on servers with low population.

2.4 Reinforcement Learning

Reinforcement learning is an exciting sub-field of artificial intelligence which involves training a model to perform optimal sequential decision-making in an environment to maximise reward.

The discovery of Q-learning (Watkins and Dayan, 1992) was one of the most influential research advances in reinforcement learning. The agent observes a state s_t and chooses an action a_t according to its policy π. After choosing this action, the agent receives a scalar reward r_t and moves to the next state s_{t+1}. This process is repeated until a terminal state is reached, or until the maximum number of state transitions has been exceeded. The action value $Q(s_t, a_t)$ denotes the expected return after taking action a_t in state s_t, which simply measures the performance of a policy. The optimal action value $Q^*(s_t, a_t)$ denotes the maximum possible reward by taking action a_t in state s_t. In model-free reinforcement learning methods, the action value function is approximated by a neural network with parameters θ. The parameters θ are learned by iteratively minimising a sequence of loss functions $L_t(\theta_t) = \mathbb{E}(r + \gamma max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{t-1}) - Q(s, a; \theta_t))^2$. Deep neural networks are ideal for approximating complex functions, but their introduction to reinforcement learning removes the guarantee of stable learning and convergence. Fortunately, recent research has identified new approaches to deep reinforcement learning which increase the stability of learning whilst also increasing the speed of convergence.

Chapter 3

Related Work

3.1 Introduction

With the rise of computing intelligence, ubiquity, interconnection and delegation, significant advances have been made in the field of artificial intelligence over the last decade. Supervised learning, unsupervised learning, and reinforcement learning are the sub-fields of artificial intelligence. Significant research has already been carried out in these sub-fields, and the knowledge gained from this can be used to provide a strong foundation on the topics which will be encountered in this project.

3.2 Trends and Themes

Reinforcement learning had traditionally been restricted to environments with low-dimensional discrete state and action spaces. However, in the real world, the state and action spaces are continuous. This presents a number of design challenges which cannot be overcome using traditional reinforcement learning, but can be overcome by incorporating deep neural networks.

Mnih et al. (2015), presented one of the first papers to explore the combination of deep neural networks with reinforcement learning. They investigated the use of deep neural networks to derive efficient representations of the environment from high-dimensional sensory data in Atari games with minimal prior domain knowledge. The raw frames from the emulator were initially preprocessed using a max-pooling approach for each pixel. Preprocessing is a vital step, as the raw frames are of high resolution with a large colour palette. The preprocessed frames were introduced to the neural network, and RMSProp was used to stabilise the learning process. The same network architecture and hyperparameters were used for each Atari game, demonstrating the robustness of this deep-Q learning approach. ε-greedy was used to control the balance between exploration and exploitation. The algorithm took completely random actions at the beginning and then eventually started to follow the policy, since ε was annealed from 1 to 0.1. The algorithm was trained over 50 million frames. The actual environment state was considered as a sequence of observations and actions in order to become a Markov decision process, since many frames are perceptually aliased. Experience replay was another innovative mechanism used by the authors, where the agent's state transition data is stored in a replay buffer. The Q-learning updates are performed using randomly sampled mini-batches from the replay buffer, which ultimately removes correlations in the observed sequence. The relatively low-dimensional representations learned by the deep-Q network were validated using t-SNE (van der Maaten and Hinton, 2008).This approach yielded results both superior to previous machine learning approaches and comparable to human players, with some games even exceeding human performance.

Silver et al. (2017) built upon these research advances to develop an algorithm, AlphaGo Zero, that exhibits superhuman ability in the complex game of Go. The neural network employed here has a similar structure to the neural network used

by deep-Q networks (Mnih et al., 2015), as both make strong use of convolutional layers and rectifier non-linearities. Here, the neural network outputs the probabilities of selecting each move and the probability of winning the game from the current state. These network outputs are used to guide a Monte Carlo tree search (MCTS), which itself outputs a stronger probability of selecting each move and winning the game from the current state. The neural network parameters are trained to minimise the difference between the outputs of the neural network and the MCTS. The MCTS uses the improved outputs from the network on each training iteration to produce even better outputs, which brings about a positive feedback loop. L2 regularisation is used to stabilise the learning process. The Q-learning updates were performed by randomly sampling mini-batches from a replay buffer which held the positions from the most recent 500,000 games. This algorithm was found to be superior to previous supervised learning approaches, but was also found to take different actions than these approaches, suggesting that the algorithm may be following a strategy that is qualitatively different to professional human players.

While deep-Q networks can effectively solve problems in high-dimensional environment spaces, it can only handle discrete and low-dimensional action spaces. A continuous action space can be broken down into a discrete action space, but this naive approach quickly breaks down for high-resolution action spaces with multiple features. Lillicrap et al. (2019) proposed an approach that infused deterministic policy gradient (Silver et al., 2014) with deep neural networks, known as deep deterministic policy gradient (DDPG). In this approach, there exists an actor (a function which maps states to actions) and a critic (a function which generates the Q values using the actor function), each of which have their own neural networks. The parameters of the actor and critic networks are improved by applying Q-learning updates by sampling mini-batches from a replay buffer.

The actor and critic neural networks use the target value to iteratively improve themselves, but the target value depends on the critic's output itself. This can bring about instability and divergence when training the networks. 'Soft' target updates were performed to overcome these issues. To perform soft target updates, the actor and critic networks (with parameters θ) are duplicated, and these duplicates (with parameters θ') are instead used to calculate the target value. The parameters of these target networks are updated by slowly tracking the parameters of the learned networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$, where $\tau \leq 1$. This mechanism breaks the direct link between the target value and the learned parameters, which boosts learning stability. A significant challenge with reinforcement learning in continuous spaces is exploration, and the authors overcame this hurdle by adding noise sampled from a noise process to the actor policy. Another challenge is the inconsistent range of the features, as different components of the environment space may have different units. Batch normalisation (Ioffe and Szegedy, 2015) was used to normalise the dimensions in the mini-batch during training. DDPG was found to train significantly faster for Atari games than the original deep-Q network for Atari games Mnih et al. (2015). DDPG yielded impressive results across many challenging continuous action space problems, including cartpole and gripper. DDPG, like most most model-free reinforcement learning approaches, requires a significant number of training episodes.

3.3 Debates and Conflicts

The use of experience replay in the approaches thus far has been a fundamental component to the success of these approaches, but replay buffers are computationally expensive to maintain. The original deep-Q network has been modified by Nair et al. (2015) to introduce parallel asynchronous training on distributed

devices, but this improved approach still requires significant computing resources which are not available for this project. Mnih et al. (2016) introduced a new approach, asynchronous advantage actor-critic (A3C), which executes multiple agents in parallel on multiple instances of the environment without the use of a replay buffer. A3C is computationally cheap, as it can be run on a single machine. Agent-environment pairs are executed on multiple threads in a CPU, which removes the communication overhead imposed by distributed parallel execution. Different exploration policies are implemented on each thread to promote diversity and reduces correlation, therefore removing the need for a replay buffer. The actor function uses a convolutional neural network with softmax activation, and the critic function uses a convolutional neural network with linear activation, both of which follow a similar architecture to the actor and critic networks presented by Lillicrap et al. (2019). Premature convergence was mitigated by adding the entropy of the actor function to the objective function, as proposed by Williams and Peng (2015). A3C was able to beat the deep-Q network approach presented by Mnih et al. (2015) in just half the training time. The authors discovered that performing updates from parallel agent-environment pairs to a shared model stabilised the learning process. Even though replay buffers are computationally expensive, the authors suggested that combining both A3C and experience replay can yield more powerful results.

Reinforcement learning approaches require a vast amount of data and training before reaching reasonable performance. When learning, their performance can be poor, which wastes both time and resources. In real life, humans learn faster by observing other humans complete a task, rather than completing a task by trial and error themselves. This imitation learning ideology was applied to reinforcement learning by Hester et al. (2018), where a small dataset of demonstration data was used to massively accelerate the learning process. In this approach,

training data in the form of state transitions is harvested from a human play session. These expert state transitions are permanently stored inside the replay buffer. The agent network is pre-trained by sampling mini-batches from the replay buffer and updated by applying a combination of four loss functions: 1-step double Q-learning loss, n-step double Q-learning loss, supervised large margin classification loss, and L2 regularisation loss. The Q-learning losses guide the network to satisfy the Bellman equation for Q-learning, while the supervised loss is used to classifying the demonstrator's actions. Since the expert state transitions only cover a narrow segment of the state space, the remaining state-action pairs in the state space do not have data to constrain them to realistic values. The supervised loss constrains the Q-values of unexplored state-action pairs in the state space to be less than the Q-values of the state-action pairs in the demonstration data. After pre-training, the Q-learning updates are performed by sampling a mini-batch from the replay buffer with prioritisation (Schaul et al., 2016). This blend of imitation learning and deep-Q networks gives a large boost in initial performance and enables the use of reinforcement learning in complex exploration problems.

Chapter 4

Data

In reinforcement learning, the states in the environment space and possible actions in the action space form the data that will be used in Q-learning updates.

The environment consists of an array of friendly tank objects and enemy tank objects. Each tank object has a number of different parameters that can be recorded at run-time, as shown in Table 4.1. Each tank has an individual reinforcement learning agent controlling it, and the actions taken by this agent will modify the agent's state.

Property Name	**Description**	**Type**
ID	The ID of the tank	Integer value
Type	The type of the tank	String value
Location	The location of the tank on the map	Continuous (x,y,z) value
Hitpoints	The health of the tank	Integer value
Hull Angle	The direction in which the tank is pointing	Triple continuous value in degrees
Turret angle	The direction in which the tank's turret is pointing	Triple continuous value in degrees
Gun angle	The angle of the tank's gun	Continuous value in degrees
Aimed Plate	The current armour plate the gun is aiming at, if any	String value
Velocity	The velocity of the tank	Continuous (x,y,z) value
Alpha Damage	The damage that can be caused by the tank's gun	Integer value
Muzzle speed	The muzzle velocity of the tank's rounds	Integer value in m/s
Penetration	The amount of armour which the tank's rounds can penetrate	Integer value in millimetres

Table 4.1: The parameters of a tank object.

Chapter 5

Methodology

5.1 Simulation of the Environment

The agents and the environment are to exist inside an armoured tank game. Game engines can be used to accurately model the behaviour of the agents and to realistically simulate the environment.

Unity is a popular cross-platform game engine which supports both 2D and 3D development. Unity is written in C-Sharp, a robust language that is simple, modern, and general purpose. Unity has many features, including customisable editors, post-processing, and a marketplace ecosystem with a vibrant community (Unity, b). Unity also provides a reinforcement learning framework, Unity ML-Agents (Unity, a), which can be used to train and embed intelligent agents.

Unity was chosen as the project's physics engine due to its native support for Blender 3D models, ease of use, and the integration of the Unity ML-Agents architecture. The ML-Agents architecture itself cannot be used to create a complex, strategic, efficient algorithm in a multi-agent environment, but it can be used as a baseline to measure the proficiency of the final algorithm.

A number of environments (game maps) will be created to analyse the be-

haviour of the agent across different settings. The game maps will contain indestructible objects, such as boulders and large buildings, and destructible objects, such as fences and trees. The tank agents will need to learn to use suitable cover, such as buildings and hills, to their advantage in order to protect them from enemy fire.

5.2 Proposed Algorithm

A successful tank agent will need to display prowess in three key areas to be successful - movement, armour knowledge, and tactics knowledge. There are multiple types of tanks in the project, and separate reinforcement learning parameters will be needed due to their varying combat capabilities.

5.2.1 Tank Movement

First and foremost, the tank agent will need to have the ability to move around the environment. The agents should be able to both avoid obstacles when necessary, and also be able to use obstacles as cover when necessary.

A primitive approach would use raycasts for obstacle detection and collision avoidance. A raycast is an infinite ray which is fired outwards in a certain direction from a point in space, allowing for the details of any objects colliding with this ray to be analysed. Ten different raycasts can be fired outwards in specific directions from the agent's location, and the details of these raycasts can be leveraged by the agent to introduce spatial awareness, as shown in Figure 5.1. However, this would take a greedy approach to path-finding, therefore lacking the ability to take alternate paths to a target. Reinforcement learning can be combined with the raycast approach to develop advanced path-finding capabilities.

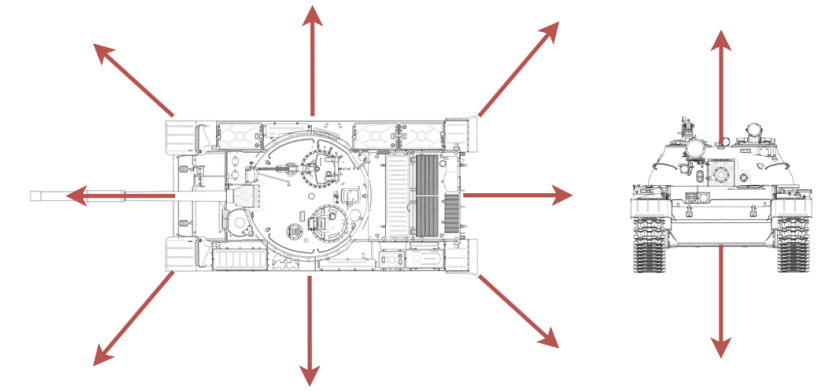

Figure 5.1: Raycasts can be used to add spatial awareness to the agent.

However, the simplest and most effective approach to tank movement is Unity NavMesh. This is a Unity feature that automatically generates impassable boundaries and zones of movement in the environment. NavMesh is both quick to set up and is highly customisable with modifiable properties including traverse speed, acceleration, and stopping distance.

5.2.2 Armour Knowledge

There are multiple different types of tanks in this project. Heavy tanks and medium tanks have armour layouts which need to be considered in order to penetrate their armour. Reinforcement learning can be used to train agents by providing knowledge on armour angling and weak spots. A training environment consisting of a single enemy tank and an agent tank will be created. Shells will be fired randomly by the agent tank at the enemy tank, and a reward is given to the agent if it penetrates the armour of the enemy tank. The enemy tank cannot return fire, but it can adjust its hull angle and turret angle to maximise its overall armour and protect itself from the agent tank. In this scenario, the attributes in the spate space would be the hull angle of the enemy tank, the turret angle of the enemy tank, and the velocity of the shell launcher. The reinforcement learning

algorithm for the agent tank will learn a policy which outputs the best armour plate to aim at on the enemy tank when given these variables. The reinforcement learning algorithm for the enemy tank will learn a policy which outputs the best hull and turret angles when given the vector to the agent tank. The representations learned by both deep-Q networks will validated using t-SNE.

The reinforcement learning model in this mechanism will be inspired by recent research. The deep deterministic policy gradient algorithm (Lillicrap et al., 2019) will be used to perform the agent and enemy parameter changes in a continuous action space. High performance computing resources are not available for this project, therefore massively parallel asynchronous training(Mnih et al., 2016) will be used to train multiple agent-environment pairs on multiple CPU threads simultaneously with n-step Q-learning. Demonstrator data will be provided to the tanks from a human play session prior to learning (Hester et al., 2018). Since both algorithms are adversaries to each other, this training mechanism will behave similarly to generative adversarial networks (GANs). However, GANs are very difficult to converge, particularly due to the issues of mode collapse (the agent tank produces a limited variety of shots) and diminished gradient (the enemy tank becomes too successful and the agent tank's gradient will therefore vanish). Stable GAN training is a new research area that is actively being explored, therefore limited information is available about GAN training techniques which guarantee convergence. If interleaved training does not converge when training the agent tank and enemy tank simultaneously, these algorithms can simply be trained independently.

5.2.3 Tactics Knowledge

Raw armour angling is one of the many tactics which tanks can utilise to protect themselves from incoming fire. More complex tactics can be used by exploiting

the shape of the terrain and the obstacles in the map. For example, a desert map has many sand dunes where hull-down tactics can be used. In a city map, hard cover such as buildings can be used to perform sidescraping, as shown in Figure 5.2.

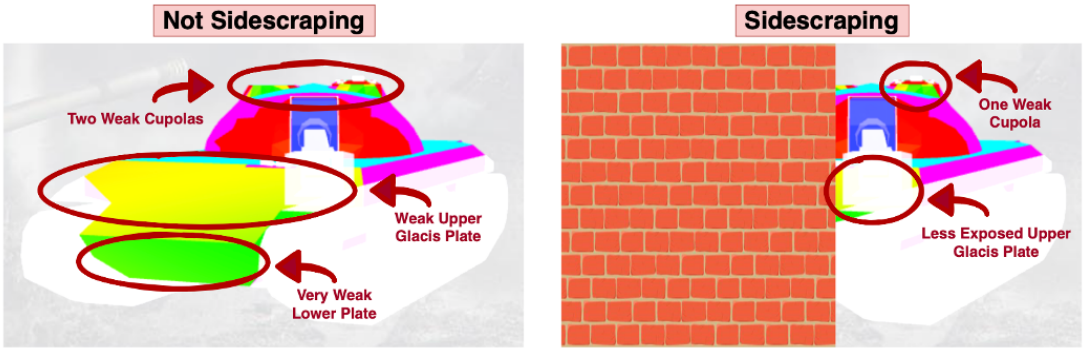

Figure 5.2: Sidescraping behind solid cover only exposes the heavily angled side armour to the enemy.

There are many methods to teach tactics knowledge to the agents using reinforcement learning. Deep Q-learning from demonstrations can be used to teach the agents currently known gameplay tactics. This will allow the agents to have relatively good performance early during training, which will cut down the training time and resource costs. Asynchronous n-step Q-learning (Mnih et al., 2016) will be used to train the networks. With this approach, the algorithm will be trained in a massively parallel fashion, with each CPU thread running a simulation of identical maps in a 1 vs 1 battle format. One episode of training will last until one tank has been defeated. The agents will receive a reward which is a function of their damage caused, damage blocked by armour, and total survival time. This will undoubtedly be a challenging component of this project, since there are numerous attributes to each tank at run-time, and the values of each attribute affects the outcome of each episode.

Chapter 6

Experimental Settings

After training, the agents will be placed in a 15 vs 15 battle on each map used during the training process. Every map in the training process will be closely related to existing maps in World of Tanks, allowing for a fair comparison between the algorithm's optimal policy and a human player's optimal policy.

Heatmaps will be used to analyse the movements of different types of tanks over the battles. Heatmaps can also be superimposed on top of tank armour models to highlight the weakest armour as perceived by the agents. The movement heatmaps will be compared against human player heatmaps (WoTInspector), and the armour heatmaps will be compared against armour thickness models (TanksGG).

The agent's combat capability will then be compared against an experienced human player in a team battle, and the scores of the human player and the agents will be analysed. Finally, the agent's policy will be tuned by adjusting ϵ, the exploration parameter, to modify the skill level of the agents. A number of inexperienced players will then be placed in team battles with the agents, and their feedback regarding enjoyment and agent difficulty will be recorded.

Chapter 7

Conclusion

This project will develop advanced intelligent agents that can master tank-based warfare across a variety of different settings. This is a challenging task, as tanks have a diverse set of parameters at run-time which need to be optimised to maximise the cumulative reward given to the agent. The agents will learn to incorporate the different features of the terrain to take advantage of gameplay tactics such as sidescraping and hull-down positions. Recent advances in machine learning such as deep-Q learning and deep deterministic policy gradient will be used to produce a reinforcement learning approach which displays stable learning and relatively fast convergence. The low-dimensional representations learned by the deep-Q networks will be verified using t-SNE. The overall performance of the algorithm will be compared against human performance by comparing the scores, rates of armour penetration, and tank survival time. The tactics used by the agents will be compared against known gameplay tactics by the use of heatmaps. One of the limitations of this project is the lack of team-play, due to the limited research surrounding collaborative reinforcement learning in multi-agent systems. However, even though agents are not explicitly awarded for teamwork, they are expected to settle into roles which mutually benefit their team-mates.

References

Hester, Vecerik, Pietquin, Lanctot, Schaul, Piot, Horgan, Quan, Sendonaris, Osband, Dulac-Arnold, Agapiou, Leibo, and Gruslys. Deep q-learning from demonstrations. In *Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 11, 18

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, 07–09 Jul 2015. 10

Lillicrap, Hunt, Pritzel, Heess, Erez, Tasa, Silver, and Wiersta. Continuous control with deep reinforcement learning. In *Proceedings of ICLR 2016*, 2019. 9, 11, 18

Viktor Malginov. T62 russian medium tank blueprints. `https://drawingdatabase.com/t-62-tank/`. Accessed: 2021-20-06. v, 3

Megapixie. Hull-down tactics demonstration. `https://commons.wikimedia.org/wiki/File:Hull_down_tank_diagram.png`. Accessed: 2021-20-06. v, 3

Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, Ostrovski, Petersen, Beattie, Sadik, Antonoglou, King, Kumaran, Wier-

stra, Legg, and Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. 7, 9, 10, 11

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, 2016. 11, 18, 19

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. Massively parallel methods for deep reinforcement learning. In *Proceedings of The International Conference on Machine Learning*, 2015. 10

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silvers. Prioritized experience replay. In *Proceedings of ICLR 2016*, 2016. 12

Silver, Lever, Hess, Degris, Wierstra, and Riedmiller. Deterministic policy gradient algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China, 22–24 Jun 2014. PMLR. 9

Silver, Schrittwieser, Simonyan, Antonoglou, Huang, Guez, Hubert, Baker, Lai, Bolton, Chen, Lillicrap, Hui, Sifre, van den Driessche, Graepel, and Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017. 8

TanksGG. Wot armour models. `https://tanks.gg`. Accessed: 2021-25-06. v, 5, 20

Unity. Unity ml agents. `https://unity.com/products/machine-learning-agents`, a. Accessed: 2021-10-06. 15

Unity. Unity website. `https://unity.com`, b. Accessed: 2021-10-06. 15

van der Maaten and Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2580–2605, 2008. 8

Wargaming. World of tanks webpage. `https://worldoftanks.eu`. Accessed: 2021-26-06. 5

Watkins and Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992. 6

Ronald J. Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3:241–268, 2015. 11

WoTInspector. Wot inspector heatmaps. `https://wotinspector.com/en/heatmaps/`. Accessed: 2021-25-06. 20