Undergraduate Thesis

MULTI-AGENT ADVERSARIAL REINFORCEMENT LEARNING IN A MUSEUM HEIST SCENARIO

Spyridon Sachmpazidis

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2026

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Multi-Agent Adversarial Reinforcement Learning in a Museum Heist Scenario

Spyridon Sachmpazidis

Supervisor

Dr. Andreas Aristidou

The Thesis was submitted in partial fulfillment of the requirements for obtaining the Computer Science degree of the Department of Computer Science of the University of Cyprus

May 2026

I declare that this dissertation and any accompanying software are my own original work, conducted in full compliance with the University of Cyprus Code of Conduct for Research (https://www.ucy.ac.cy/legislation/kodikas-deontologias-kai-politikes/), and with full accountability on my part for the accuracy, integrity, and validity of all content.

Acknowledgements

I would like to express my gratitude to my supervisors Dr. Andreas Aristidou for his guidance, advice, and patience throughout this project.

I would also like to thank Dr. Panayiotis Charalambous for taking the time to evaluate my thesis as part of the examination committee.

Finally, I would like to thank my friends and family for their continuous support.

Abstract

Non-Playable Characters (NPCs) play an integral role in making games more interactive, by acting as intelligent opponents or allies to players. In most games however, these characters are manually designed and controlled by scripts, thus making their behavior quite predictable after multiple interactions. Reinforcement Learning offers an alternative approach with unique benefits, where the characters can be controlled by intelligent agents that are able to learn behaviors through experimentation instead of following specified command. This project is an extension of a multi-agent collaborative museum heist scenario implemented in Unity using ML-Agents. The environment consists of two opposing teams: robbers vs guards. The robbers are assigned with the task of stealing valuables from a randomly generated museum map and escape without being caught, while the guards (and security cameras) are on a mission to detect and stop them. The robber team consists of two specialized agents, the Locksmith and the Technician, with the abilities of opening gates and disabling cameras respectively. In previous work, the robbers were trained against scripted guards. However, this work investigates how the scenario changes when the guard team is also controlled by learned agents. Guard agents are trained using Proximal Policy Optimization (PPO) and we evaluate if previously trained robber policies can generalize to these newly learned guards. We then examine if the robber team can improve when trained again directly against the learned guard policy. Finally, we introduce a decoy mechanic that allows robbers to distract guards and investigate whether this additional action has an impact on the robber team's performance. The experiments are evaluated using robber win rate, guard win rate, alarm outcomes, collected valuables, and episode duration. The results show that learned guards reduced the success rate of the original robber policies, while adaptive training and the decoy mechanic barely improved robber performance. Overall, the project contributes to the development of more adaptive and less predictable NPC behavior in game environments, while also highlighting that adversarial reinforcement learning requires careful design of opponent adaptation, reward signals, and balanced agent capabilities rather than a straightforward training process.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Topic

The use of Machine Learning in video games has created new possibilities for developing more adaptive NPCs, primarily using Reinforcement Learning methodologies that enable agents to improve their behavior via environmental interactions throughout numerous repetitions [1]. NPCs are a key factor in many games because they can act as enemies, teammates, guards, civilians, or other interactive characters in the game world. Their behavior directly influences how challenging and engaging the game feels to the player. When they behave in a predictable way, players can often exploit their weaknesses to win easily.

This thesis focuses on a multi-agent adversarial reinforcement learning scenario in a museum heist environment, extending the museum robbery environment introduced in previous work [4]. The environment contains two opposing teams: robbers and guards. The robbers attempt to steal valuables from the museum and escape, while the guards and security cameras attempt to detect them and prevent the robbery. The robber team consists of two agents with different skills. The Locksmith can open locked gates, while the Technician can disable security

cameras. The guard team is responsible for patrolling the museum, detecting robbers, protecting valuables, and triggering the alarm.

The scenario is adversarial because the two teams have conflicting objectives, and it is asymmetric because the agents do not all share the same abilities or reward functions. This makes the environment suitable for studying cooperation between teammates, competition between opposing teams, and the effect of learned defensive behavior on previously trained robber agents.

1.2 Motivation

The motivation for this work comes from the need to create NPCs that are less predictable and more responsive to their environment. Traditionally, NPC behavior is implemented using hard coded methods such as Finite State Machines (FSM), Decision Trees, Behavior Trees, navigation systems, and scripted patrol routes. Designers tend to have an easier time with these methods since they are relatively straight forward to understand and control, but they also limit the behavior of the agents. For instance, a scripted guard may always follow the same route or react in the same way as the player enters its field of view, thus making the game less challenging and stale.

This is especially important in stealth or adversarial games, where the behavior of guards or enemies directly affects the player's strategy. If the defensive agents are fixed, the opposing team may learn to exploit their specific patterns. Reinforcement Learning offers a way to create agents that learn from interaction instead of following fixed rules, by optimizing their behavior according to reward signals received from the environment [1]. An agent observes the environment, chooses actions, and receives rewards or penalties depending on the result of those actions. Through repeated training, the agent can learn behaviors that are not directly programmed by the developer. This makes Reinforcement Learning useful for creating agents that adapt to different situations.

The previous version of the museum robbery project focused on training the robber team against scripted guards [4]. Although this allowed the robbers to learn cooperative behavior, the defensive side remained fixed. This creates an important motivation for this thesis: to investigate what happens when the guards are also trained agents. If both sides are controlled by learned policies, the environment becomes more dynamic and adversarial.

1.3 Problem

The key problem addressed in this thesis is that robber agents trained against scripted guards may not perform well against learned guards. A policy that succeeds against a fixed opponent may learn to exploit that opponent's specific weaknesses. For example, if the guards always follow scripted patrol paths, the robbers may learn strategies that work only because those paths are predictable.

This becomes an issue when the defensive behavior changes. Learned guards move differently, detect robbers in different situations, and protect valuables more effectively. Therefore, the robber policy fails to generalize to the new opponent. This is a common issue in adversarial multi-agent environments, where the behavior of one team affects the difficulty faced by the other team.

The original project did not fully address this issue because the guards were scripted. The focus was the cooperation between the two robber agents. This thesis extends the problem by introducing trainable guard agents and studying how their behavior affects the pretrained robbers. It also investigates whether the robbers can improve when they are trained or adapted against the learned guard policy.

1.4 Usefulness

This work is useful because it explores how Reinforcement Learning can be applied to a game-like adversarial environment where both cooperation and competition are present. The museum heist scenario includes continuous movement, discrete interactions, partial observability, role-specific abilities, guards, cameras, valuables, and alarm conditions. These elements make it more complex than a simple navigation task and allow it to be used as a small experimental environment for studying learned NPC behavior.

The project is also useful because it tests whether agents trained against one type of opponent remain effective when the opponent changes. By comparing pretrained robbers against learned guards, and then training robbers against those learned guards, the experiments examine the importance of adaptation in adversarial reinforcement learning. The decoy mechanic further extends the scenario by adding a deception-based action, which is relevant to stealth and strategy games.

1.5 Brief Implementation

The project was implemented in Unity using the ML-Agents Toolkit, which provides tools for training agents inside Unity simulations [3]. The agents were trained using Proximal Policy Optimization, a policy-gradient reinforcement learning algorithm commonly used for continuous and discrete control problems [2]. The museum environment contains four rooms, locked gates, valuables, security cameras, guards, a starting region, and a goal region.

The implementation was divided into three experimental stages. First, scripted guards were replaced with trainable guard agents, while the pretrained robber models from the previous project were kept fixed. Each guard observes its own movement, its assigned room, alarm state, protected valuable information, and robbers detected through its field of view. The guards use continuous movement actions and discrete rotation actions, and their reward function encourages detection, alarm success, and valuable protection.

Second, the robber team is trained again against the learned guard policy to test whether adaptation improves performance. Third, a decoy mechanic is added to give the robbers a possible way to distract or mislead guards. The experiments are evaluated using robber win rate, guard win rate, alarm outcomes, number of collected valuables, and episode duration.

1.6 Contribution

The main contribution of this thesis is the extension of the museum heist environment from a setting with trained robbers and scripted guards [4] to a more adversarial setting where the guards are also trained using Reinforcement Learning. This allows the project to test whether robber agents trained against scripted opponents can still succeed against learned defensive agents.

A second contribution is the design of a learned guard baseline. The guards are given observations, actions, and rewards that support defensive behavior, including robber detection, valuable protection, and alarm triggering. This changes the defensive side from a fixed scripted system into a learned policy.

A third contribution is the evaluation of adaptation and deception in the robber team. The thesis investigates whether robbers improve when trained against the learned guard policy, and

whether a decoy mechanic can further improve their performance by adding a distraction-based strategy.

The results suggest that robber performance can improve when the agents are trained against the learned guard policy, and that the decoy mechanic can provide an additional strategic tool for dealing with adaptive defensive agents

Chapter 2

Theoretical Background

2.1 Reinforcement Learning in Games

Games have often been used as test environments for Artificial Intelligence because they provide clear rules, repeated interactions, and measurable objectives. In a game environment, an agent can perform actions, observe the result of those actions, and receive feedback based on whether its behavior helps it succeed. This makes games suitable for Reinforcement Learning, where the goal of an agent is to learn a policy that maximizes its cumulative reward [1].

Reinforcement Learning has been applied to many different types of games. Early examples focused on board games such as Backgammon, Chess, and Go, where the rules are discrete and the outcome of the game is clearly defined. These games helped demonstrate that learning-based agents can discover strong strategies through repeated play. Deep Reinforcement Learning was later applied to Atari games, showing that agents could learn successful behavior directly from game interaction and reward feedback [5]. However, many modern video games are more complex than traditional board games. They often include real time decision making, continuous movement, partial observability, multiple agents, and long term planning.

This is important for game NPCs because their behavior must often react to a changing environment. A scripted NPC can follow predefined rules, but it cannot easily adapt to

situations that were not considered by the developer. Reinforcement Learning allows an NPC to learn behavior through experience. Instead of manually defining every possible reaction, the developer defines observations, actions, and rewards. The agent then learns which actions are useful through trial and error.

In this thesis, Reinforcement Learning is used to train agents in a museum heist environment. The agents must move in a continuous 3D space, avoid or detect enemies, interact with objects, and complete team based objectives. The environment is dynamic, partially observable, and adversarial.

Figure 2.1: Reinforcement Learning Loop

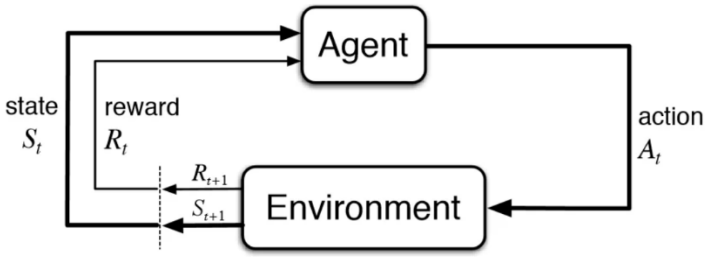

Figure 2.1: Basic reinforcement learning interaction loop between an agent and an environment.

2.2 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning studies environments where more than one agent acts at the same time. These agents may cooperate, compete, or do both depending on the task. In cooperative environments, agents share the same objective and must learn to coordinate their actions. In competitive environments, the success of one agent or team may reduce the success of another. Some environments contain both cooperation and competition [6].

The museum heist scenario is a multi-agent environment because it contains several agents with different roles. The robber team contains two agents, the Locksmith and the Technician, that must cooperate. The Locksmith can open gates, and the Technician can disable cameras, so neither agent has all the abilities needed to solve the task alone. The guard team has a different objective: to protect the museum and stop the robbers. This creates a setting where cooperation exists within each team, while competition exists between teams.

Multi-agent environments are more difficult than single-agent environments because each agent is part of the environment observed by the other agents. If one agent changes its policy

during training, the environment becomes non-stationary from the point of view of the other agents. For example, if guards become better at detecting robbers, the robber team must adapt its strategy. Similarly, if robbers learn to avoid guards more effectively, the guards must learn better defensive behavior. This problem is common in mixed cooperative-competitive reinforcement learning environments [7].

This interaction is one of the main reasons multi-agent learning is interesting for games. It can lead to behaviors that are not directly scripted by the developer. Agents may learn to cooperate, avoid threats, exploit weaknesses, or counter the strategies of other agents. However, it also makes training more unstable and sensitive to reward design.

2.3 Adversarial and Asymmetric Environments

Adversarial environments contain agents or teams with opposing objectives. In many games, one team tries to complete a task while another team tries to prevent it. These environments can be symmetric or asymmetric. In a symmetric game, both sides usually have similar abilities and objectives. For example, in many sports games, both teams can attack and defend in similar ways. In an asymmetric game, the teams have different roles, abilities, or objectives.

The museum heist environment is adversarial and asymmetric. The robbers try to steal valuables and escape, while the guards try to detect them and protect the museum. The two teams do not have the same actions or rewards. The robbers are rewarded for collecting valuables and reaching the goal, while the guards are rewarded for detecting robbers, triggering the alarm, patrolling, and protecting valuables. This makes the environment different from simple zero-sum games where one player's reward is exactly the negative of the other player's reward.

The environment is also asymmetric within the robber team. The Locksmith and Technician have different abilities, which means they require different policies and slightly different observation and action spaces. This creates a role-based cooperation problem. The agents must not only avoid the defensive team but also use their own abilities in ways that help the team.

This type of environment is useful for studying game AI because many real games are asymmetric. Stealth games, team-based shooters, strategy games, and role-based multiplayer games often contain agents with different abilities and objectives. Large-scale multi-agent game environments, such as StarCraft II and Dota 2, have shown that reinforcement learning

can be applied to complex adversarial scenarios with partial observability and longtime horizons [8], [9]. A museum heist is a simplified example of such a setting, but it still contains important elements such as navigation, detection, cooperation, and adversarial behavior.

2.4 Self-Play and Adaptive Opponents

In adversarial learning, the choice of opponent is very important. If an agent is trained only against a fixed opponent, it may learn strategies that exploit that opponent rather than strategies that are generally robust. This can be useful in the short term but may fail when the opponent changes. Self-play and opponent adaptation are common ways to address this problem.

In self-play, an agent or team trains against versions of itself or against past versions of the opponent. This can create an automatic increase in difficulty. When one side discovers a stronger strategy, the other side must adapt. This idea has been used in several important game AI systems. For example, Alpha Star used league-based training in StarCraft II, OpenAI Five used large-scale reinforcement learning in Dota 2, and Hide-and-Seek showed how multi-agent competition can produce an automatic curriculum of new strategies [8], [9], [10].

This idea is directly related to the museum heist environment. In the previous version of the project, the robbers were trained against scripted guards [4]. This allowed the robber agents to learn a strategy, but the opponent did not adapt. As a result, the robbers may have learned behaviors that are effective only against that fixed defensive logic. In this thesis, the guards are trained using Reinforcement Learning, creating a different type of opponent. This allows us to test whether the original robber policies generalize to learned guards.

The second stage of the project is based on the idea of adaptation. After training the guard policy, the robber team can be trained or fine-tuned against the learned guards. This allows the robbers to experience the new defensive behavior during training. If their performance improves, this suggests that adaptation against a stronger or different opponent can help produce more robust robber behavior.

2.5 NPC Behavior, Guards, and Deception

NPC guards are common in stealth and action games. Their role is often to patrol an area, detect the player, and respond to suspicious behavior. Scripted guards are usually implemented using patrol routes, field-of-view checks, and state transitions such as idle, patrol, chase, and alarm.

These methods are part of traditional game AI and are widely used because they are understandable and controllable by developers [11]. This can create believable behavior, but it can also become predictable if the same logic is repeated every time.

A learned guard can potentially behave less predictably. Instead of following a fixed path, it can learn movement patterns that are rewarded by the environment. In this thesis, the guards are encouraged to patrol, detect robbers, and protect valuables. The goal is not only to make the guards move, but to make their movement useful for the defensive task.

The decoy mechanic introduces another concept: deception. In many stealth games, players can distract guards by throwing objects, creating sounds, or placing traps. These mechanics allow the player to manipulate the attention of the guard. In the museum heist environment, a decoy gives the robber team an additional strategic option. Instead of only avoiding guards, the robbers may attempt to influence guard's behavior.

This is useful because deception is an important part of adversarial environments. In multi-agent settings, agents may develop strategies that influence or mislead opponents, as seen in competitive environments where new behaviors emerge through interaction [10]. If the guard policy reacts to the decoy, the robber team may use it to create safer paths, reduce detection, or gain time to collect valuables. However, adding a decoy also increases the complexity of the action space and training process. The agent must learn not only how to move and interact with existing objects, but also when the decoy action is useful.

Chapter 3

Methodology and Implementation

3.1 Environment Overview

The methodology of this thesis is based on extending the existing Unity museum heist environment. The environment represents a robbery scenario where two opposing teams interact inside a museum. The first team is the team of robbers, whose goal is to steal valuables and escape. The second team is the defensive team, composed of guards and security cameras, whose goal is to detect the robbers and prevent the robbery from succeeding.

The museum consists of four connected rooms. Each room contains obstacles, security cameras, possible valuable spawn positions, and one guard. At the beginning of each episode, the robbers spawn in a starting region outside the museum. A goal region is also connected to the museum, and the robbers must move through the rooms before reaching it. The museum layout forces the robbers to pass through multiple rooms, interact with gates, avoid cameras, and deal with guards before completing the heist.

Each room contains one active valuable. The active valuable is selected randomly from a set of possible spawn positions at the beginning of each episode. This means that the robbers cannot always follow the same path to collect the valuables. The guards are also assigned to protect the active valuable in their room, which allows the defensive reward function to penalize a guard if the valuable under its responsibility is stolen.

The security cameras use a FOV detection system. A robber is detected by a camera when the robber enters the camera's detection area and a raycast confirms that there is no obstacle blocking the view. The guards use a similar idea for detecting robbers. A robber must be inside the guard's field of view, and a raycast must confirm that the robber is visible. If the cumulative detection time becomes high enough, the alarm is triggered.

In the previous version of the project, the guards followed scripted behavior. In this thesis, the main methodological change is that the guards are converted into trainable reinforcement learning agents. This allows the defensive side to learn its behavior instead of following fixed patrol routes.

Figure 3.1: Museum Heist Environment Overview

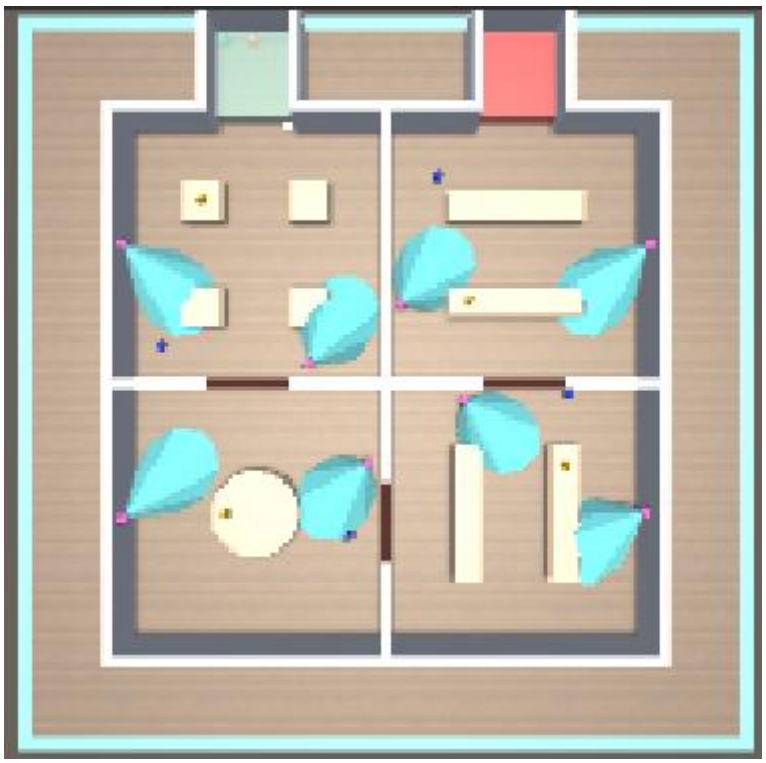

Figure 3.1: Overview of the museum heist environment, including rooms, guards, cameras, valuables, and robber start/goal areas.

3.2 Agent Roles and Abilities

The environment contains multiple agents with different roles. The robber team consists of two agents: the Locksmith and the Technician. Both robbers can move, rotate, collect valuables, and reach the goal region. However, each robber also has a special ability. The Locksmith can open locked gates between rooms, while the Technician can disable security cameras. These specialized abilities make cooperation important, because each agent can help the team in a different way.

The guard team contains one guard assigned to each room. In the original environment, the guards were scripted and followed predefined patrol behavior. In this implementation, each guard is controlled by a learned policy. The guards are responsible for patrolling their assigned area, detecting robbers, contributing to alarm activation, and protecting the valuable in their room.

The agents therefore form an asymmetric multi-agent environment. The two teams do not have the same goals, actions, or rewards. The robbers are rewarded for stealing and escaping, while the guards are rewarded for defensive behavior. Even within the robber team, the agents are asymmetric because their special abilities are different. This makes the environment suitable for studying both cooperation and competition.

3.3 Learned Guard Implementation

The main extension implemented in this thesis is the replacement of scripted guard behavior with learned guard behavior. Each guard is implemented as a Unity ML-Agents Agent. The guard policy is trained using Proximal Policy Optimization. All guards share the same behavior name, meaning that their experiences are used to train one common guard policy.

Each guard has access to local information rather than complete information about the whole environment. This was done to avoid giving the guards unrealistic knowledge. A guard observes its own movement, its room number, the alarm state, the active valuable it is responsible for protecting, and information about robbers only when they are detected. If a

robber is not currently detected, the guard does not receive the robber's actual position or velocity. This prevents the learned guard from becoming omniscient.

The guard can move using two continuous actions. These actions control horizontal and vertical movement relative to the guard's local orientation. Rotation is controlled through a discrete action branch with three possible values: no rotation, rotate clockwise, or rotate anticlockwise. The guard action space also contains a second discrete branch related to decoy investigation. During the baseline experiments, the decoy system is disabled, so this branch does not affect the environment. In the decoy experiment, this branch allows guards to interact with decoys when the decoy system is enabled.

The guards detect robbers using a field of view system. The final guard field of view angle used in the implementation is 150 degrees. When a robber is within the guard's detection radius and angle, a raycast is used to check whether the robber is visible or blocked by an obstacle. If the robber is visible, the guard receives a detection reward and its detection counter increases. The total detection time across guards is tracked by the guard controller. When the total detection threshold is reached, the last guard that detected the robber triggers the alarm.

This implementation keeps the original idea of cumulative guard detection but changes the way guards decide where to move. Instead of following fixed patrol points, the guards learn movement behavior from rewards. Their reward function encourages movement through patrol-cell coverage, robber detection, valuable protection, and successful alarm outcomes.

3.4 Robber Policies and Decoy Mechanic

The robber agents used in the baseline experiment are the pretrained robber policies from the previous curriculum training. During the guard baseline experiment, these robber policies are kept fixed in inference mode. This allows the experiment to test whether robber agents trained against scripted guards can still perform well against learned guards.

The Locksmith and Technician retain their original abilities. The Locksmith can open gates when close enough to a gate and when selecting the appropriate action. The Technician can disable security cameras when close enough to a camera and when selecting the "disable" action. Both agents can pick up valuables and reach the goal area.

A decoy mechanic is also introduced as part of the final experiment. The purpose of the decoy

is to give the robber team a deception based action. In stealth games, distraction mechanics are often used to manipulate guard attention. In this environment, the decoy is intended to distract guards or create an opportunity for the robbers to move more safely.

The decoy mechanic required special attention because changing an agent's action space can make old, trained models incompatible. For this reason, the baseline experiments keep the original robber action space and disable the decoy action. The decoy is only enabled in the experiment where the robber policy is trained with the decoy mechanic included.

The decoy mechanic was implemented as an additional robber ability that can be enabled or disabled depending on the experiment. In the baseline experiments, the decoy action is disabled to preserve compatibility with the pretrained Curriculum II robber models. In the decoy experiment, the robber policy is trained with the decoy system enabled.

When a robber uses the decoy action, a decoy object is spawned at the robber's current position. The decoy remains active for a limited duration and acts as a distraction target for nearby guards. Guards receive decoy related observations, including whether a decoy is available and its relative position. If the decoy system is enabled and a guard chooses the investigation action, the guard can redirect its attention towards the decoy instead of only reacting to robber detection. This gives the robber team a possible way to influence guard movement and create safer opportunities to move, collect valuables, or reach the next room.

The purpose of the decoy is not to guarantee a robber win, but to introduce a deception based strategy into the environment. Its effect is evaluated by comparing robber win rate, valuables collected, alarm source, and episode duration between experiments with and without the decoy mechanic.

3.5 Observations and Actions

Observation and action spaces are important because they define what each agent can know and what each agent can do. The robbers and guards have different observation spaces because their roles are different. The robbers need information that helps them move through the museum, collect valuables, interact with gates and cameras, and cooperate with the other robber. The guards, on the other hand, need information that helps them patrol, protect their assigned valuable, detect robbers, and contribute to alarm activation.

The robber agents observe information about the museum layout, their own state, the state of gates and cameras, valuables, the goal region, the guard in the current room, and the fellow robber. The Locksmith and Technician have slightly different observations because of their different abilities. When the decoy mechanic is disabled, the vector observation size is 56 for the Locksmith and 57 for the Technician. When the decoy mechanic is enabled, two additional observations are added to each robber: whether the agent can throw a decoy and the number of rooms already used for decoys. Therefore, the decoy enabled observation size becomes 58 for the Locksmith and 59 for the Technician.

The guards use a vector observation space of size 20, shown in Table 3.1. The guard observations include the guard's local velocity, room number, detection information for the two robbers, protected valuable information, decoy information, and alarm state. For each robber, the guard observes a detection flag, relative position, and relative velocity. However, when the option to observe robbers only when detected is enabled, the relative position and velocity values are set to zero unless the robber is currently detected through the guard's field of view and raycast system. This preserves partial observability and prevents the guard from receiving unrealistic global robber information.

Table 3.1: Guard vector observation space.

Guard Vector Observation	Dimensions
Guard local velocity	2
Current room index	1
Locksmith detected flag	1
Locksmith relative position	2
Locksmith relative velocity	2
Technician detected flag	1
Technician relative position	2
Technician relative velocity	2
Protected valuable exists flag	1
Protected valuable relative position	2
Decoy active/in range flag	1
Decoy relative position	2
Alarm active flag	1
Total	20

The guard also observes information about the active valuable it is assigned to protect. This consists of whether the protected valuable exists and its relative position. This observation was included because the guards are not only trained to detect robbers, but also to protect the valuable in their assigned room. Without this information, the guard would not have a direct observation that connects it to the object it is supposed to defend.

Decoy observation slots are also included in the vector. When the decoy system is disabled, these values remain zero. When the decoy system is enabled, the guard can observe whether a decoy is in range and its relative position. This design allows the same guard observation size to be used in both the baseline and decoy experiments, while still allowing the decoy system to affect the guard policy when enabled.

In addition to vector observations, the guards use movement ray sensors. These rays are used only for navigation and obstacle awareness. The front movement rays detect walls, gates, and obstacles, while the back movement rays provide limited rear obstacle detection. These ray sensors do not detect robbers, valuables, decoys, goals, or other guards. Robber detection is therefore still controlled by the manual field of view and raycast logic, not by the movement ray sensors. This distinction is important because the movement rays help guards avoid obstacles, but they do not give the guards extra knowledge about robber positions.

The guard action space is shown in Table 3.2. It consists of two continuous movement actions and two discrete branches. The continuous actions control movement relative to the guard's local orientation. The first discrete branch controls rotation with three possible actions: no rotation, rotate clockwise, or rotate anticlockwise. The second discrete branch is reserved for decoy investigation. During baseline training, the decoy system is disabled, so only movement and rotation affect the guard's behavior. During the decoy experiment, this branch allows the guard to interact with decoys when they are available.

Table 3.2: Guard action space used by the learned guard agents.

Guard Action	Type / Values
Horizontal movement	Continuous value in [-1, 1]
Vertical movement	Continuous value in [-1, 1]
Rotation	{no rotation, rotate clockwise, rotate anticlockwise}
Decoy investigation	{do not investigate, investigate}

3.6 Reward Design

Reward design was one of the most important parts of the implementation. The rewards need to encourage useful behavior without forcing the agents to follow a completely scripted solution. The robber reward functions were inherited from the previous project, with adjustments depending on whether the decoy mechanic is enabled. The robbers receive rewards for collecting valuables and reaching the goal, and penalties for being detected, triggering the alarm, failing to enter the museum, or leaving a teammate behind.

The new part of the reward design is the guard reward function. The final guard reward function is summarized in Table 3.3. It was designed to encourage active defensive behavior without manually scripting the final policy. A guard should not simply stand still, move randomly, or rely only on the camera system to stop the robbers. Instead, the reward structure encourages guards to move through their assigned room, detect robbers, protect the active valuable, and contribute to the defensive team objective.

A guard receives a time penalty at each step. This discourages passive behavior and prevents the guard from receiving no consequence for doing nothing. Although this penalty is small, it creates pressure for the guard to find useful actions during the episode. This is important because without any step penalty, a guard could remain inactive while still occasionally benefiting from events caused by other defensive systems.

A guard also receives a reward when it detects a robber. In the final implementation, the detection reward is calculated using the detection reward scale divided by the maximum number of guard field of view steps required to trigger the alarm. With a detection reward scale of 1.5 and a guard detection threshold of 100 steps, this gives a reward of 0.015 for each timestep in which a robber is detected. This encourages guards to position and rotate

themselves in ways that increase the chance of detecting robbers through the field of view and raycast system.

To avoid guards learning to stand still, a patrol reward was added. The room is divided into grid cells, and a guard receives a small reward when it enters a cell it has not visited during the current episode. The patrol reward is 0.005 for each new cell, with a maximum patrol reward of 0.4 per episode. This encourages exploration of the assigned area while preventing the guard from optimizing only for unlimited movement reward. The reward cap is important because without it, the guard could focus too much on movement and not enough on the actual defensive objective.

A valuable protection penalty was also added. Since each room contains one active valuable, each guard is assigned the active valuable in its own room. If that valuable is stolen, the guard responsible receives a penalty of -1.0. This encourages the guard to protect the room objective rather than only moving randomly or chasing detection rewards. It also connects each guard more directly to the room it is assigned to defend.

The guard reward function also includes team level defensive rewards. When the guard team wins, all guards receive a positive reward. In addition, the final guard that detected the robber before the alarm outcome receives an extra bonus. This helps connect individual detection behavior to the overall team objective. Without this type of credit, a guard that contributes directly to the alarm may not receive enough feedback for its role in stopping the robbers.

Table 3.3: Final reward function used for learned guard training.

Reward Component	Value	Applied To
Time penalty	-0.001 per step	Individual guard
Robber detection	+0.015 per detected step	Detecting guard
New patrol cell	+0.005	Individual guard
Patrol reward cap	0.4 per episode	Individual guard
Valuable stolen	-1.0	Responsible guard
Guard team win	+1.0	All guards
Final detecting guard bonus	+0.5	Last detecting guard

3.7 Training Procedure

The experiments were trained using Unity ML-Agents and the PPO trainer. The project uses a build of the Unity environment for faster training and can run multiple environment instances in parallel. Parallel training allows more experience to be collected in less real time.

The first experiment trains the guard policy while the robber policies remain fixed. The robbers are set to inference mode using the pretrained Curriculum II models. The guards are set to training mode with no model assigned. This experiment creates the learned guard baseline. During this experiment, the decoy system is disabled to keep the robber observation and action spaces compatible with the pretrained Curriculum II models.

After training the guard policy, the exported guard model is tested in Unity by setting the guards to inference mode. The pretrained robbers are also kept in inference mode. The purpose of this evaluation is to measure how well the original robbers perform against the learned guards.

The second experiment trains or adapts the robber agents against the learned guard policy. In this setup, the learned guard model is fixed in inference mode, while the robbers are trained from their Curriculum II checkpoints. The decoy system remains disabled. This allows the robber team to adapt to the new defensive behavior without changing its original action space.

The third experiment enables the decoy mechanic. In this setup, the decoy system is enabled for both robbers and guards. Because the decoy mechanic changes the robber observation and action spaces, the previous robber checkpoints cannot be reused directly. Therefore, the robber agents are trained without initialization in the decoy experiment, while the guard policy can be initialized from the learned guard baseline. This experiment is intended to test whether the decoy mechanic can introduce deception-based behavior into the adversarial environment.

3.8 Data Collection and Evaluation Metrics

During training, Unity ML-Agents records cumulative reward and reward standard deviation through Tensor Board. These graphs are used to observe whether training is progressing and whether the policy is receiving meaningful reward signals. However, reward alone is not enough to evaluate the behavior, because the goal of the project is not only to maximize reward but also to compare team success.

For this reason, additional data is written to CSV files during evaluation episodes. Each completed episode stores information such as whether the alarm was triggered, whether the robbers won, whether the loss was caused by cameras or guards, how many valuables were collected, the final rooms reached by the robbers, and the episode duration.

The main metric used for evaluation is robber win rate. A robber win occurs when the robber team successfully completes the episode objective. A guard win is counted when the robbers fail to win, including alarm losses, camera losses, and timeouts. Additional metrics include the average number of collected valuables and the average episode length.

3.9 Implementation Difficulties and Failed Attempts

Several difficulties arose during the implementation. One major issue was that policies can become inactive if the reward signal is too weak or sparse. During early guard training, some exported guard models produced little or no movement in inference mode. To debug this, the guards were tested in heuristic mode and in visible training mode. This confirmed that the movement code worked correctly and that the issue was related to the learned policy rather than the physics implementation.

Another issue was caused by observation and action space compatibility. The pretrained robber models require the same observation and action spaces used during their original training. Adding a new decoy action changes the action space and can make old models unusable. To avoid this problem, the decoy action was disabled during baseline experiments using the old robber models. The decoy mechanic is only enabled in experiments where the robber policy is trained with the modified action space.

A further issue was related to guard rewards. Initially, the guards did not receive enough useful feedback from the environment. Since camera detections or robber failures could happen without meaningful guard behavior, the guard policy sometimes received almost no learning signal. To address this, patrol rewards, detection rewards, valuable protection penalties, and alarm outcome rewards were used. These rewards encourage guards to move around their assigned rooms, detect robbers, care about the valuable they are supposed to protect, and contribute to the team defensive objective.

These difficulties were important because they affected the reliability of the experiments. They also show that training agents in a multi-agent Unity environment requires not only designing rewards and observations, but also carefully checking inference behavior, action space compatibility, data logging, and the difference between training behavior and exported ONNX behavior.

Chapter 4

Results and Analysis

4.1 Evaluation Method

The experiments in this thesis were evaluated using both training statistics and episode outcome data. During training, the Unity ML-Agents Toolkit recorded the cumulative reward of each trained behavior. These values were visualized using TensorBoard and were used to monitor whether the agents were receiving meaningful reward signals during training. In addition to the mean cumulative reward, the standard deviation of the reward was also observed, since a reward standard deviation close to zero can indicate that most episodes are ending in a very similar way.

However, the reward alone is not enough to evaluate the success of the agents. The purpose of the project is to compare the performance of the robber and guard teams under different training conditions. For this reason, additional episode data was collected in CSV files during evaluation. Each completed episode stored information about the outcome of the episode, including whether the alarm was triggered, whether the robbers won, whether the loss was caused by cameras or guards, how many valuables were collected, and how long the episode lasted.

The main evaluation metric used in this thesis is the robber win rate. A robber's win is counted when the robber team successfully completes the heist objective. A guard win is counted when the robbers fail to win, either because the alarm is triggered, because the cameras detect them, or because the episode reaches the maximum time limit. This definition allows the experiments to be compared directly, since each episode is assigned either to the robber team or to the defensive team.

4.2 Experiment I: Pretrained Robbers Against Learned Guards

The first experiment evaluates whether the previously trained Curriculum II robber policies can still succeed against learned guards. In the original version of the environment, the robbers were trained against scripted guards. Therefore, this experiment tests whether the learned robber strategy generalizes to a new defensive opponent.

In this experiment, the robber agents were kept fixed. The Locksmith and Technician used the pretrained Curriculum II models. The decoy mechanic was disabled, so the robber action space remained compatible with the original trained models. The guards were trained using PPO and then evaluated in inference mode. The guard policy was trained to patrol, detect robbers, protect valuables, and trigger the alarm after the detection threshold was reached.

The expected outcome of this experiment was that the pretrained robbers would perform worse against learned guards than against the original scripted guards. This is because the robbers were not trained against this new defensive policy. If the learned guards move differently or protect valuables more effectively, the robber policy may fail to generalize.

The learned guard baseline was trained using PPO, with the main configuration summarized in Table 4.1. The Curriculum II robber policies were fixed in inference mode, while the guard agents were trained under the shared behavior name Guard. The final guard architecture used a vector observation size of 20, two continuous movement actions, and two discrete action branches. Robber information was provided only when the robber was detected through the guard field of view and raycast system, preserving partial observability. The final guard field-of-view angle was 150 degrees. The reward function combined a timestep penalty, patrol cell exploration rewards, rewards for detecting robbers, a penalty when the protected valuable was stolen, and positive rewards when the guard team successfully stopped the robbers.

Table 4.1: Learned guard baseline configuration.

Parameter	Value
Guard training run ID	baseline
Training steps	approximately 10 million
Number of parallel environments	4
Final checkpoint used	exported Guard ONNX model
Evaluation episodes	100
Guard vector observation size	20
Guard continuous actions	2
Guard discrete branches	2
Robber policies	Curriculum II pretrained models
Decoy mechanic	disabled
max_steps_in_fov_guard	100
max_steps_in_fov_camera	200
max_time_alarm	100

4.2.1 Training Reward

During training, the guard reward was monitored to check whether the policy was receiving useful learning signals. At the beginning of development, several training attempts collapsed to near zero reward. These early failures were caused by setup and reward design issues, including incorrect field of view layer masks, mismatched observation sizes, and duplicate Behavior Parameters components. After these issues were corrected, the final training run did not collapse.

In the final baseline run, the guard policy used a reward structure that encouraged exploration, detection, valuable protection, and alarm success. The patrol reward encouraged guards to move through new areas of their assigned room, while the detection reward encouraged them to position and rotate themselves in ways that allowed robber detection through the field of view and raycast system. The valuable protection penalty discouraged guards from ignoring the active valuable in their room.

The reward curve showed that the model continued to receive nonzero and varied rewards during training. This indicates that the policy did not collapse into a completely inactive

strategy and that episodes continued to produce varied outcomes during training. The training reward therefore suggests that the learned guard baseline was stable enough to be used for evaluation.

4.2.2 Evaluation Results

The learned guard baseline was evaluated for 100 episodes against the fixed Curriculum II robber policies. The main evaluation results are shown in Table 4.2. The robbers did not win any of the evaluation episodes, while the defensive team won all 100 episodes. This gives robbers a win rate of 0% and a guard win rate of 100%. An alarm was triggered in every episode.

Most recorded alarm outcomes were associated with guard detection rather than camera detection. Guard alarm time was recorded in 95% of episodes, while camera alarm time was recorded in 5% of episodes. The robbers collected an average of 1.11 valuables per episode, meaning that they were still able to make partial progress inside the museum but were unable to complete the full heist. The average episode length was 1063.84 steps.

Table 4.2: Evaluation results for Experiment I.

Metric	Value
Evaluation episodes	100
Robber win rate	0%
Guard win rate	100%
Alarm triggered	100%
Guard alarm time recorded	95%
Camera alarm time recorded	5%
Average valuables collected	1.11
Average episode length	1063.84 steps
Decoy enabled	No

4.2.3 Analysis

The purpose of this experiment was to determine whether the original robber policies are robust to a change in defensive behavior. The results show that they are not robust to this change. The

pretrained Curriculum II robber policies achieved a 0%-win rate against the learned guard baseline. This supports the hypothesis that robbers trained against scripted guards may not generalize well to learned defensive agents.

The 100% defensive win rate shows that the learned guard policy created a strong baseline opponent. This does not mean that the guards learned perfect defensive behavior, but it does show that the robber strategy learned against scripted guards was no longer reliable. The robbers were still able to collect some valuables, with an average of 1.11 valuables per episode, but they always failed to escape before the alarm condition ended the episode.

The alarm source breakdown is especially important. Guard alarm time was recorded in 95% of episodes, compared with 5% for camera alarm time. This indicates that the learned guards were not only passive objects in the environment. They contributed directly to the defensive outcome in most runs. This is important because the goal of the experiment was to create a learned guard baseline, not simply to rely on the existing camera system.

Overall, Experiment I provides a strong baseline for the following experiments. The result shows that learned guards significantly reduce the success rate of the original robber policies. Therefore, the next experiments evaluate whether robber adaptation and the decoy mechanic improve performance against this learned defensive policy.

4.3 Experiment II: Robber Adaptation Against Learned Guards

The second experiment investigates whether the robber team can improve after additional training against the learned guard policy from Experiment I. In Experiment I, the original Curriculum II robbers achieved a 0%-win rate against the learned guard baseline. This suggested that the robber policies did not fully generalize from scripted guards to learned defensive agents. Experiment II therefore tests whether further training can help the robbers adapt to this new opponent.

In this experiment, the learned guard model from Experiment I was kept fixed in inference mode. The guard agents used the exported Guard.onnx model from the learned guard baseline, and the decoy system remained disabled. The Locksmith and Technician agents were set to training mode and initialized from the previous Curriculum II checkpoints. This means that Experiment II was an adaptation experiment rather than training the robber agents from the beginning.

The same guard and camera alarm parameters from Experiment I were used. The guard detection threshold was set to 100 steps, the camera detection threshold was set to 200 steps, and the alarm duration was set to 100 steps. The robber curriculum parameters also followed the same structure as Curriculum II, beginning with more forgiving detection thresholds and gradually reducing them across lessons. This allowed the robbers to adapt to the learned guards while keeping the training structure consistent with the previous robber curriculum.

The purpose of this experiment is to determine whether the poor performance observed in Experiment I is permanent, or whether the robber team can recover performance when exposed to the learned guard policy during training. This experiment also provides a direct comparison between generalization without adaptation and performance after opponent specific training. The setup used for this experiment is summarized in Table 4.3.

Table 4.3: Robber adaptation experiment configuration.

Parameter	Value
Guard model	Learned Guard model from Experiment I
Guard behaviour type	Inference Only
Robber behaviour type	Default
Robber initialisation	Curriculum II checkpoints
Decoy mechanic	Disabled
Guard detection threshold	100 steps
Camera detection threshold	200 steps
Alarm duration	100 steps
Robber curriculum	Curriculum II-style adaptation
Initial robber FOV curriculum threshold	300 steps
Final robber FOV curriculum threshold	10 steps
Number of parallel environments	4
Training steps	approximately 10 million
Evaluation episodes	100

4.3.1 Training Reward

During training, the reward trends of the Locksmith and Technician were monitored separately

because the two agents use different policies and have different responsibilities. The Locksmith is responsible for opening gates and helping the team progress through the museum, while the Technician is responsible for disabling security cameras. Since the learned guard policy changes the defensive pressure in the environment, both agents must adjust their behavior compared with the original Curriculum II training.

At the beginning of adaptation training, the robbers faced difficulty because the learned guards behaved differently from the scripted guards used in their previous training. The reward curves were therefore expected to be unstable during the early stages of the experiment. This instability is expected in an adversarial setting, since the robbers must relearn how to move safely, collect valuables, and reach the goal while avoiding a new defensive policy.

The reward trends were useful for identifying whether adaptation was taking place. In this experiment, the final evaluation suggests that the robbers did learn to survive longer against the learned guards, even though this did not translate into a robber win. This indicates that adaptation changed the robber's behavior, but was not sufficient to solve the full heist objective under the learned guard baseline.

4.3.2 Evaluation Results

After training, the adapted robber policies were evaluated against the same learned guard model used in Experiment I. The goal was to compare the original Curriculum II robber policies with the adapted robber policies under the same defensive conditions. The comparison between Experiment I and Experiment II is shown in Table 4.4.

The adapted robbers again achieved a 0%-win rate, while the defensive team won all 100 episodes. However, the average episode length increased from 1063.84 steps in Experiment I to 1422.73 steps in Experiment II. This suggests that the adapted robbers survived longer against the learned guards, even though they still failed to complete the full heist. The average number of valuables collected decreased from 1.11 to 0.88, indicating that longer survival did not necessarily lead to more successful stealing behavior.

All recorded alarms in Experiment II were associated with guard detection. Guard alarm time was recorded in 100% of episodes, while camera alarm time was recorded in 0% of episodes.

Table 4.4: Comparison between Experiment I and Experiment II.

Metric	Experiment I	Experiment II
Evaluation episodes	100	100
Robber win rate	0%	0%
Guard win rate	100%	100%
Alarm triggered	100%	100%
Guard alarm time recorded	95%	100%
Camera alarm time recorded	5%	0%
Average valuables collected	1.11	0.88
Average episode length	1063.84 steps	1422.73 steps
Decoy enabled	No	No

4.3.3 Analysis

The main comparison in this experiment is between the original Curriculum II robbers and the adapted robbers. Experiment I showed that the original robber policies struggled against the learned guard baseline. Experiment II shows that additional training against the learned guard policy did not recover robber wins, since the robber win rate remained 0%. However, the increase in average episode length suggests that the robbers were able to survive for longer after adaptation.

This result indicates that adaptation had a partial effect. The robbers did not learn a complete successful strategy against the learned guards, but they appeared to have changed their behavior enough to delay episode termination. This may mean that the agents learned to avoid immediate detection more effectively or moved more cautiously through the environment. However, this improvement did not translate into higher completion of the heist objective.

The decrease in average valuables collected from 1.11 to 0.88 also suggests a tradeoff. The adapted robbers may have become more conservative, surviving longer but collecting fewer valuables. In adversarial environments, this type of behavior is possible when agents learn to avoid danger without also learning an effective path to victory.

The alarm breakdown further shows that the learned guards remained the dominant defensive factor. Guard alarm time was recorded in all episodes, while camera alarm time was not recorded. This means that, despite longer survival, the adapted robbers were still ultimately stopped by the learned guard policy. Therefore, Experiment II supports the idea that opponent specific adaptation can change behavior, but it also shows that adaptation alone was not sufficient to overcome the learned guard baseline.

4.4 Experiment III: Decoy Mechanic

The third experiment investigates the effect of adding the decoy mechanic to the museum heist environment. Unlike Experiment I and Experiment II, where the decoy system was disabled, this experiment enables decoys and trains the agents with the new mechanic available. The purpose is to examine whether deception can create a new attacker defender adaptation problem between robbers and guards.

In this experiment, all three behaviors were trainable: Guard, TechGuy, and Locksmith. The guard policy was initialized from the learned guard baseline trained in Experiment I, while the robber agents were trained from scratch. This was necessary because enabling the decoy mechanic changed the robber observation and action spaces, making the old Curriculum II robber checkpoints incompatible with the new architecture.

The decoy mechanic gives the robber agents an additional action that can be used to throw a decoy. The purpose of the decoy is to distract or influence guards, potentially creating opportunities for the robbers to move, collect valuables, or escape more safely. At the same time, guards receive decoy related observations and can learn whether to investigate the decoy or continue focusing on robber detection. This creates a more complex interaction between the two teams.

The experiment used the same main alarm parameters as the previous experiments, and the full setup is summarized in Table 4.5. The guard detection threshold was set to 100 steps, the camera detection threshold was set to 200 steps, and the alarm duration was set to 100 steps. The robber curriculum parameters began at 300 steps for both robber agents. The training run used four parallel environments and a time scale of 20.

Table 4.5: Decoy co-training experiment configuration.

Parameter	Value
Run file	config/experiment_II.yaml
Run ID	experiment_II
Thesis experiment	Experiment III: Decoy Co-Training
Trainable behaviours	Guard, TechGuy, Locksmith
Guard initialisation	Learned guard baseline checkpoint
TechGuy initialisation	None
Locksmith initialisation	None
Decoy mechanic	Enabled
Guard vector observation size	20
TechGuy vector observation size	59
Locksmith vector observation size	58
Guard continuous actions	2
Robber continuous actions	2
Guard discrete branches	2
Robber discrete branches	4
Guard detection threshold	100 steps
Camera detection threshold	200 steps
Alarm duration	100 steps
Number of parallel environments	4
Time scale	20
Training steps	approximately 10 million
Evaluation episodes	100

4.4.1 Training Reward

The decoy co-training run started successfully, and ML-Agents connected to all three behaviors: Guard, TechGuy, and Locksmith. The guard checkpoint from Experiment I loaded correctly, while the TechGuy and Locksmith agents were trained without initialization because the new decoy action and observation spaces were incompatible with their previous checkpoints.

The guard behavior and robber behaviors accumulated steps at different rates because ML-Agents counts steps separately for each behavior. Since there are multiple guards in the environment and only one agent for each robber role, the shared guard behavior collects experience faster than each individual robber's behavior. This does not mean that the agents were trained in separate environments, they still interacted in the same episodes, but their experience counters were reported separately.

The reward trends showed that the decoy co-training setup was functional. However, because the robbers were trained from scratch with an expanded action space, learning the decoy strategy was more difficult than the non-decoy adaptation experiment. The robbers had to relearn movement, stealing behavior, avoidance, and decoy usage at the same time, while the guard policy was already initialized from a learned baseline.

4.4.2 Evaluation Results

The decoy co-training experiment was evaluated for 100 episodes, with the main results shown in Table 4.6. As in the previous experiments, the robber win rate remained 0% and the guard win rate remained 100%. This means that the decoy mechanic did not produce successful robber escapes in the final evaluation.

However, the alarm source distribution changed. Guard alarm time was recorded in 70% of episodes, while camera alarm time was recorded in 31% of episodes. These values add to 101% because one episode had both guard and camera alarm times recorded. The important interpretation is that guard triggered alarms decreased compared with the previous experiments, while camera-related alarms became more frequent.

The average number of valuables collected decreased to 0.37, while the average episode length was 1374.84 steps. This means that robbers survived longer than in Experiment I but collected fewer valuables. The results suggest that the decoy system affected defensive behavior, especially guard interaction, but did not allow the robbers to complete the heist.

Table 4.6: Evaluation results for Experiment III.

Metric	Value
Evaluation episodes	100
Robber win rate	0%
Guard win rate	100%
Alarm triggered	100%
Guard alarm time recorded	70%
Camera alarm time recorded	31%
Average valuables collected	0.37
Average episode length	1374.84 steps
Decoy enabled	Yes

4.4.3 Analysis

The purpose of this experiment was to test whether the decoy mechanic could introduce a useful deception-based strategy into the museum heist environment. The experiment successfully integrated the decoy system into both robber and guard learning. Guards received decoy related observations, while robbers received additional observations and an additional discrete action branch for throwing decoys.

The final evaluation shows that the decoy mechanic did not improve the robber win rate. The robbers still achieved 0% wins across 100 evaluation episodes. However, the alarm source distribution changed noticeably. Guard alarm time dropped from 95% in Experiment I and 100% in Experiment II to 70% in Experiment III. This suggests that decoys may have affected the guards' direct ability to stop the robbers or changed the way guards interacted with the environment.

At the same time, camera alarm time increased to 31%. This indicates that even if decoys distracted guards or reduced direct guard interception, the robbers still failed to manage the full defensive system. In other words, reducing guard pressure was not enough to complete the heist because cameras remained a major obstacle. The lower average number of valuables collected also suggests that training from scratch with the new decoy action space made robber learning more difficult.

This result should therefore be interpreted carefully. The decoy mechanic changed the behavior of the environment, but it did not produce a successful robber policy. It introduced a new strategic option but also increased the complexity of the learning problem. The robbers had to learn when to throw decoys, how to use the time created by distractions, and how to avoid cameras while still collecting valuables. The evaluation suggests that this was not fully learned within the available training set-up.

4.5 Comparison of Experiments

The three experiments are compared in Table 4.7 to evaluate how each training stage affects the robber team's performance. Experiment I tests generalization. Experiment II tests whether robbers can recover performance when trained against the learned guard policy. Experiment III tests whether a decoy mechanic can improve performance or change defensive outcomes.

Table 4.7: Final comparison of the three experiments.

Metric	Experiment I: Learned Guard Baseline	Experiment II: Robber Adaptation	Experiment III: Decoy Co-Training
Evaluation status	Completed	Completed	Completed
Evaluation episodes	100	100	100
Robber win rate	0%	0%	0%
Guard win rate	100%	100%	100%
Alarm triggered	100%	100%	100%
Guard alarm time recorded	95%	100%	70%
Camera alarm time recorded	5%	0%	31%
Average valuables collected	1.11	0.88	0.37
Average episode length	1063.84 steps	1422.73 steps	1374.84 steps
Decoy enabled	No	No	Yes

Experiment I shows that learned guards created a strong defensive baseline. The pretrained robbers achieved no wins, and most alarms were associated with guard detection. This supports the idea that robbers trained against scripted guards did not generalize well to learned guards.

Experiment II shows that adaptation changed robber behavior but did not recover a win rate. The average episode length increased from 1063.84 to 1422.73 steps, suggesting that the adapted robbers survived longer. However, the average number of valuables collected decreased, and guard alarm time was recorded in every episode. This indicates that the adapted robbers were still unable to overcome the learned guard policy.

Experiment III shows that the decoy mechanic changed the defensive outcome distribution but did not improve final robber's success. Guard alarm time decreased to 70%, while camera alarm time increased to 31%. This suggests that decoys may have reduced direct guard interception or changed guard behavior, but the robbers still failed to escape and were often stopped by the broader defensive system, especially cameras.

Overall, the final comparison shows that none of the experiments produced robber wins against the learned defensive setup. However, the experiments still provide useful information. The learned guards strongly changed the difficulty of the environment, robber adaptation increased survival time, and the decoy mechanic changed the alarm source distribution. These results show that the proposed changes affected agent behavior, even when they did not produce successful heist completions.

4.6 Analysis

The results must be interpreted in relation to the original problem of the thesis. The main question is whether learned defensive agents create a more difficult and adaptive opponent than scripted guards. The result of Experiment I strongly supports this idea. The pretrained robbers achieved 0%-win rate against learned guards, suggesting that their original policy did not fully generalize beyond the scripted defensive behavior.

The second question is whether robbers can improve when trained against learned guards. Experiment II shows partial improvement, but not in the form of final wins. The adapted robbers survived longer on average, which suggests that additional training against the learned

guard policy changed their behavior. However, they still achieved 0%-win rate and collected fewer valuables on average. This means adaptation helped with survival but did not solve the full heist objective.

The third question is whether the decoy mechanic improves performance. Experiment III shows that decoys did not improve robber win rate, but they did change the distribution of alarm sources. Guard alarm time decreased compared with the previous experiments, while camera alarm time increased. This suggests that the decoy mechanic influenced guard-related outcomes, but the robbers still failed to complete the full task because the overall defensive system remained too difficult.

It is also important to consider the limitations of the results. Training time was limited, and the policies may not have fully converged. The learned guard reward function required tuning, and some early models learned inactive behavior. Furthermore, the environment contains both guards and cameras, so some robber losses may be caused more by cameras than by the learned guard policy. The decoy experiment also introduced a larger robber action space, and the robbers had to train from scratch because their old checkpoints were incompatible with the new decoy action and observation spaces.

Overall, the chapter shows that the proposed extensions changed the behavior and performance of the agents. The learned guards created a much stronger defensive baseline than scripted guards. Robber adaptation increased survival time but did not recover a win rate. The decoy mechanic changed the balance between guard and camera alarms but did not lead to successful robber escapes. These results connect directly to the initial motivation of the thesis: creating NPC behavior that is less predictable than scripted logic and testing whether agents trained against one opponent can adapt to another.

Chapter 5

Discussion

5.1 Interpretation of the Results

The aim of this thesis was to investigate how a museum heist environment changes when the defensive team is no longer controlled only by scripted behavior. The original version of the environment trained the robber agents against scripted guards. In this work, the guards were converted into learned agents, and the robber team was evaluated against this new defensive policy. The results therefore need to be interpreted in relation to the main question of the thesis: whether learned defensive behavior creates a more difficult and adaptive opponent for the robbers.

The results show that replacing scripted guards with learned guards significantly changed the difficulty of the environment. In Experiment I, the pretrained Curriculum II robbers achieved a 0%-win rate against the learned guard baseline. This indicates that the original robber policies did not generalize successfully to the new defensive behavior. The robbers were still able to collect some valuables, but they could not complete the full heist.

Experiment II showed that adaptation changed robber behavior but did not recover final success. The adapted robbers also achieved a 0%-win rate, but their average episode duration

increased compared with Experiment I. This suggests that additional training against the learned guard policy helped the robbers survive for longer. However, this survival improvement did not translate into more robber wins or more collected valuables.

Experiment III showed that the decoy mechanic changed the defensive outcome distribution but again did not improve the robber win rate. The decoy enabled robbers achieved 0% wins, but guard alarm time decreased compared with the previous experiments, while camera alarm time increased. This suggests that decoys affected guard related outcomes, but the robbers still failed to manage the full defensive system. In other words, reducing direct guard pressure was not enough to complete the heist because cameras remained a major obstacle.

The results should therefore be interpreted as partial behavioral improvements rather than full task success. The learned guards created a stronger defensive baseline. Robber adaptation increased survival time. The decoy mechanic changed how the defensive system stopped the robbers. However, none of the experimental conditions allowed the robber team to successfully recover a positive win rate.

5.2 Learned Guards and Generalization

The first major observation of this project is related to generalization. The pretrained robber agents were originally trained against scripted guards. These guards followed fixed defensive behavior, which made them predictable over repeated episodes. When defensive behavior was replaced with learned guards, the robbers faced a different type of opponent.

The learned guards reduced the robber win rate to 0%. This indicates that the original robber policy was specialized to the scripted defensive behavior. This is an important result because it shows that success against a fixed opponent does not guarantee success against a learned opponent. In adversarial environments, the behavior of the opponent is part of the task. When the opponent changes, the effective environment also changes.

This result is related to the wider problem of robustness in reinforcement learning. A policy can achieve strong performance in the environment where it was trained but still fail when some aspects of the environment change. In this project, the change was not the map or the robber's abilities, but the behavior of the guards. This makes the experiment useful for studying how opponent behavior affects learned policies.

The learned guards also changed the role of the defensive team. Instead of being only scripted obstacles, the guards became active agents with their own reward function. This made training more difficult, because the guard policy needed meaningful rewards to avoid inactive behavior. Early training attempts showed that if the guard reward was too weak or sparse, the learned policy could collapse into behavior that produced little movement. The addition of patrol rewards, detection rewards, valuable-protection penalties, and alarm outcome rewards helped produce active defensive behavior.

5.3 Robber Adaptation

The second major part of the thesis was to examine whether robbers could improve after being trained against the learned guard policy. This is important because an adversarial environment should not only test whether an old policy fails, but also whether the agent can adapt to the new opponent.

Experiment II did not improve the robber win rate, since the robber team still achieved 0% wins. However, the average episode length increased compared with Experiment I. This suggests that the robbers adapted in a limited way. They survived longer against the learned guards, but they did not learn a complete strategy for stealing valuables and escaping.

This result suggests a possible tradeoff in the adapted robber behavior. The robbers may have become more cautious, avoiding direct or early detection, but this caution did not produce better completion of the heist objective. The lower average number of valuables collected in Experiment II supports this interpretation. Surviving longer is useful, but in this environment the robber team must also continue making progress through the museum.

The adaptation experiment therefore contributes to understanding the interaction between offensive and defensive learning. It shows that training against a learned opponent can change behavior, but it also shows that adaptation alone may not be enough when the defensive policy is strong. More training, improved robber rewards, or staged curriculum design may be necessary for the robbers to convert longer survival into successful heist completions.

5.4 Decoy Mechanic

The decoy mechanic was introduced to investigate whether deception can improve robber performance. In stealth games, distraction mechanics are common because they allow a player to manipulate enemy attention. A decoy can create an opening by pulling a guard away from a valuable, a doorway, or the robber's path. In this project, the decoy mechanic gives the robber team an additional strategic action beyond movement, gate opening, camera disabling, and valuable collection.

The final decoy experiment did not improve the robber win rate. The robber team again achieved 0% wins. However, the alarm source distribution changed. Guard alarm time decreased compared with the previous experiments, while camera alarm time increased. This suggests that the decoy mechanic affected the guard side of the environment, but the robbers did not use that effect well enough to complete the heist.

This result is important because it shows that adding a new mechanic does not automatically improve performance. The decoy increased the complexity of the learning problem. The robber agents had to learn not only that the decoy existed, but also when to use it, where to place it, and how to take advantage of the time it created. At the same time, the robbers had to continue avoiding guards and cameras, collecting valuables, and moving toward the goal.

Another challenge is that the decoy mechanic changed the robber observation and action spaces. This made the old Curriculum II robber checkpoints incompatible, so the robbers had to be trained from scratch in the decoy experiment. This created a difficult training situation: the guard policy started from a learned baseline, while the robbers had to relearn basic heist behavior with an expanded action space.

For this reason, the decoy experiment should not be interpreted as proof that deception mechanics are ineffective. Instead, it shows that deception-based actions require careful training design. A more suitable future approach would be to train robbers with decoys first against easier guards, then gradually introduce learned guards once basic decoy usage has been learned.

5.5 Comparison with Previous Work

The previous museum robbery project focused mainly on training the robber team against scripted guards. Its main result was that the Locksmith and Technician could learn cooperative behavior using different policies and role-specific abilities. The current thesis

builds on that work by changing the defensive side of the environment. Instead of treating guards as fixed scripted obstacles, this work trains the guards using reinforcement learning.

This difference changes the interpretation of the environment. In the previous version, the key question was whether the two robber agents could cooperate. In this thesis, the key question is whether that cooperation remains effective when the opponent becomes learned and less predictable. Therefore, the focus moves from only cooperative learning to adversarial adaptation.

The results of this thesis also relate to previous multi-agent reinforcement learning work such as AlphaStar, OpenAI Five, and Hide-and-Seek. Those systems showed that agents can improve through competition, self-play, and exposure to changing opponents. The museum heist environment is much smaller, but it studies a similar idea in a role based stealth scenario. The agents do not all share the same abilities, and the two teams have different objectives. This makes the environment asymmetric and closer to many game scenarios where different characters have different roles.

Unlike largescale systems, this thesis uses a smaller Unity environment and limited computational resources. Therefore, the goal is not to reach human level game performance. Instead, the goal is to investigate how changing scripted opponents into learned opponents affects agent behavior and evaluation outcomes in a controlled setting.

This thesis also adds more direct episode level evaluation compared with relying only on cumulative reward curves. The final evaluation includes win rate, alarm source, valuables collected, and episode duration. These metrics make it easier to identify different types of partial progress, such as surviving longer or changing the alarm source distribution, even when the final win rate does not improve.

5.6 Limitations

The results of this thesis have several limitations. The first limitation is training time. Multi-agent reinforcement learning in a Unity environment can require many training steps, especially when both teams interact in a complex environment. Due to limited time and computational resources, some policies may not have fully converged.

A second limitation is sensitivity of the reward. The learned behavior depends strongly on the

reward function. During the implementation, some guard policies received very low or nearly constant rewards, which resulted in poor inference behavior. This required reward shaping through patrol rewards, detection rewards, and valuable protection penalties. Although these rewards were useful, they also influenced the final behavior of the guards.

A third limitation is that the cameras and guards both contribute to robber losses. In some cases, robbers may lose because of cameras rather than because of learned guard behavior. This can make it difficult to isolate the exact effect of the guard policy. Future experiments could evaluate guards and cameras separately to better understand their individual contributions.

A fourth limitation is action space compatibility. The pretrained robber models could only be used when the observation and action spaces matched their original training setup. Adding the decoy mechanic changed the action space, which meant that old models could not always be reused directly. This made the decoy experiment more difficult and required separate training.

Finally, the environment is a simplified research scenario. Although it includes multiple rooms, agents, guards, cameras, gates, and valuables, it is still much smaller than a commercial stealth game. The results should therefore be interpreted as evidence from a controlled experimental environment, not as a complete solution for general game AI.

Chapter 6

Conclusion

The aim of this thesis was to extend a museum heist environment into a more adversarial multi-agent reinforcement learning scenario. In the previous version of the project, the robber team was trained using reinforcement learning while the guards followed scripted behavior. This thesis focused on modifying the defensive side of the environment by introducing learned guard agents and then studying how this affected the performance and behavior of the robber team.

The first objective was to train guards that could move, patrol, detect robbers, protect valuables, and trigger the alarm. This was implemented using the Unity ML-Agents Toolkit and the Proximal Policy Optimization algorithm. The guards were given local observations and a reward function that encouraged defensive behavior. During development, it became clear that guard reward design was important. If the reward signal was too weak or sparse, the learned policy could become inactive. For this reason, additional reward components, such as patrol rewards, detection rewards, valuable protection penalties, and team outcome rewards, were introduced to encourage more useful defensive behavior.

The second objective was to evaluate whether robber agents trained against scripted guards could still succeed against learned guards. This experiment tested the generalization of pretrained robber policies. The results showed that the pretrained robbers achieved a 0%-win rate against the learned guard baseline. This supports the idea that the robber policy had adapted to the scripted guards' behavior and did not fully generalize to a learned defensive opponent.

The third objective was to investigate whether the robber team could improve after being trained against the learned guard policy. The adapted robbers also achieved a 0%-win rate, so adaptation did not recover final task success. However, the average episode length increased, suggesting that the robbers survived longer against the learned guards. This indicates that opponent-specific training affected robber behavior, but was not sufficient to produce successful heist completions.

The final objective was to introduce a decoy mechanic as an additional strategic action for the robber team. The purpose of the decoy was to allow the robbers to distract or mislead guards, creating a deception-based strategy. The results showed that the decoy mechanic did not improve the robber win rate, but it changed the alarm source distribution. Guard alarm time decreased, while camera alarm time increased. This suggests that the decoy affected guard-related outcomes, but the robber team still failed to manage the full defensive system.

Overall, this thesis contributes to the original museum robbery project by moving from a setting with learned robbers and scripted guards to a setting where the defensive side can also learn. This makes the environment more dynamic and creates a stronger test for the robber policies. The project also shows that training learned NPCs in an adversarial environment is sensitive to reward design, observation design, opponent strength, and compatibility between trained models and action spaces.

The findings of this work are relevant to game AI because they show how reinforcement learning can be used to create more adaptive NPC behavior. Even though the environment is simplified compared to a commercial game, it includes important elements of stealth and team-based gameplay, such as guards, cameras, partial observability, cooperation, detection, and deception. These elements make it useful as a small research environment for studying adversarial NPC behavior.

Future work could train both teams together using a more formal self-play or league-based approach. This would allow the guards and robbers to continuously adapt to each other instead of training one side against a fixed opponent. Future experiments could also compare shared policies against separate role specific policies, evaluate the effect of removing cameras or guards separately, and expand the decoy mechanic with richer distraction behaviors. Finally, the environment could be extended with more room layouts, more types of guards, and additional stealth mechanics.

In conclusion, this thesis provides a step toward more adaptive adversarial NPC behavior in a museum heist scenario. By training guards, evaluating robber generalization, and introducing a decoy mechanic, the project demonstrates how reinforcement learning can be used not only to create cooperative agents, but also to study the interaction between opposing teams with different roles and objectives.

Bibliography

1] R.S. Sutton and A.G. Barto, Reinforcement Learning: An Introduction, 2nd ed., MIT Press, Cambridge, MA, 2018.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347, 2017.

[3] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar and D. Lange, "Unity: A General Platform for Intelligent Agents," arXiv preprint arXiv:1809.02627, 2018.

[4] E. Evripidou, "Multi-Agent Reinforcement Learning: Collaborative Agents in a Museum Robbery," MSc Thesis, University of Cyprus, 2023.

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

[6] L. Busoniu, R. Babuska and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 38, no. 2, pp. 156-172, 2008.

[7] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," Advances in Neural Information Processing Systems, vol. 30, 2017.

[8] O. Vinyals, I. Babuschkin, W.M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D.H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J.P. Agapiou, M. Jaderberg and D. Silver, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," Nature, vol. 575, pp. 350-354, 2019.

[9] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Jozefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H.P.O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski and S. Zhang, "Dota 2 with Large Scale Deep Reinforcement Learning," arXiv preprint arXiv:1912.06680, 2019.

[10] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew and I. Mordatch, "Emergent Tool Use from Multi-Agent Autocurricula," arXiv preprint arXiv:1909.07528, 2019.

[11] I. Millington and J. Funge, Artificial Intelligence for Games, 2nd ed., CRC Press, Boca Raton, FL, 2009.