Integrating Agent Modelling in Multi-Agent Reinforcement Learning Algorithms

Rahat Santosh

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2024

Abstract

In multi-agent systems, agents often rely on implicit modelling, where they infer the behaviours of others through indirect observation, which can result in slower learning and suboptimal performance. However, explicit agent modelling—where agents directly anticipate and adapt to the strategies of their peers—has the potential to significantly enhance learning efficiency and outcomes. This dissertation explores the potential of explicit agent modelling within Multi-Agent Reinforcement Learning (MARL), with a focus on autoencoder-based techniques designed to capture and predict opponent behaviours with greater precision.

The study introduces autoencoder-based agent modeling as a method to balance the complexities of policy reconstruction with the limitations of classification-based approaches, while providing greater control over the agent modeling process. By encoding and reconstructing key elements of opponent strategies—such as action frequencies, rewards, and observations—this approach provides a structured and adaptable framework for understanding and responding to other agents' strategies. Through systematic experimentation across a variety of environments, from cooperative tasks to intricate competitive scenarios, the research evaluates the impact of these modelling techniques on the performance of MARL algorithms.

This dissertation demonstrates the potential of autoencoder-based methods for agent modelling in complex environments. A comprehensive evaluation reveals that when these techniques are carefully tailored with appropriate reconstruction targets and latent dimensions, they have the potential to enhance MARL algorithms. By accurately capturing both short-term and long-term opponent strategies, agent modelling can lead to potential improvements in learning outcomes. However, balancing model complexity with computational efficiency is crucial. The findings provide valuable insights for future research, emphasizing the importance of agent modelling in developing more robust, adaptable, and intelligent multi-agent systems.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics Committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Rahat Santosh*)

Acknowledgements

I am deeply grateful for the opportunity to pursue my Master's dissertation at the School of Informatics, University of Edinburgh. This project has been a significant stepping stone in my academic and professional journey, allowing me to delve into a fascinating area of research that has greatly contributed to my development.

I would like to express my sincere gratitude to my supervisor, Prof. Dr. Stefano Albrecht, for his invaluable guidance, support, and encouragement throughout this project. His insights and expertise have been instrumental in shaping the direction and success of my research. I am also thankful to the Autonomous Agents Research Group for providing me with the necessary resources, without which this work would not have been possible.

Additionally, I would like to extend my heartfelt appreciation to Lukas Schäfer, a PhD student at the School of Informatics, for his mentorship and assistance throughout the dissertation process. His guidance and willingness to share his knowledge have been immensely helpful.

I would also like to acknowledge the unwavering support and encouragement from my family and friends, who have been a constant source of motivation.

This dissertation marks a significant milestone in my academic career, and I am committed to applying the knowledge and experience gained from this work to future endeavours. I look forward to continued collaboration with the individuals I have had the privilege to work with during this project.

Sincerely,
Rahat Santosh

Place: Edinburgh
Date: 19/09/2024

Table of Contents

Chapter 1

Introduction

The advancement of Multi-Agent Reinforcement Learning (MARL) has opened new avenues for optimising decision-making processes where multiple agents interact within complex environments. Each agent in these settings must learn to adapt to the strategies and behaviours of others while pursuing its objectives. This complexity becomes particularly pronounced in scenarios where agents lack direct access to the policies of their peers, creating a need for sophisticated techniques that can predict and respond to the actions of others. This process, known as agent modelling or opponent modelling [1], involves creating models that anticipate the strategies of other agents, thereby enabling more effective decision-making in multi-agent systems.

Recent studies (Albrecht and Stone [1], Papoudakis, Christianos, and Albrecht [2], Baarslag, Hendrikx, Hindriks, *et al.* [3], Karpinskyj, Zambetta, and Cavedon [4]) have highlighted the potential benefits of integrating explicit agent modelling techniques into MARL algorithms, demonstrating improvements in sample efficiency, stability of learning, and overall performance in complex environments. However, the literature addressing the integration of agent modelling within the context of deep neural networks and modern MARL algorithms remains relatively limited; for instance, papers such as Papoudakis, Christianos, and Albrecht [2] that focus on the application of autoencoder networks to agent modelling assume that the opponent agents' policies are fixed or non-learning. Moreover, a significant portion of the literature, including Baarslag, Hendrikx, Hindriks, *et al.* [3] and Karpinskyj, Zambetta, and Cavedon [4], dedicated to agent modelling is from a game-theoretic or video game perspective, aiming to predict human player behaviour. Notably, there is a dearth of research that explores the consideration of dynamic learning opponents within a MARL context. This dissertation aims to bridge this gap by thoroughly investigating the impact of agent modelling on

MARL performance across a variety of environments, focusing on policy reconstruction and classification-based modelling techniques.

This study will explore the nuances of agent modelling by implementing autoencoder-based policy representations, which offer a modular and flexible approach to modelling opponent behaviour. The use of autoencoders allows for the generation of latent representations of opponent policies [2], which can be varied in complexity, thereby providing insights into the effectiveness of different levels of abstraction in opponent modelling. The methodology involves evaluating these techniques across a range of MARL benchmark tasks and assessing the trade-offs between model complexity and performance gains. The research will also address several key challenges in MARL, such as handling partial observability, non-stationarity, and the balance between computational overhead and performance improvements. By systematically analysing these aspects, the study aims to provide a comprehensive understanding of the conditions under which agent modelling enhances MARL and where it may introduce challenges.

The dissertation is structured to first establish a solid theoretical foundation in reinforcement learning and multi-agent systems, followed by an extensive review of the existing literature on agent modelling. This is followed by the methodology chapter, which details the proposed integration of autoencoder-based agent modelling into MARL algorithms and the experimental setup designed to test these integrations. The results of these experiments will be analysed to identify key insights and trends, which will then be discussed in the final chapters, culminating in recommendations for future research directions.

This study showcases the potential of autoencoder-based agent modelling in a multi-agent reinforcement learning (MARL) context. A comprehensive analysis of various ablations in the proposed agent modelling system reveals that incorporating action frequency and reward-based reconstruction leads to significant performance improvements. Furthermore, the study thoroughly examines the interplay between environment dynamics, algorithm configurations, and agent modelling configurations and their impact on episodic returns. This analysis offers valuable insights into the optimal settings for each hyperparameter configuration. Notably, the study highlights the substantial potential of autoencoder-based agent modelling in complex and strategic MARL environments, underscoring the importance of carefully selecting reconstruction targets, latent dimensions, and loss functions to enhance agent modelling effectiveness. Finally, a roadmap is presented for future research and to provide guidance on seamlessly integrating autoencoder-based modelling into MARL tasks tailored to specific task and

algorithm characteristics.

1.1 Research Objectives

The primary objectives of this study are to evaluate the effectiveness of agent modelling techniques in improving the episodic returns of MARL algorithms across various environments, to analyze the impact of different levels of abstraction in opponent modelling on the converged returns in MARL, to investigate the computational trade-offs associated with integrating agent modelling into standard MARL algorithms, and to explore the effects of environment characteristics, such as partial observability and agent heterogeneity, on the efficacy of agent modelling.

1.2 Contributions

This dissertation makes several key contributions: It provides a comprehensive literature review that synthesizes existing research on agent modeling in MARL, highlighting gaps and identifying potential areas for future exploration. It also involves the development and implementation of an autoencoder-based opponent modeling framework, which is evaluated across various MARL environments. Additionally, the dissertation presents an extensive experimental evaluation that offers insights into the effectiveness of different modeling strategies and the conditions under which they provide the most benefit. Finally, it includes a detailed analysis of the trade-offs between model complexity and performance, offering guidance on the practical application of agent modeling techniques in MARL.

In conclusion, this chapter has laid the groundwork for exploring the integration of agent modelling into MARL algorithms. By identifying key challenges such as partial observability, non-stationarity, and the need for computational efficiency, this study aims to bridge the existing gaps in the literature. The research focuses on the use of autoencoder-based techniques to create flexible and modular opponent models, which are tested across various environments to assess their impact on MARL performance. Through systematic experimentation and analysis, this dissertation seeks to advance the understanding of agent modelling's role in enhancing the effectiveness of multi-agent systems.

Chapter 2

Background & Literature Review

This chapter introduces the concepts essential to this study, starting with reinforcement learning and progressing to multi-agent reinforcement learning. Additionally, agent modelling is explored in the context of reinforcement learning, followed by a comprehensive literature review outlining the work conducted in this field and identifying gaps in the existing research. This discussion culminates in a discussion of the reviewed literature, highlighting key insights and establishing the foundation for subsequent chapters.

2.1 Reinforcement Learning

Reinforcement learning (RL) is a type of machine learning where an agent learns to make decisions by performing actions and receiving feedback in the form of rewards. Over time, through trial and error, the agent optimises its action selection to maximize returns. More formally, RL can be defined as the optimal control of a Markov Decision Process (MDP) [5], which is commonly used as a mathematical framework to model decision-making processes. An MDP comprises the following components:

- **State space** (S): The set of states of the environment in which the agent can exist.
- **Action space** (A): The set of actions that the agent can take.
- **Transition function** ($P(s_{t+1}|s_t, a_t)$): The transition probability, or the likelihood of the agent transitioning to state s_{t+1} from state s_t upon taking action a_t.
- **Reward function** ($R(s_t, a_t, s_{t+1})$): The reward or penalty (negative reward) received by the agent when transitioning from state s_t to state s_{t+1} via action a_t.

The goal of reinforcement learning is to find an optimal policy $\pi^*(a_t|s_t)$, which is a function that defines the probability of taking action a_t in state s_t. The policy represents the solution to the MDP, guiding the agent's actions to maximize the expected cumulative rewards over time.

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi\right]$$

where:

- t is the time step,
- γ is the discount factor, $0 \leq \gamma \leq 1$, which determines the importance of future rewards.

In conclusion, RL aims to find an optimal policy that maximises the expected cumulative reward over time by interacting with the environment and learning from the observed rewards.

2.2 Multi-Agent Reinforcement Learning

RL can be extended to Multi-Agent Reinforcement Learning (MARL), where multiple agents interact within an environment. Each agent must now make decisions that consider not only the environment but also the interactions with other agents. In the context of MARL, the problem is modelled using Markov Games or Stochastic Games [6], [7], which are extensions of MDPs.

A Markov Game for N agents can be defined as:

- **State space** (S): The set of all possible states of the environment.
- **Action space** (A_i): The set of all possible actions for each agent $i \in \{1, 2, \ldots, N\}$.
- **Transition function (P)**: A probability function $P(s'|s, a_1, a_2, \ldots, a_N)$ that defines the probability of transitioning from state s to state s' given the joint actions $(a_1, a_2, \ldots, a_N)$ of all agents.
- **Reward function** (R_i): A function $R_i(s, a_1, a_2, \ldots, a_N, s')$ that gives the immediate reward received by agent i after transitioning from state s to state s' due to the joint actions $(a_1, a_2, \ldots, a_N)$.

In this multi-agent setting, the goal for each agent is to find an optimal policy $\pi_i^*(a_i|s)$, which is a function that defines the probability of agent i taking action a_i in state s. This joint policy represents the solution to the Markov Game, guiding each agent's actions to maximise their expected cumulative rewards, considering the strategies of all other agents. Formally the game would reach a Nash Equilibrium [8] with an optimal joint policy when,

$$\pi_i^* = \arg\max_{\pi_i} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_{1,t}, a_{2,t}, \ldots, a_{N,t}, s_{t+1}) \mid \pi_1, \pi_2, \ldots, \pi_N\right]$$

where, t is the time step, γ is the discount factor, $0 \leq \gamma \leq 1$, which determines the importance of future rewards.

In multi-agent environments, each agent perceives a constantly changing environment because other agents are also learning and updating their policies, leading to the issue of non-stationary environments [9]. This dynamic makes it challenging for any single agent to converge on an optimal policy. Explicit agent modelling can help manage this complexity by enabling agents to more effectively predict and adapt to the actions of others.

Partial observability further complicates multi-agent reinforcement learning (MARL). Agents often have limited views of the environment and the states and actions of other agents, which can lead to suboptimal decision-making. Explicit agent modelling can mitigate this by allowing agents to infer hidden information and better predict the actions of others. Coordination and cooperation are also critical in many MARL scenarios where agents must align their strategies to achieve common goals. Without explicit modelling, agents may struggle to coordinate effectively, resulting in conflicts and inefficiencies. By modelling other agents, individuals can better anticipate the actions and intentions of their peers, enhancing overall coordination.

Adaptation to diverse strategies is essential in environments with heterogeneous agents that possess different capabilities. Explicit agent modelling can help agents develop flexible and robust policies that can handle a variety of behaviours exhibited by other agents by explicitly guiding them to model and predict such behaviours. Additionally, the credit assignment problem, which involves determining which agent's actions contributed to a particular outcome, adds another layer of complexity when learning in a multi-agent setting. Explicit agent modelling can address this issue by providing a clearer understanding of the interaction dynamics and individual

contributions to collective rewards.

Incorporating explicit agent modelling into MARL algorithms can help overcome these challenges in multi-agent settings. By predicting and accounting for the behaviours and strategies of other agents, MARL algorithms can achieve better performance, stability, and coordination in complex multi-agent environments.

2.3 Agent Modeling

Agent modelling, also known as opponent modelling, refers to the explicit modelling of the behaviours of other agents within a multi-agent environment. This approach contrasts with most existing Multi-Agent Reinforcement Learning (MARL) algorithms, which typically rely on the implicit modelling of opponents. In implicit modelling, agents learn about the behaviours and strategies of other agents indirectly by analyzing patterns in the environmental observation data they receive during interactions. For example, an agent may observe changes in the environment that result from the actions of others and adjust its strategy accordingly without explicitly modelling the intentions or policies of those agents.

However, this implicit approach has limitations. It generally requires more data and interactions to effectively learn about the behaviours of other agents, as it lacks direct mechanisms for understanding or predicting opponents' strategies. This inefficiency in sample usage can slow down the learning process and make it more challenging to achieve optimal performance, particularly in complex environments where the dynamics between agents are intricate and not easily inferred from observations alone.

Recognizing these limitations, researchers have hypothesized that implicit modelling may be less efficient in learning accurate representations of opponent behaviours due to its reliance on indirect inference. As a result, there has been a growing focus on introducing explicit methods of agent modelling into MARL algorithms. Explicit modelling involves directly incorporating strategies that predict or simulate the actions and policies of other agents. This approach can provide clearer insights into the behaviours of opponents and allow for more precise and proactive adjustments in strategy, potentially leading to improved performance in complex, multi-agent environments.

This section provides a comprehensive review of the existing literature on both implicit and explicit agent modelling approaches. It highlights the differences between these methods, discusses their respective advantages and challenges, and identifies gaps that present opportunities for further research. This analysis lays the groundwork

for potential areas of focus in this dissertation, particularly in the development and evaluation of more effective agent modelling strategies within MARL.

2.3.1 Modelling Agents with Deterministic Finite Automata

Model-based approaches to opponent modelling have been explored as methods for developing effective interactive strategies in classical multi-agent systems. One such approach involves representing opponent strategies as Deterministic Finite Automata (DFA)—a mathematical model characterised by a finite set of states and transitions that determine an agent's responses to different inputs [10]. These models are particularly suited for environments where agents are assumed to be selfish and rational, seeking to maximize their cumulative rewards.

Interactions in such environments can be framed as repeated two-player games, where each agent's goal is to maximise its expected cumulative rewards. To achieve this, agents construct models of their opponents' strategies by observing their behaviours and representing them as finite automata. This approach allows agents to use the observed strategies to inform and refine their own policies.

The optimal policy against a given model of opponent strategies can be determined through dynamic programming [11]. In this context, agents iteratively update their models based on new observations, gradually improving their predictions of opponents' future actions. This dynamic updating process enables agents to continuously refine their policies as they gather more data about their opponents.

The underlying mathematical framework includes a formal representation of the game between two agents, $G = \langle A_1, A_2, R_1, R_2 \rangle$, where A_i is the finite set of actions available to agent i, and R_i represents the reward function of agent i. The expected cumulative reward for agent i can be expressed as:

$$V_i(\pi_i, \pi_{-i}) = \sum_{k=t}^{\infty} \gamma^{k-t} R_i(s_k, a_i^k, a_{-i}^k) \tag{2.1}$$

where π_i is the strategy (policy) of agent i, π_{-i} represents the strategies (policies) of the opponents (all agents except i), γ is the discount factor, and $R_i(s_k, a_i^k, a_{-i}^k)$ is the reward function that gives the immediate reward received by agent i at time k for being in state s_k and taking action a_i^k while the opponents take actions a_{-i}^k.

To determine the optimal strategy M_{opt} against a given opponent model M, the problem is transformed into a Markov decision process (MDP). This transformation

involves incorporating the opponent's predicted action selections into the environment model, effectively reducing the two-player game to a single-agent MDP. The agent can then solve this MDP using dynamic programming or reinforcement learning techniques to identify the best sequence of actions that maximizes its expected rewards over time.

In the context of learning DFAs without supervised learning, the $US - L^*$ (Unsupervised Learning) algorithm, based on $Angluin's\ L^*$ [12], provides a mechanism for agents to learn a DFA incrementally, refining the automaton whenever a counterexample is encountered. This allows the model to improve its accuracy over time, particularly in environments with limited complexity, where such a heuristic approach can effectively match the opponent's behavioural patterns.

Model-based approaches utilizing DFAs and dynamic programming have shown promise in enabling agents to learn and adapt their strategies effectively in multi-agent environments with complex and evolving opponents. Although these initial methods were limited to specific frameworks, they laid the groundwork for enhancing agent interactions in environments characterized by uncertainty and dynamic conditions.

2.3.2 Opponent Modelling in Negotiation and Strategic Interactions

Opponent modelling has been explored in various competitive settings, such as automated bilateral negotiations, where agents engage in strategic interactions to resolve mutual concerns [3]. In these scenarios, agents typically have opposing objectives, and the outcome is shaped by their strategic choices. The primary goal for each agent is to improve their outcome relative to the status quo or Nash equilibrium [8]. A significant challenge in these environments is the lack of complete information about the opponent's strategies and intentions, which is particularly critical in competitive contexts. Accurate predictions of the opponent's strategy can lead to better individual outcomes and prevent exploitation by the opponent.

Negotiation scenarios are often partially observable, with limited visibility into the preferences, strategies, or actions of the counterpart. Here, opponent modelling becomes essential, as it allows agents to infer missing information and anticipate the opponent's moves, thereby enabling more informed and strategic decisions. By constructing a model of the opponent, agents can overcome the limitations imposed by partial observability, leading to more effective negotiation strategies.

Unlike heuristic-based approaches (as in Carmel and Markovitch [10]), more recent work has focused on learning methods for opponent modelling, including Bayesian

learning, non-linear regression, kernel density estimation, and neural networks. These methods provide a more structured and data-driven approach to capturing opponent behaviour. In the context of automated negotiations, core motivations for opponent modelling can be identified, such as preference estimation, strategy prediction, and opponent classification [3].

Preference estimation involves inferring the opponent's goals, rewards, or utility functions, allowing an agent to anticipate actions that the opponent is likely to take to maximize their outcomes. This approach provides long-term insights into opponent strategies, making it particularly effective in dynamic environments where core objectives remain consistent despite contextual changes.

Strategy prediction focuses on forecasting the sequence of future actions the opponent might follow, enabling the agent to prepare for and counter specific moves. This method offers tactical precision and adaptability to immediate changes in the environment, but it may introduce challenges in complex settings where short-term action distributions appear non-stationary.

Opponent classification categorizes different types of opponents, such as aggressive, defensive, or risk-averse, allowing agents to tailor their responses based on these classifications. This approach simplifies the modelling process, making it scalable and adaptable to new opponent types, though it may risk generalizing opponents and misclassification.

In automated negotiations, these modelling approaches can be applied to learn key attributes such as the acceptance strategy, deadline, preference profile, and bidding strategy of the opponent. For example, learning the acceptance strategy helps optimize proposal timing, while modelling the opponent's deadline can be leveraged to apply strategic pressure as negotiations progress. Understanding the opponent's preference profile allows for tailored offers that are more likely to be accepted, and anticipating the opponent's bidding strategy enables the agent to optimize its own offers.

Ultimately, the key takeaway from this study is the importance of preference estimation and reward modelling, which are central to the opponent modelling methods utilized later in this research (Chapter 3).

2.3.3 Opponent Modeling in MARL Setting

Albrecht and Stone [1] provide a more comprehensive survey of opponent modelling, specifically within the context of learning algorithms in multi-agent systems. The

authors establish potential underlying assumptions in modelling methods and classify various opponent modelling approaches based on the different models surveyed in the (relatively modern, compared to surveys of heuristic methods in Carmel and Markovitch [10]) literature.

The discussion here is on the underlying assumptions that help understand the applicability, limitations, and open challenges of these modelling methods, which further assist in building up to the classification of opponent modelling (classification taken from Albrecht and Stone [1]):

- **Deterministic vs Stochastic Action Choices**: Deterministic actions have a probability of 1 for each trajectory, simplifying modelling but missing complexities like randomization and action selection errors, which stochastic models capture better.

- **Fixed vs. Changing Policy**: Fixed strategies are easier to model but miss the complexity of adaptive, dynamic learning models.

- **Decision Factors Known vs. Unknown**: Unknown decision factors make modelling more challenging and less reliable, as seen in complex environments like multi-agent stock trading.

- **Independent vs. Correlated Action Choices**: Independent actions are simpler to model, but correlated actions, common in cooperative settings, require more complex models.

- **Common vs. Conflicting Goals**: Shared goals simplify modelling, while conflicting goals add complexity, influencing how actions are interpreted.

Furthermore, underlying assumptions about the environment also impact the design of opponent modelling, ranging from the order of action selection and observability to the representation of states and actions. These assumptions are further explored in Chapter 3 of the dissertation. Based on these assumptions, both for the agents and, to a lesser extent, the environment, the authors propose a comprehensive system of classification (Table A.1) of opponent modelling methods using existing literature. This system, along with Section 3.1, has been used as a guiding classification for the work proposed and undertaken in this dissertation. The authors highlight several open problems in the field, some of which are particularly relevant to this dissertation:

- **Synergistic Combination of Modeling Methods:** This involves exploring how different modelling approaches can be integrated to leverage their respective strengths, an area this study addresses by combining multiple methods to enhance overall performance.

- **Policy Reconstruction under Partial Observability:** Developing techniques for effective policy reconstruction in scenarios with partial observability, which is a significant focus of this work. This dissertation explores methods to reconstruct policies where agents lack complete information about the environment or other agents' actions.

- **Modelling Changing Behaviors:** Tracking and predicting changing behaviours in adaptive agents is critical in dynamic environments. This dissertation delves into modelling these evolving strategies to improve agent interaction and adaptation.

This dissertation specifically addresses the challenges mentioned above. These aspects are central to the research and will be discussed in greater detail in Chapter 3.

2.3.4 Autoencoder-Based Opponent Modeling

Building on the concept of policy reconstruction, a latent representation generated by an autoencoder network—trained to predict the opponent's actions and observations based on the local observations and actions of the ego agent—can be used to condition the agent's policy to infer opponent policies. The term "ego agent" refers to the agent that is being actively controlled and trained in the environment. The core idea is that the autoencoder's latent space captures the underlying structure and patterns of interaction dynamics. This enables the ego agent to predict and respond more effectively to opponent behaviour by leveraging a compact and informative representation of the observed data.

To understand the methodology discussed, it is essential to first grasp the concept of an autoencoder and its role in this context. An autoencoder is a type of neural network used to learn efficient encodings of input data. It consists of two main components: an encoder and a decoder. The encoder compresses the input data into a latent representation—a compact, encoded version of the input that captures its most salient features. This latent representation, denoted as z_t in our context, is crucial because it encapsulates the interaction dynamics between the ego agent and its opponents in a compressed form. The decoder then reconstructs the original data from this latent representation.

In the framework depicted in Figure 2.1, the encoder network processes the ego agent's observations and actions to generate this latent variable z_t, which is then used to predict or reconstruct the opponents' actions and observations. The decoder in this architecture has two reconstruction heads: one for reconstructing the opponent's observations (f_u^o) and another for reconstructing the opponent's action distributions (f_u^π).

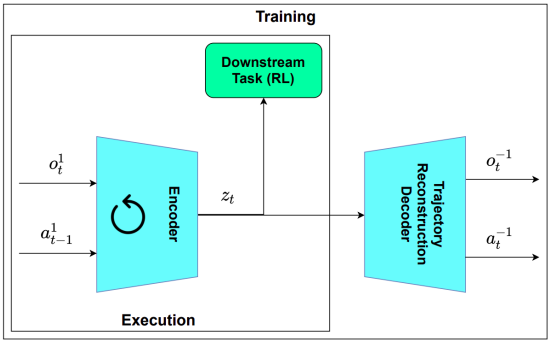

Figure 2.1: "Diagram of LIAM architecture. The two solid-line rectangles show the components of LIAM that are used during training and during execution, respectively." (Adapted from Papoudakis, Christianos, and Albrecht [2])

The reconstruction loss for these tasks differs based on the nature of the data being reconstructed. For continuous data, such as observations, the Mean Squared Error (MSE) is typically used, as it is suitable for regression tasks where the goal is to minimize the squared differences between predicted and actual values. On the other hand, for discrete data, like action distributions, the Cross-Entropy loss is more appropriate, as it is designed for classification tasks where the objective is to maximize the probability of the correct class (in this case, the correct action). These loss functions are combined to form the total loss $\mathcal{L}_{ED}$, which is minimized during training.

The advantage of using an autoencoder in this manner lies in its ability to capture the underlying structure of the data, providing a powerful tool for the ego agent to predict and adapt to the opponents' strategies. This is particularly important in complex multi-agent environments, where understanding and anticipating opponents' actions can significantly enhance the agent's performance.

This approach is explored in the work of Papoudakis, Christianos, and Albrecht [2], where the authors propose *Local Information Agent Modeling (LIAM)*. In their setup, the environment is divided into controlled agents—referred to as ego agents—who learn both policies and opponent models, and modelled agents, which have fixed heuristic policies. The controlled agent (or ego agent) is the focus of the learning process, while the other agents' strategies are static and serve as the environment's conditions. These modelled agents are treated as a single combined entity for modelling purposes.

The policy for the controlled agent, π_θ, is parameterized by θ and is optimized to

maximize the average return against the fixed policies of the modelled agent, under the assumption that these policies remain constant:

$$\arg\max_{\theta} \mathbb{E}_{\pi_\theta, \pi^{-1,k} \sim u(\Pi)} \left[\sum_{t=0}^{H-1} \gamma^t r_{t+1}^1 \right] \tag{2.2}$$

where r_{t+1}^1 is the reward received by the controlled agent (ego agent), H is the episode length, $\gamma \in (0, 1)$ is the discount factor, Π represents the set of fixed policies, π is the policy for the controlled agent, and π^{-1} denotes the fixed policy for the modelled agents. The primary goal is to learn the relationship between the trajectory of the controlled agent and that of the modelled agents.

The autoencoder loss is defined as:

$$\mathcal{L}_{ED} = \frac{1}{H} \sum_{t=1}^{H} \left[\left(f_u^o(z_t) - o_t^{-1} \right)^2 - \log f_u^\pi(a_t^{-1} \mid z_t) \right] \quad \text{where} \quad z_t = f_w(o_{1:t}, a_{1:t-1}) \tag{2.3}$$

where f_u^o is the observation reconstruction head of the decoder, f_u^π is the policy reconstruction head of the decoder, and f_w represents the encoder. One important point in such a system is that the weights and output dimensions of the decoder heads change linearly with the number of opponent agents. The latent variable z (from equation 2.3) from the encoder is then fed into the RL algorithm for the controlled agent, which in Papoudakis, Christianos, and Albrecht [2] was the Advantage Actor-Critic algorithm (A2C) [13]. This approach can be similarly extended to other RL algorithms by conditioning the policies on the latent variable input along with the controlled agent trajectories.

The authors demonstrated the performance improvement in LIAM compared to non-agent modelling baselines, as well as compared to other agent modelling baselines such as Full Information Agent Modeling (FIAM), an ablation of LIAM with full information, VariBad [14], Classification-Based Agent Modeling [1], and Contrastive Agent Representation Learning (non-reconstruction baseline based on Oord, Li, and Vinyals [15]). Among these, only FIAM, an ablation of LIAM with access to full information instead of limited, and VariBad are autoencoder reconstruction-based agent modelling methods. One point of interest is that the VariBad algorithm uses the observation, action, and reward triplet and a variational autoencoder along with a controlled agent, as in LIAM. This approach is one of the variations studied in this dissertation (as seen later in Chapter 3), especially considering the use of rewards as a

reconstruction target, similar to the preference estimation target in Baarslag, Hendrikx, Hindriks, *et al.* [3]. Furthermore, LIAM also employs a recurrent autoencoder, considering the relative complexity of their target environments.

This idea has been extended to a multi-agent reinforcement learning (MARL) setting as discussed in Albrecht, Christianos, and Schäfer [9] (Multi-Agent Reinforcement Learning Book, Chapter 9.6). In this context, a simple autoencoder architecture is employed to reconstruct opponent actions using the ego agent's observations. Unlike in LIAM, where opponent policies are fixed, this approach deals with dynamic opponent policies, leading to a more complex and non-stationary target distribution. The policies are conditioned on the latent variable from the encoder, allowing for an algorithm-agnostic implementation that can potentially enhance robustness and learning efficiency. The authors demonstrate this through preliminary experiments using a centralised A2C algorithm in a relatively simple environment.

However, the experiments presented in the book are limited to action reconstruction in basic settings, which leaves open questions about the impact of reconstructing other targets, such as observations (as in LIAM) and rewards (as inspired by Baarslag, Hendrikx, Hindriks, *et al.* [3]), in a MARL context. Additionally, it is unclear how these methods will perform in more challenging tasks. These open questions form the basis of this thesis, which aims to explore the use of autoencoders to learn latent representations of opponent behaviours and condition policies on these representations, extending the approach to more complex and varied scenarios.

2.4 Literature Review Insights

This chapter established a foundation for integrating agent modelling in MARL algorithms, beginning with an introduction to RL and building up to the complexities of RL in multi-agent systems. The importance of explicit agent modelling is emphasised, where agents must consider the behaviours of others to develop effective strategies. As discussed, multi-agent systems give rise to additional challenges, including non-stationarity, scalability, partial observability, and the necessity for coordination. These challenges make it difficult for traditional RL methods to achieve optimal policies in multi-agent environments, underscoring the need for more sophisticated approaches where explicit agent modelling has the potential to mitigate the mentioned issues.

This chapter has explored the state of research in this area, identifying both the progress made and the existing gaps, which in turn shape the direction of this dissertation.

The key takeaways from the current literature are:

- **Importance of Explicit Agent Modeling**: The review highlights the limitations of implicit modelling approaches, which rely on analysing patterns in observational data. These methods are often less sample-efficient and may struggle with complex, dynamic environments. Explicit agent modelling, on the other hand, provides a more structured and efficient way to predict and adapt to other agents' actions.

- **Diverse Methodological Approaches**: Various methods for opponent modelling have been explored, including classical approaches such as deterministic finite automata and dynamic programming, as well as more modern techniques like Bayesian learning, neural networks, and autoencoder-based methods. Each approach offers distinct advantages but also comes with specific limitations, such as scalability issues or the need for extensive prior knowledge.

- **Gaps in Current Research**: The review identifies several gaps in the existing literature, particularly the need for more robust and scalable opponent modelling techniques to handle dynamic multi-agent environments' complexities. There is also a noted need for methods that can efficiently address partial observability and non-stationarity, which are common challenges in MARL settings.

- **Potential of Autoencoder-Based Methods**: Autoencoder-based methods, which involve reconstructing opponent actions and strategies from compact latent representations, are identified as a promising approach. These methods offer a flexible and algorithm-agnostic solution that can be adapted to various MARL scenarios, potentially improving both the efficiency and effectiveness of learning in complex environments.

These insights provide a solid conceptual and practical foundation for exploring the integration of opponent modelling into MARL. This chapter lays the groundwork for further development and experimentation with advanced modelling techniques in multi-agent systems by identifying the strengths and weaknesses of existing methods and recognising the potential of autoencoder-based approaches. This exploration aims to contribute new understanding and solutions to the challenges inherent in multi-agent reinforcement learning.

Chapter 3

Methodology & Experiments

This chapter delves into the integration of opponent modelling techniques into MARL algorithms, beginning with an explanation of the model architecture and the integration methodology. It also covers the training procedures within an algorithm-agnostic framework, the design of the selected environments, and the MARL algorithms considered in this dissertation. The evaluation design is outlined, emphasising key metrics derived from both reinforcement learning and unsupervised learning perspectives, specifically in the context of using autoencoders to reconstruct features of the opponents, such as actions, observations or rewards. The primary focus is on implementing an autoencoder-based reconstruction method (inspired by Section 9.6 in Albrecht, Christianos, and Schäfer [9]), utilising neural networks to model opponents effectively. This approach will be tested across various environments and benchmark MARL algorithms to evaluate its impact on a variety of metrics.

3.1 Autoencoder Reconstruction Based Modelling

The LIAM Papoudakis, Christianos, and Albrecht [2] and Centralised A2C + AM Albrecht, Christianos, and Schäfer [9] agent modelling algorithms (Section 2.3.4) have demonstrated the potential of autoencoders for use in agent modelling. Building on the foundational work in this area, this study presents an advanced method termed Autoencoder Reconstruction-Based Modelling. This approach extends previous research by incorporating a wider array of reconstruction targets, including actions, rewards, and observations, both individually and in combination. This method aims to capture more nuanced aspects of opponent strategies and systematically evaluates the impact of varying latent space dimensions. By addressing the limitations of earlier studies, this

research provides a more comprehensive analysis of how different reconstruction targets and latent dimensions influence the effectiveness of opponent modelling in multi-agent reinforcement learning environments. It is crucial to clarify that the term "opponent" refers to any agent other than the primary agent; thus, for each agent, all other agents are considered "opponent" agents.

Using different reconstruction targets leads to learning varied representations that encode different aspects of opponents' behaviours. This dissertation focuses on several specific reconstruction targets. First, conditional action frequencies involve directly reconstructing opponents' actions or action frequencies, which can effectively capture their current policies. However, this approach tends to focus on the short-term trajectories of the opponents and may not fully capture their long-term planning. Additionally, it may be susceptible to minor variations in policy. This issue is further exacerbated by the fact that all agents are continuously learning, meaning the opponents' policies are drawn from a non-stationary distribution. A key consideration is the choice between modelling actions, which capture discrete decisions at specific moments, and modelling action frequencies, which provide a broader view of an agent's behaviour over time by reflecting the likelihood of different actions. Using action frequencies can introduce a degree of stochasticity, which, while more complex, may allow for the reconstruction of both the confidence in different actions and the entropy within the opponent's policies rather than just capturing the final selected actions.

Second, reward reconstruction focuses on the rewards obtained by opponent agents, which helps in inferring strategies from the perspective of goal-directed behaviour. This approach is similar to preference estimation as discussed in Baarslag, Hendrikx, Hindriks, *et al.* [3]. However, a potential drawback is the complexity involved in tracking and effectively interpreting the policy from the rewards, which may result in a less straightforward and slower learning curve. This approach can be considered as capturing more long-term strategic elements, given that it leverages reward signals, which inherently influence the trajectory of policy updates. However, it is crucial to recognize that this method primarily reflects rewards at individual timesteps rather than providing direct insight into the overarching objectives that the agents are optimizing for.

Finally, Observation reconstruction involves capturing the observations of the opponents. This method is useful for encapsulating spatial navigation patterns and strategic movements of opponent agents, which is particularly beneficial in scenarios where spatial reasoning is crucial, such as with autonomous vehicles. However, there

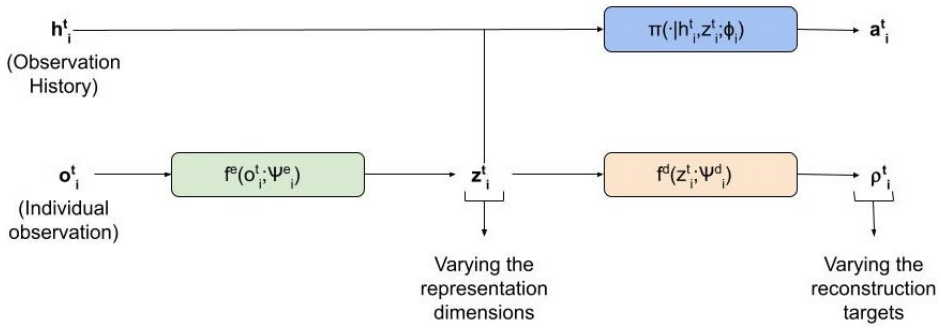

Figure 3.1: The encoder network f_e generates a representation m_i^t depicting the policies of other agents. Throughout the training process, a decoder network f_d is trained to reconstruct various targets, denoted as ρ, pertaining to the actions, rewards, or state trajectories. During execution, the representation is used to condition the agent's policy in addition to the history, h_i^t. Additionally, varying the dimensions of the representation serves as a focal point for evaluation studies.

is an added layer of complexity in inferring the agent's policies based solely on the observed states of the opponents. Unlike direct action reconstruction, where the actions themselves are the primary focus, trajectory-based methods require the model to deduce how these actions translate into movement patterns within the environment.

In terms of classification, autoencoder-based reconstruction can be seen as a spectrum from classification-based modelling to policy-based reconstruction. In simpler terms, as the latent space dimensions increase, the modelling shifts from classification-based (with a single latent variable) to policy-based reconstruction, where the dimension equals the number of agents multiplied by the action dimensions, i.e., $n\_agents * action\_dim$.

As in Figure 3.1, the autoencoder reconstructs the chosen reconstruction target ρ_i^t depicting characteristics of opponent agents, using the agent's individual observation as inputs, and passing it through a bottleneck, i.e. the latent variable controlled by the representation dimensions. The policy and/or critic of the MARL algorithm is then conditioned on the encoded representation, thereby passing in a latent representation of the reconstructed opponents' reconstruction targets, i.e.

$$a_i^t \sim \pi_i(\cdot|o_i^t, z_i^t, \theta_i),$$

where a_i^t is the action selected by agent i at time t, o_i^t is the observation of agent i at time t, z_i^t is the latent representation produced by the autoencoder that encodes information about the opponent's high-level strategy (inferred based on the reconstruction target), and θ_i represents the policy parameters of agent i. The policy π_i defines a probability distribution over possible actions, conditioned on the current observation o_i^t, the latent

representation z_i^t, and the policy parameters θ_i.

Since the encoded representation can be used to condition both the policy or the critic, this type of agent modelling can be easily adopted by most MARL algorithms without much friction with the preexisting algorithms. This can be evidenced by the adaptation of the existing EPyMARL library [16] to extend Autoencoder Reconstruction Based Modelling to the algorithms of Multi-Agent Proximal Policy Optimization, Independent Proximal Policy Optimization, Independent Q Learning (See Algorithm 2).

Extending this, the reconstruction targets are not limited to the individual ones mentioned above; multiple targets can be combined to capture a broader range of unique aspects from each target. For example, combining observation and action trajectories could encode both the opponent agents' spatial reasoning and action selection strategies. Combining the advantages of both approaches could provide a more holistic understanding of the opponent's behaviour. However, it is important to note that a bottleneck in the autoencoder, which is smaller than the input, leads to lossy compression. Consequently, the encoder's capacity to capture detailed information is constrained, depending on the dimensionality of the encoding, which may result in diminished performance if all reconstruction targets are used simultaneously.

Further, the autoencoder is trained in conjunction with the agent's learning process; in other words, the autoencoder model is updated with the agent's observations and opponent targets each time the MARL algorithm undergoes an update. This is done to ensure up-to-date modelling since, in this proposed method, the opponents' policies are also updated with each agent policy update. Hence, from the point of the autoencoder, the dataset is non-stationary, requiring continuous training. Also, due to this reason, combining multiple experiences across multiple episodes to form larger batches to train the autoencoder on might lead to lower performance since the dataset would contain reconstruction targets and ego observations from different distributions due to changes in the policies of all agents.

The losses for the autoencoder have been calculated based on the reconstruction target. For observation and reward as reconstruction targets, being continuous variables, mean square error is used to calculate the loss:

$$Loss_{obs} = \frac{1}{N}\sum_{i=1}^{N}\|\hat{o}_i - o_i\|^2,$$

where o_i is the true observation vector for the ith data point, $\hat{o}_i$ is the reconstructed observation vector, obtained from the autoencoder network, for the ith data point, and N

is the total number of observations.

$$Loss_{rew} = \frac{1}{N}\sum_{i=1}^{N}\|\hat{r}_i - r_i\|^2,$$

where r_i is the true reward value for the ith data point, $\hat{r}_i$ is the reconstructed reward value for the ith data point, and N is the total number of data points.

For action frequencies, which are continuous variables representing probabilities between 0 and 1, there are two primary approaches when dealing with discrete actions. The first approach involves recording the opponent's actions and applying cross-entropy loss, which provides a straightforward method of minimizing the difference between predicted and observed actions. The second approach involves recording the opponent's action frequencies and applying Kullback-Leibler (KL) divergence loss to minimize the divergence between the predicted and actual action distributions. KL divergence is particularly beneficial as it captures the opponent's action preferences and the entropy of their action frequencies, offering a more nuanced understanding of the opponent's policy. In contrast, cross-entropy loss does not carry an implicit assumption of access to the complete distribution of the opponent agents' policies, which is arguably more difficult to obtain. For cross-entropy loss:

$$Loss_{ce} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(\hat{y}_{i,j}),$$

where $y_{i,j}$ is the true label for action j taken by opponent i, $\hat{y}_{i,j}$ is the predicted probability of action, obtained from the Autoencoder network, j for opponent i, M is the number of possible actions, and N is the total number of observations.

For Kullback-Leibler (KL) divergence loss:

$$Loss_{kl} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log\left(\frac{y_{i,j}}{\hat{y}_{i,j}}\right),$$

where $y_{i,j}$ represents the true action frequency distribution for opponent i, and $\hat{y}_{i,j}$ represents the predicted action frequency distribution.

These losses, when combined with the observation and reward reconstruction losses, can effectively guide the model to learn comprehensive representations of the opponent's behaviour:

$$Loss_{total} = \alpha \cdot Loss_{obs} + \beta \cdot Loss_{rew} + \lambda \cdot Loss_{ce/kl},$$

where α, β, and λ are hyperparameters that control the contribution of each loss component to the total loss. Depending on the specific application and the nature of the opponent's strategy, one may choose between cross-entropy loss or KL divergence to fine-tune the model's learning process. However, for the purpose of this study, the scope has been limited to $\alpha = 1$, $\beta = 1$ and $\lambda = 1$. This algorithm is detailed further in Algorithm 1.

The key idea is to adjust the model's complexity by varying the dimensionality of the latent space, thereby enabling a gradient of abstraction levels in opponent modelling and exploring various permutations of reconstruction targets to extract different aspects of the opponents' strategies.

3.1.1 Autoencoder Model Training Procedure

The autoencoder will be trained as part of the integrated MARL framework (Algorithm 2), with reconstruction targets derived from the opponent agents' observed actions, action frequencies, and rewards. The training process involves several key steps. First, dataset generation is conducted by running MARL algorithms in simulated environments, where data comprising observations, actions, action frequencies, and rewards of agents are collected. This data is structured and managed using the replay buffer, which is reset per episode to retain opponent samples only from the most recent policy updates.

Next, the loss functions play a critical role in the training process. The primary objective function incorporates mean squared error (MSE) for continuous variables, such as observations and rewards, and Kullback-Leibler (KL) divergence loss for action frequencies, which are treated as probabilistic outputs. Alternatively, cross-entropy loss may also be employed to calculate probabilistic action frequencies. In addition to the reconstruction loss, KL divergence may serve as a regularisation term to maintain a well-distributed latent space, ensuring that the encoded latent representations effectively capture the diversity of opponent behaviours.

Optimisation is achieved using the Adam optimiser, with learning rates specifically tuned for the opponent modelling task. The autoencoder is updated in tandem with the policy updates within the MARL framework, ensuring that the opponent models adapt as training progresses. This synchronisation helps maintain a stationary distribution by using data from before the policy updates.

Algorithm 1 Autoencoder Reconstruction Based Agent Modelling Learner (Per Agent)

Input: Episode Replay Buffer (batch), epochs, N_Agents, Encoder Parameters (Ψ^e), Decoder Parameters (Ψ^d)
Output: Updated Encoder Parameters (Ψ^e), Updated Decoder Parameters (Ψ^d)
for each *epoch* in [1, epochs] **do**
    Sample mini-batch from replay buffer (batch)
    for each agent i in [1, N_Agents] **do**
        Obtain a Batch of ego-agent observations (o_i) and opponent agents' observations, actions, and rewards.
        Encode ego-agent observations (o_i) using the encoder network (f^e) to obtain latent representation:

$$\mathbf{z_i} = f^e(\mathbf{o_i}, \Psi_i^e)$$

        Decode latent representation (z_i) to reconstruct observations ($\hat{\mathbf{o}}_i$), action frequencies ($\hat{\mathbf{p}}_i$), and rewards ($\hat{\mathbf{r}}_i$), using decoder network (f^d):

$$\hat{\mathbf{o}}_i, \hat{\mathbf{p}}_i, \hat{\mathbf{r}}_i = f^d(\mathbf{z_i}, \Psi_i^d)$$

    end for
    Calculate losses:

- **Observation Reconstruction Loss**: If observation reconstruction target is used: Mean Square Error (Refer Equation 3.1)
- **Action Frequency Reconstruction Loss (KL Divergence)**: If action reconstruction target is used: KL Divergence Loss (Refer Equation 3.1)
- **Reward Reconstruction Loss**: If reward reconstruction target is used: Mean Square Error (Refer Equation 3.1)

    Compute the total loss:

$$\mathcal{L} = \alpha \cdot \text{Loss}_{\text{obs}} + \beta \cdot \text{Loss}_{\text{rew}} + \lambda \cdot \text{Loss}_{\text{act}},, \text{ for this study } \alpha = 1, \beta = 1, \lambda = 1.$$

    Update encoder parameters Ψ^e:

$$\Psi^e \leftarrow \Psi^e - \eta \cdot \nabla_{\Psi^e} \mathcal{L}$$

    Update decoder parameters Ψ^d:

$$\Psi^d \leftarrow \Psi^d - \eta \cdot \nabla_{\Psi^d} \mathcal{L}$$

end for
Return: Updated Encoder Parameters (Ψ^e), Updated Decoder Parameters (Ψ^d)

3.2 Experimental Framework

3.2.1 Selected Study Environments

The focus here lies on selecting an environment and identifying the key factors and design considerations that ensure alignment with the goals of investigating opponent

Algorithm 2 Generic MARL with Integrated Opponent Modelling

Input: Initial policy parameters for each agent (θ_i), autoencoder model parameters (ϕ_i), MARL environment, total episodes (episodes), learning rate (α), and opponent modeling frequency (τ).
Output: Trained policies for all agents.
Initialize policy parameters θ_i for each agent i.
Initialize autoencoder model parameters ϕ.
for episode = 1 to episodes **do**
    Reset environment and observe initial state s_0.
    for t = 0 to T **do**
        for each agent i in $1, \ldots, N$ **do**
            Select action $a_i^t \sim \pi_{\theta_i}(a_i^t | h^t, \mathbf{z}_i^t)$, where $\mathbf{z}_i^t$ is the encoded latent representation from the autoencoder.
            Execute action a_i^t and observe next observation o^{t+1} and reward r_i^t.
        end for
        Store transition $(o^t, \mathbf{a}^t, r^t, o^{t+1})$ in replay buffer.
        Call Autoencoder Reconstruction Based Agent Modelling Learner (Algorithm 1) **for each agent** i
        Sample mini-batch from replay buffer for a policy update.
        for each agent i in $1, \ldots, N$ **do**
            Calculate policy loss (e.g., Proximal Policy Optimization (PPO), Q-learning, etc.).
            Perform backpropagation on policy and update θ_i using optimizer.
        end for
    end for
end for
Return: Trained policies θ_i for all agents.

behaviour, learning dynamics, and evaluating the effectiveness of the proposed modelling methodologies. The key focus is to utilise an environment that offers sufficient flexibility to test various scenarios while being complex enough to discern subtle differences in algorithmic performance. Additionally, the environment should be capable of simulating real-world scenarios where agents contend with uncertain and adversarial opponents, allowing for a more thorough assessment of the effectiveness in addressing challenges such as opponent deception, collaboration, and adaptation.

One important factor is the level of observability, which pertains to whether the agent has partial or complete observability within the environment. Another key consideration is whether the environment is competitive, cooperative, or somewhere in between, meaning whether the agents have opposing goals, supportive goals, or non-opposing goals that are not entirely aligned. Additionally, agent heterogeneity is an important aspect, referring to the presence of agents with different observation spaces, action

spaces, or interaction dynamics, such as locomotion dynamics. This heterogeneity can exist even among agents working toward the same goal.

Based on these design considerations, the following environments were chosen as the focus of this study:

- Multi-Particle Environments (MPE) [17] [18]: A collection of particle-based environments featuring a variety of tasks that enable variable observability, as well as a mix of competitive and cooperative scenarios. These environments offer a wide variety of tasks, coupled with being sufficiently complex yet computationally lightweight, making them suitable for use within a reasonable computational range.

- Level Based Foraging (LBF) [19] [16]: This is a mixed cooperative-competitive environment where agents navigate a grid, each with a level, to collect food. The emphasis, however, is on cooperation, as agents must combine their levels to meet or exceed the food's level for successful collection. The environment challenges agents to balance individual and group strategies, making it complex due to sparse rewards and the need for strategic decision-making.

- StarCraft Multi-Agent Challenge lite (SMAClite) [20]: This environment is a more computationally efficient version of the original SMAC [21], providing a relatively more complex environment as compared to others on this list, with dynamic interactions among diverse agents and additional environmental landmarks.

Hence, selecting the right environment is essential for effective agent modelling, as it must capture diverse interactions and support the investigation of opponent behaviour, learning dynamics, and the evaluation of modelling methodologies.

Specifically, the environments utilized for the experiments in this study are:

- SMAClite 2s3z: A simplified version of SMAC with two Stalkers and three Zealots. It was chosen for its dynamic interactions, being a cooperative environment, and serving as a moderate complexity test for agent modelling in real-time adversarial settings.

- LBF 10x10 3p 3f (Cooperative with Common Reward): A cooperative foraging scenario where three agents must coordinate to collect food. This environment tests the effectiveness of agent modelling in enhancing collaborative efforts in a fully cooperative setting.

- LBF 10x10 3p 3f (Competitive with Individual Reward): A mixed competitive cooperative variant where agents compete to collect food individually. It assesses the impact of agent modelling on competitive strategies and decision-making.
- LBF 2s 10x10 3p 3f (Cooperative with Common Reward; Partial Observability): Introduces partial observability in a cooperative foraging scenario, increasing complexity by requiring agents to infer the actions of others. It tests the robustness of agent modelling under uncertainty.
- MPE Simple Spread (Cooperative with Individual Rewards): A simple scenario where agents must learn to cover the landmarks on a plane while avoiding collisions. It combines cooperation and spatial reasoning, making it ideal for testing decentralized coordination in agent modelling.
- SMAClite MMM2: A challenging mixed-unit scenario where different unit types (Marine, Marauder, Medivac) must coordinate attacks and defences. This environment assesses how well agent modelling enhances performance in complex, multi-faceted tactical situations.

3.2.2 MARL Algorithm of Focus

In this study, the focus is exclusively on the MAPPO (Multi-Agent Proximal Policy Optimization) [22] algorithm due to its superior performance in solving complex environments more efficiently. MAPPO is a widely-used and well-regarded algorithm that extends Proximal Policy Optimization (PPO) [23] to multi-agent settings, coordinating multiple agents through centralized training while allowing for decentralized execution. This makes it particularly effective in both cooperative and competitive environments, where it can navigate and optimize strategies faster than other algorithms.

By using MAPPO, the study ensures that environments can be solved more rapidly, allowing for a direct and meaningful comparison of converged rewards across different experimental setups. This approach not only streamlines the investigation but also leverages MAPPO's robust performance to highlight the potential improvements brought by agent modelling.

3.3 Experimental Procedure

The experiments will be designed to systematically assess the impact of opponent modelling on the performance of MARL algorithms across different environmental, algorithmic, and model characteristics. This approach will allow for a more comprehensive understanding of the effects of agent modelling:

1. **Environmental Comparative Performance**: Run the MAPPO algorithm with and without opponent modelling to establish baseline comparisons across different environments. The focus here is on fixing the latent variable dimensions and reconstruction targets while observing variations in the comparative performance of MARL algorithms with and without agent modelling across different environments. This will help identify which environmental characteristics are most impacted by agent modelling.

2. **Performance on Different Combinations of Reconstruction Targets**: Run the MAPPO algorithm with agent modelling, varying the different possible permutations of the reconstruction targets, and contrast the results with those obtained without agent modelling. This will provide a better understanding of which reconstruction targets, or combinations thereof, are more effective in extracting opponent strategies across different scenarios.

3. **Ablations on the Latent Variable Dimension**: Conduct experiments that vary the latent variable dimensions to determine the optimal bottleneck size needed to extract information about opponent strategies while minimizing excessive data loss due to the lossy compression inherent in the autoencoder network.

4. **Performance of Different MARL Algorithms with Agent Modeling**: Vary the MARL algorithms to determine which algorithms are more compatible with agent modelling and analyze the learning curves for each. While the focus of the other experiments is primarily on MAPPO, this set of experiments will explore a broader range of algorithms.

5. **Ablations on the Autoencoder Type**: Experiment with different variations of autoencoders, such as variational autoencoders, to determine which are better suited for extracting and compressing the maximum amount of detail from opponent strategies in a multi-agent setting.

6. **Computational Overhead**: Measure the computational overhead to identify the appropriate tradeoff between performance and computational cost.

Together, these experiments aim to provide an in-depth understanding of agent modelling and its varied effects in different scenarios. This will help determine in which scenarios agent modelling can be effectively utilised, considering the computational overhead, and which scenarios, along with specific hyperparameters such as latent variable dimensions or reconstruction targets, are best suited for maximising the effectiveness of agent modelling.

3.4 Evaluation Metrics

The evaluation framework focuses on both the accuracy of the reconstructions and the overall impact on multi-agent learning performance, considering a range of key metrics.

To evaluate the performance of the autoencoder-based approach, the following metrics will be tracked:

- **Accuracy**: The accuracy of reconstructed actions compared to the true actions taken by the opponent. This metric is applied only when action reconstruction is used due to the nature of action probabilities, which can be inferred as classification problems.

- **Entropy**: The entropy of the predicted action distributions is used as a measure of the uncertainty and diversity in the policy reconstructions. Comparing this with the entropy of the ground truth opponent actions provides a better understanding of the action reconstruction performance.

- **Loss**: Monitoring the reconstruction losses for observations (Mean Square Error), actions (Cross Entropy Loss or KL Divergence Loss), and rewards (Mean Square Error) separately to understand which aspects of the opponent's behaviour are best captured by the model.

The evaluation will also consider broader performance indicators to understand the impact of the autoencoder-based approach on MARL algorithms:

- **Converged Returns**: Assessing the stability and effectiveness of learning after convergence.

- **Win Rate**: Measuring the success rate in achieving goals or defeating opponents. The specific focus here is on the SMAClite environments, as in this context, win rate is a meaningful metric when comparing against a fixed opponent team.
- **Computational Efficiency**: Measuring the computational overhead introduced by the opponent model in terms of training time.

Overall, this evaluation design provides a comprehensive assessment of the autoencoder-based approach, balancing the accuracy of opponent behaviour modelling with the broader implications for multi-agent learning performance and computational efficiency. These metrics will guide the refinement of the approach and ensure its effectiveness in diverse and dynamic environments.

3.5 Conclusion

In conclusion, this chapter has established a comprehensive framework for integrating and evaluating opponent modelling within MARL algorithms. By focusing on a carefully selected set of environments and algorithms, the study aims to assess the effectiveness of autoencoder-based modelling across different scenarios. The evaluation metrics, encompassing both accuracy in reconstruction and overall performance impact, provide a robust foundation for analyzing the benefits and trade-offs of this approach. This framework not only facilitates a deeper understanding of opponent behaviour modelling but also sets the stage for future exploration of how such models can enhance multi-agent learning dynamics in complex, real-world settings.

Chapter 4

Results & Insights

This chapter presents the results of the experiments designed to assess the impact of opponent modelling on the performance of MARL algorithms. The analysis is structured according to the experimental procedure outlined in Section 3.3.

4.1 Environmental Comparative Performance

In this case, the focus of the experimentation is to find the environment characteristics where agent modelling would have the maximal impact. To compare this performance and to isolate the effect of the different target environments on the algorithm's performance, the latent variable dimensions and reconstruction targets are fixed. Here, only action frequencies have been used as the reconstruction target. Further, only the MAPPO algorithm is utilized in this scenario. Further in-depth details of the hyperparameters and training configurations per experiment have been included in Appendix D.

To thoroughly test agent modelling across the environmental characteristics mentioned in Section 3.2.1, a combination of environments with varying levels of complexity, degrees of observability, cooperation, and heterogeneity was chosen. The following environments were utilized to ensure the broadest coverage of scenarios *SMAClite 2s3z*, *LBF 10x10 3p 3f*, *LBF 10x10 3p 3f*, *LBF 2s 10x10 3p 3f*, *MPE Simple Spread*.

Figure 4.1 shows that when using cross-entropy loss, the inclusion of opponent modelling in MAPPO generally results in either equivalent or lower episodic returns across the tested environments. In the *2s3z* and the *LBF* environment, the returns when using agent modelling are up to par with the baseline episodic returns. However, in *Simple Spread* and *LBF* with partial observability, the agent modelling results in lower episodic returns. In simpler environments, agent modelling gives a similar return to the

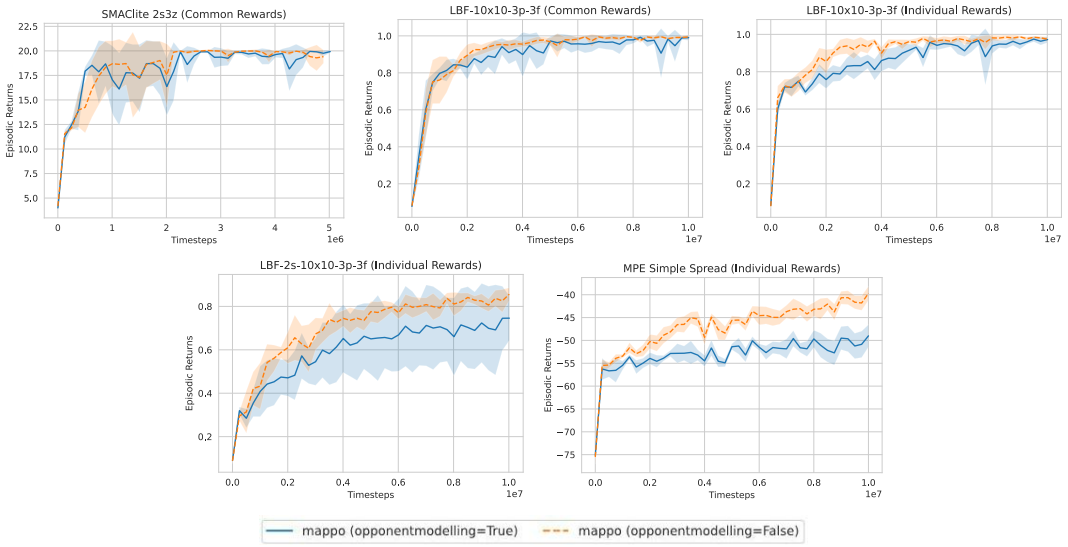

Figure 4.1: Episodic returns comparison of MAPPO with and without opponent modelling across various environments. Using action frequencies as reconstruction targets, and using cross-entropy loss for the autoencoder. The selected environments include MPE Simple Spread, LBF in cooperative and competitive settings (both fully observable and partially observable), and SMAClite 2s3z. (Note: Here 2s in the context of LBF environment represents partial observability; refer Christianos, Schäfer, and Albrecht [19])

baseline, the performance plateaus and achieves a similar episodic return; hence, for further experiments, the more complex environments are the focus. Further, it can also be hypothesised that in complex environments, more timesteps are required to close the gap and for agent modelling to improve upon the baseline performance.

Building on this, Figure 4.2, shows that on switching to KL Divergence loss for the action frequency significantly improves performance, and narrows the gap. In the case of LBF with partial observability, the gap is completely closed, and agent modelling results in the same performance as the baseline performance. KL Divergence is generally well-suited for capturing the nuances of probabilistic distributions [24], such as action frequencies. By focusing on minimizing the divergence between the predicted and actual action distributions, the model may be better able to capture the strategic behaviour of opponents, leading to more effective opponent modelling. Hence, it can be reasoned that when capturing the action frequencies is an option, the use of KL divergence would result in better performance.

Further extending the experiment to a more complex environment—the SMAClite MMM2 environment, as shown in Figure 4.3—using action frequency as the reconstruction target results in marginal improvements in episodic returns and leads to a

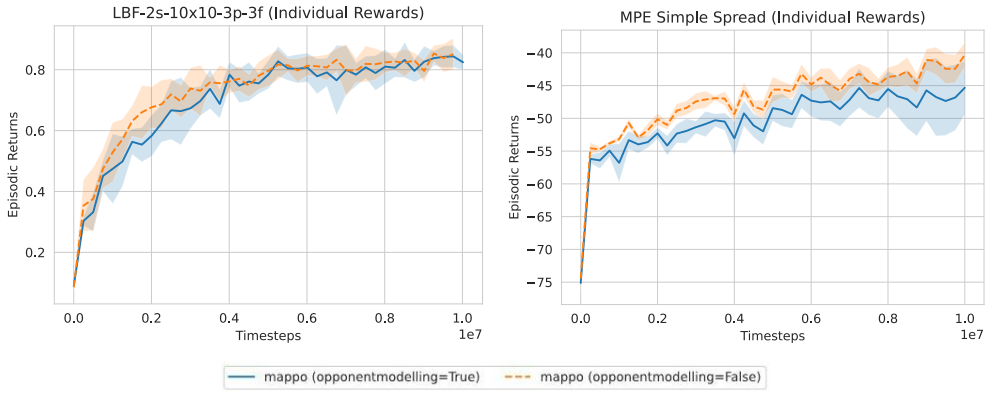

Figure 4.2: Episodic returns comparison of MAPPO with and without opponent modelling across various environments. Using action frequencies as reconstruction targets, and using KL Divergence loss for the autoencoder. (Note: Here 2s in the context of LBF environment represents partial observability; refer Christianos, Schäfer, and Albrecht [19])

more robust algorithm, as evidenced by the lower deviations compared to the baseline. In MMM2, the coordination of diverse unit types (Marine, Marauder, Medivac) with different abilities and roles introduces a higher level of tactical decision-making. Agent modelling could be more beneficial in this context because it allows the algorithm to better anticipate and adapt to the varied strategies employed by the opponent, particularly in managing the synergies between different units. Here, in the case of simpler, more predictable environments, opponent modelling does not provide a significant advantage, leading to similar performance with or without agent modelling. Therefore, the added complexity and the need for nuanced decision-making in MMM2 likely make agent modelling more impactful in this environment, resulting in the observed performance improvement.

These results demonstrate that the impact of opponent modelling varies significantly across different environments, with more complex and strategic settings like SMAClite MMM2 showing noticeable performance improvements when agent modelling is utilized. Conversely, in simpler or more predictable environments, the benefits of opponent modelling are less apparent, leading to a comparable

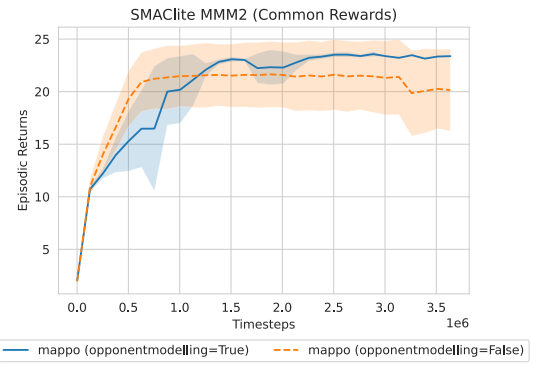

Figure 4.3: Episodic returns comparison of MAPPO with and without opponent modelling on the SMAClite MMM2 map, with KL Divergence loss and action frequency as reconstruction target.

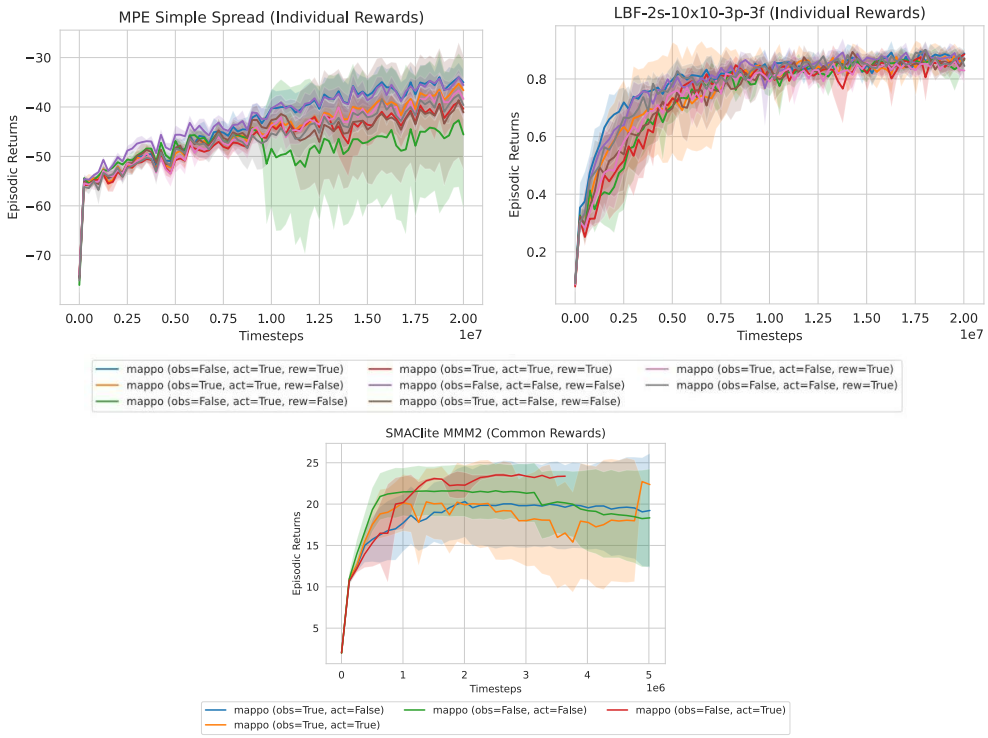

Figure 4.4: Episodic returns comparison of MAPPO with different permutations of reconstruction targets on the SMAClite MMM2 map, LBF and MPE Simple Spread environments, with KL Divergence loss for action frequency reconstruction.

performance with the baseline. These findings suggest that the complexity of the environment plays a crucial role in determining the effectiveness of opponent modelling, highlighting its potential in scenarios that demand sophisticated coordination and strategy.

4.2 Performance on Different Combinations of Reconstruction Targets

This experiment evaluates the effectiveness of different combinations of reconstruction targets on the performance of MAPPO with agent modelling. The results are also compared to the baseline performance without agent modelling (Figure 4.4).

Figure 4.4 shows that in the MPE Simple Spread environment, unlike the performances observed in the previous section using only action frequency as a reconstruction target, certain combinations here surpass the non-agent modelling baseline performance, though the differences are not as pronounced. Notably, the combination of action

and reward reconstruction performs better than the baseline without agent modelling. This improvement can be attributed to the action reconstruction capturing short-term strategies and the reward reconstruction capturing long-term strategies, as discussed in Section 3.1. Additionally, it is evident that solely using reward reconstruction results in the lowest performance, likely because the policy must infer the opponents' short-term policies from latent representations of the reward patterns. The combination of short-term action frequencies and long-term reward reconstruction yields the most significant performance improvement. Reward reconstruction appears to act as a performance enhancer when combined with both action and observation reconstruction. This pattern is consistent across different environments and environmental characteristics, although it is less pronounced in simpler environments.

A consistent pattern observed across the experiments is the lower performance of observation reconstruction compared to action frequency reconstruction. Observations often contain a significant amount of information that may not be directly relevant to predicting the opponent's strategy or future actions. This extraneous information can introduce noise, diluting the effectiveness of the model's learning process. In contrast, action reconstruction focuses directly on the opponent's decision-making process, which is more closely aligned with the strategic behaviours that the model seeks to capture. This, combined with the possibility of increased variability, as shown in Figure 4.4, can explain the lower performance of observation reconstruction targets.

However, as seen in Figure 4.5, utilizing cross-entropy loss in combination with action reconstruction and observation reconstruction appears to improve performance compared to other reconstruction target combinations, although it still falls short of baseline levels. This improvement can be attributed to the fact that incorporating observation reconstruction can lead to the development of richer and more informative latent representations. These enhanced representations may capture more subtle dependencies between the environment and the opponent's actions, which, in the case of cross-entropy loss, are not present when learning is focused solely on optimal actions rather than on opponents' preferences. Nevertheless, the high variability associated with observation reconstruction remains a challenge even in this combined approach.

Overall, combining action and reward reconstruction generally results in the most significant performance improvements, effectively capturing both short-term and long-term strategies. However, observation reconstruction typically underperforms; this could potentially be due to its higher variability and the potential inclusion of irrelevant information. Despite this, it can slightly enhance performance when paired with action

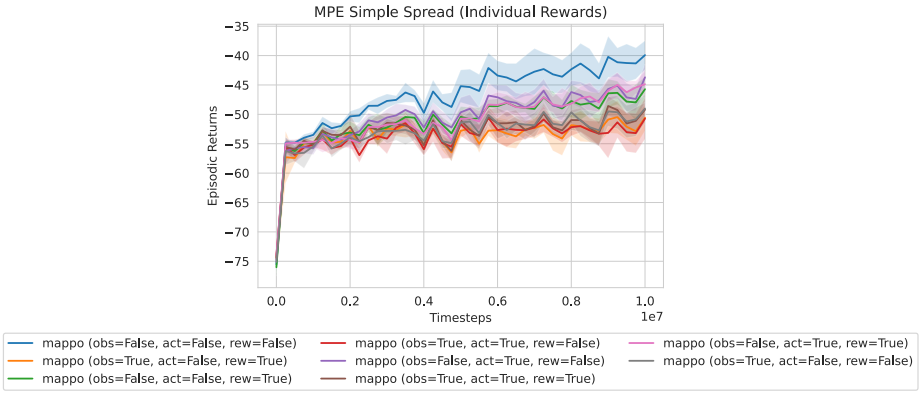

Figure 4.5: Episodic returns comparison of MAPPO with different permutations of reconstruction targets on the MPE Simple Spread environment, with Cross Entropy loss for action frequency reconstruction.

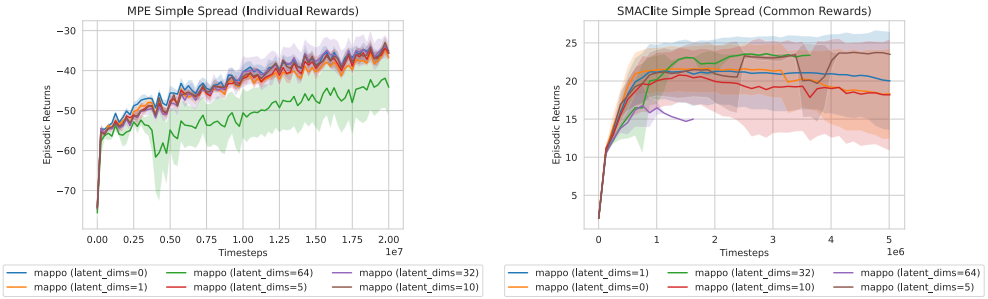

Figure 4.6: Episodic returns comparison of MAPPO with different dimensions of the encoded latent variable, with action frequency reconstruction targets on the MPE Simple Spread and SMAClite MMM2 environments, with KL Divergence loss for the action frequency reconstruction. (Here, environments with a common reward do not include reward reconstruction, as it is not feasible in such cases due to a single reward being shared among all agents.)

reconstruction using cross-entropy loss.

4.3 Ablations on the Latent Variable Dimension

In this set of experiments, the latent variable dimensions were varied to determine the optimal bottleneck size needed to extract opponent strategy information without excessive data loss due to the lossy compression inherent to autoencoder networks. Here, again the reconstruction target has been locked to action frequency with KL divergence loss. In this case, as seen in Figure 4.6, it can be seen that the performance differences across different latent variable dimensions vary wildly with the environment choice, particularly the action dimensions. In the case of MPE Simple Spread, the returns don't seem to vary as much; however, in the case of a very high latent variable dimension, 64 in

this case, the performance seems to deteriorate by a lot. This can be said for the SMAClite MMM2 environment as well, wherein the latent_dim = 64 performs much worse. In this case the consistent better performance is when the latent variable dimension is 32. Although not statistically significant, even in the Simple Spread environment, the better performance is when the latent dimension is around 10. It can be inferred that the latent variable dimensions are better suited to be values less than but within a certain threshold of the value of the combined reconstruction dimensions. The optimal performance is achieved with moderate latent dimensions, approximately smaller than the size of the combined reconstruction dimensions, suggesting that excessive dimensionality can hinder rather than enhance performance. Additionally, it is evident that this aspect—the latent dimension—is heavily influenced by the environmental characteristics and the selection of reconstruction targets.

4.4 Analysis of Autoencoder Learning

When comparing the learning metrics for the autoencoder (Appendix B), a consistent pattern emerged across different configurations. The losses for observation and rewards showed steady learning progress, plateauing relatively quickly. In contrast, the loss for action frequencies tended to increase over time, which was also reflected in a similar trend in accuracy. This behaviour can be attributed to the non-stationary nature of action frequencies, as opponent policies continuously evolve during training, as discussed in Chapter 3. Despite the rising loss in action reconstruction, the episodic returns still improved, especially after a plateauing of the loss, indicating that even though the model struggles to perfectly capture the shifting action distributions, it still effectively leverages the learned representations to enhance overall performance. This suggests that the autoencoder's ability to model opponent behaviour remains beneficial, even when dealing with dynamic and evolving policies, underscoring the robustness of action reconstruction in improving episodic returns despite inherent challenges.

4.5 Computational Overhead

Integrating opponent modelling into MARL algorithms does introduce some computational overhead, but it is minimal compared to the benefits (see Appendix C). The baseline, without agent modelling, is only marginally faster, running 1.21 to 1.27 times quicker than various modelling approaches. Increases in task clock time and CPU

usage are modest, even for the more complex combined reconstruction that includes observations, action frequencies, and rewards of the opponents, indicating that the additional computational demands are manageable (Table 4.1). While page faults vary slightly with complexity, the overall overhead remains minor. This demonstrates that there is no significant disadvantage to utilizing agent modelling, particularly due to the simplicity of the autoencoder network used here. Therefore, the slight improvements are not outweighed by the need for significant computational overhead.

However, it is important to note that since the agent model must predict the characteristics of all opponent agents, the computational complexity is heavily influenced by the number of agents in the environment. While most other environmental factors may result in a similar computational overhead as discussed in this section, the number of agents could lead to additional computational demands, depending on the specific characteristics of the environment.

Configuration	Hyperfine Mean Time	Std Dev	Hyperfine Median Time	Hyperfine Min Time
No Agent Modelling	72.262451	1.656283	72.183126	70.024540
Reward Reconstruction	87.755260	0.910013	87.508237	86.692285
Action Reconstruction	90.456206	0.778696	90.468437	89.656883
Observation Reconstruction	89.454507	0.777903	89.727934	88.649412
Combined Reconstruction	92.047602	0.740381	91.583217	91.470249

Table 4.1: Hyperfine Timing Results across different configurations.

4.6 Key Insights

This chapter has provided a comprehensive evaluation of the autoencoder-based opponent modelling approach, revealing several key insights into its effectiveness across different environments and configurations.

Environment Complexity Matters: The impact of opponent modelling is most pronounced in more complex environments, such as SMAClite MMM2, where nuanced decision-making and strategic coordination are required. In these scenarios, opponent modelling significantly enhances performance by enabling better anticipation and adaptation to the opponent's strategies. However, in simpler environments, the benefits are less evident, with performance often plateauing at levels similar to those achieved without opponent modelling.

Reconstruction Target Selection: The experiments demonstrated that combining action and reward reconstructions generally leads to the most significant performance improvements. Action reconstruction captures short-term behaviours, while reward

reconstruction captures long-term behaviours, creating a more holistic understanding of opponent strategies. Observation reconstruction, while sometimes beneficial when paired with action reconstruction, tends to underperform due to its potential to introduce noise and variability.

Latent Dimension Calibration: The results underscore the importance of selecting an appropriate latent dimension size. Moderate latent dimensions (e.g., 10 in simpler environments and 32 in more complex ones) were found to be optimal, effectively balancing the need for rich representation without overwhelming the model with unnecessary complexity. Excessively large latent dimensions can lead to a deterioration in performance, particularly in simpler environments like MPE Simple Spread.

Loss Function Impact: The choice of loss function for reconstructing action frequencies plays a crucial role. KL Divergence was found to be more effective than cross-entropy in capturing the nuances of probabilistic distributions and aligning the model more closely with opponent strategies.

These insights provide a focused roadmap for optimizing agent modelling in MARL algorithms. They highlight the importance of tailoring the modelling approach to the specific environment and task at hand, ensuring that reconstruction targets, latent dimensions, and loss functions are carefully selected to maximize performance while minimizing computational overhead.

Chapter 5

Conclusion

This dissertation set out to investigate the integration of agent modelling techniques within Multi-Agent Reinforcement Learning (MARL) algorithms, with a particular focus on the effectiveness of autoencoder-based modelling approaches. Through a detailed exploration of various modelling techniques, experimental setups across different environments, and an extensive evaluation of key metrics, this study provides a robust understanding of how autoencoder-reconstruction-based agent modelling can be utilized to enhance MARL performance.

5.1 Summary of Findings

The research conducted in this dissertation can be summarised across several key areas:

Effectiveness of Autoencoder-Based Modelling: The results indicate that autoencoder-based agent modelling resulted in improvements in the episodic returns of the MAPPO algorithm, particularly in complex and strategic environments such as SMAClite MMM2. However, it is important to note that in simpler environments, agent modelling does not offer any additional performance improvements over the baseline MARL performance without agent modelling. The ability of autoencoders to encode and reconstruct various aspects of opponent behaviour, such as action frequencies and reward signals, allows for a more nuanced and effective opponent modelling process. This finding aligns with the hypothesis that capturing both short-term and long-term strategies is essential for robust agent modelling.

Impact of Environment Complexity: The experiments revealed that the benefits of agent modelling are most pronounced in environments with higher complexity and strategic depth. In scenarios like SMAClite MMM2, where agents need to coordinate

diverse unit types and manage intricate tactical decisions, the use of opponent modelling resulted in improvements in episodic returns. Conversely, in simpler environments such as MPE Simple Spread, the impact of agent modelling was less apparent, often leading to performance levels similar to those of non-agent modelling baselines. This suggests that the effectiveness of agent modelling is heavily context-dependent, with more challenging environments providing greater opportunities for its application.

Optimization of Reconstruction Targets and Latent Dimensions: A key insight from this study is the importance of selecting appropriate reconstruction targets and latent dimensions. The combination of action and reward reconstruction was consistently found to yield the best performance across different environments. This approach allows the model to capture both immediate and long-term behaviours of opponents, thereby facilitating more informed decision-making by the agents.

The experiments also highlighted the critical role of latent dimension size. Moderate latent dimensions provided a balance between the richness of representation and computational efficiency. Excessively large latent dimensions, on the other hand, were shown to degrade performance, particularly in simpler environments, likely due to overfitting or unnecessary complexity.

Importance of Loss Function Selection: The choice of loss function was found to be a crucial factor in the effectiveness of the autoencoder-based approach. KL Divergence loss, which captures the nuances of probabilistic distributions, was more effective than cross-entropy loss in aligning the model with opponent strategies. This is particularly important when reconstructing action frequencies, as it allows the model to better capture the variability and uncertainty inherent in opponent behaviours.

5.2 Implications for Future Research

The findings from this dissertation have several implications for future research in the field of MARL and agent modelling:

Enhancing Model Complexity and Flexibility: Future research could explore ways to further enhance the flexibility and scalability of autoencoder-based models. This could involve developing more sophisticated architectures that can dynamically adjust their complexity based on the environment's characteristics or the observed behaviour of opponents. Additionally, investigating the integration of other machine learning techniques, such as attention mechanisms or recurrent neural networks, could lead to models that better capture temporal dependencies and complex strategic patterns.

Expanding to Different Types of Environments: While this study focused on a specific set of environments, future work could extend these findings to a broader range of settings, including continuous action spaces, mixed cooperative-competitive environments, and scenarios with more heterogeneous agent populations. Exploring how agent modelling techniques perform in these varied contexts could provide deeper insights into their generalizability and robustness.

Improving Computational Efficiency: Given the computational overhead associated with autoencoder-based modelling, future research should also focus on optimizing these models for greater efficiency. This could involve developing more lightweight architectures or exploring techniques for reducing the frequency of model updates without sacrificing performance. Additionally, investigating the trade-offs between model complexity and computational cost could help identify the optimal balance for different types of MARL applications.

5.3 Conclusion

This dissertation has demonstrated the significant potential of autoencoder-based agent modelling in enhancing the performance of MARL algorithms, particularly in complex and strategic environments. The findings underscore the importance of carefully selecting reconstruction targets, latent dimensions, and loss functions to optimize model effectiveness. Moreover, the research highlights the need for further exploration into more flexible and scalable modelling techniques that can adapt to a wide range of environments and challenges.

By providing a comprehensive evaluation of agent modelling in MARL, this study contributes valuable insights to the field and sets the stage for future research aimed at developing more advanced and efficient multi-agent systems. The integration of sophisticated agent modelling techniques promises to play a crucial role in the continued advancement of MARL, enabling more robust, adaptable, and intelligent multi-agent interactions in increasingly complex environments.

Bibliography

[1] S. V. Albrecht and P. Stone, "Autonomous agents modelling other agents: A comprehensive survey and open problems," *Artificial Intelligence*, vol. 258, pp. 66–95, May 2018, ISSN: 0004-3702. DOI: 10.1016/j.artint.2018.01.002.

[2] G. Papoudakis, F. Christianos, and S. V. Albrecht, "Agent modelling under partial observability for deep reinforcement learning," *Advances in Neural Information Processing Systems*, 2021.

[3] T. Baarslag, M. J. C. Hendrikx, K. V. Hindriks, and C. M. Jonker, "Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 5, pp. 849–898, Sep. 2016, ISSN: 1573-7454. DOI: 10.1007/s10458-015-9309-1.

[4] S. Karpinskyj, F. Zambetta, and L. Cavedon, "Video game personalisation techniques: A comprehensive survey," *Entertainment Computing*, vol. 5, no. 4, pp. 211–218, 2014, ISSN: 1875-9521. DOI: https://doi.org/10.1016/j.entcom.2014.09.002.

[5] R. BELLMAN, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957, ISSN: 00959057, 19435274.

[6] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, W. W. Cohen and H. Hirsh, Eds., San Francisco (CA): Morgan Kaufmann, 1994, pp. 157–163, ISBN: 978-1-55860-335-6. DOI: https://doi.org/10.1016/B978-1-55860-335-6.50027-1.

[7] L. S. Shapley, "Stochastic games*," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953. DOI: 10.1073/pnas.39.10.1095. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.39.10.1095.

[8] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951, ISSN: 0003486X, 19398980.

[9] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024.

[10] D. Carmel and S. Markovitch, "Opponent modeling in multi-agent systems," vol. 1042, Feb. 1997. DOI: 10.1007/3-540-60923-7_18.

[11] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954.

[12] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and Computation*, vol. 75, no. 2, pp. 87–106, 1987, ISSN: 0890-5401. DOI: https://doi.org/10.1016/0890-5401(87)90052-6.

[13] V. Mnih, A. P. Badia, M. Mirza, *et al.*, *Asynchronous methods for deep reinforcement learning*, 2016. arXiv: 1602.01783 [cs.LG].

[14] L. Zintgraf, K. Shiarlis, M. Igl, *et al.*, *Varibad: A very good method for bayes-adaptive deep rl via meta-learning*, 2020. arXiv: 1910.08348 [cs.LG].

[15] A. van den Oord, Y. Li, and O. Vinyals, *Representation learning with contrastive predictive coding*, 2019. arXiv: 1807.03748 [cs.LG].

[16] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.

[17] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," *arXiv preprint arXiv:1703.04908*, 2017.

[18] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Neural Information Processing Systems (NIPS)*, 2017.

[19] F. Christianos, L. Schäfer, and S. V. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] A. Michalski, F. Christianos, and S. V. Albrecht, *Smaclite: A lightweight environment for multi-agent reinforcement learning*, 2023. arXiv: 2305.05566 [cs.LG].

[21] M. Samvelyan, T. Rashid, C. S. de Witt, *et al.*, "The StarCraft Multi-Agent Challenge," *CoRR*, vol. abs/1902.04043, 2019.

[22] C. Yu, A. Velu, E. Vinitsky, *et al.*, *The surprising effectiveness of ppo in cooperative, multi-agent games*, 2022. arXiv: 2103.01955 [cs.LG].

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: 1707.06347 [cs.LG].

[24] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY, USA: Springer New York Inc., 2001.

Appendix A

Agent Modelling Comprehensive Classification Table

Method	Summary	Advantages	Disadvantages
Policy Reconstruction	Directly reconstructs the conditional action probabilities of opponents. Directly learns the model based on the observed opponent actions.	Simple learning structure, and can be directly learned during the agent interactions.	May require multiple observations to converge and could be more complex due to the short-term focus on single actions and the shifting distribution of actions in the case of dynamic opponent models.
Type-based Reasoning	A Bayesian approach to policy reconstruction, updating beliefs based on new information. Assumes the opponent fits into one of several types and calculates how likely each type is based on the observed action probabilities.	Can lead to faster adaptation if the models are split into appropriate types and the agents are suitably classified by the model.	In cases where faulty classification is made or flawed assumptions are made, the model may make incorrect predictions of the action frequencies.
Classification	Model simply predicts a class label for the opponent. The model can be trained on pre-determined class labels based on prior information.	Simple classification problem, and can learn to predict multiple different types of properties.	Model is generally computed before interactions, and classes are pre-defined, making it difficult to update during interactions.
Plan Recognition	Model predicts goals and future actions of opponents, using a hierarchical plan library or domain model. Can be thought of as preference estimation as mentioned in Section 2.3.2.	Useful for capturing the opponent's long-term planning. Can encode complex plans depending on the plan library.	Doesn't account for the presence of agent modeling on the opponent's side. Requires additional hyperparameter tuning or selection, such as specifying the plan library.
Recursive Reasoning	Policy reconstruction approach, but recursively simulates the reasoning of the opponent model.	Model can make more accurate predictions in complex scenarios where agents' actions depend on their beliefs about each other.	Computationally expensive, and assumes a rational modeled agent, with little scope for randomness.
Graphical Models	Policy reconstruction approach, but represents opponents' preferences and decision processes using a graphical model.	Can easily infer the opponent's MDP, allowing the model to quickly assess different scenarios and outcomes.	As episodes increase sequentially, the model graph complexity may increase exponentially, making it less scalable.
Group Modeling	Predicts the joint properties of a group of agents. Can be similar to other methods, but instead of predicting for a single agent, it predicts for the entire group.	Can capture correlations within the group, and improve efficiency using the group structure.	Reasoning is harder due to complex interdependencies within the group.

Table A.1: Summary of Agent Modeling Methods [1]

Appendix B

AutoEncoder Learning Plots

The analysis of the autoencoder learning metrics is presented through a series of plots that track loss and accuracy across different configurations in the MAPPO algorithm on the MPE Simple Spread environment. The first plot (Figure B.1) represents the observation loss, which demonstrates a steady decrease over time, eventually plateauing, indicating that the autoencoder effectively learns to reconstruct observations. Similarly, the reward reconstruction loss (Figure B.2) follows a comparable pattern, with the model quickly converging to a stable level of performance.

In contrast, the action reconstruction loss (Figure B.3) shows a different trajectory. Unlike the observation and reward losses, the action loss increases over time, which corresponds with the nature of action frequencies being non-stationary as opponent strategies evolve during training. This increase in action loss is also reflected in the accuracy metric (Figure B.4), where accuracy trends tend to fluctuate, further underscoring the challenge of modelling dynamic opponent behaviours.

Despite the challenges in accurately modelling action distributions, the episodic returns, as discussed in the main text, still exhibit improvement. This suggests that while the model may not perfectly capture every nuance of the opponents' strategies, the learned representations are sufficiently robust to contribute positively to the overall performance of the MARL system.

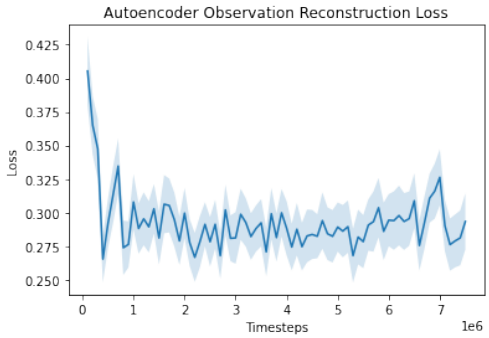

Figure B.1: Observation Reconstruction Loss over Time

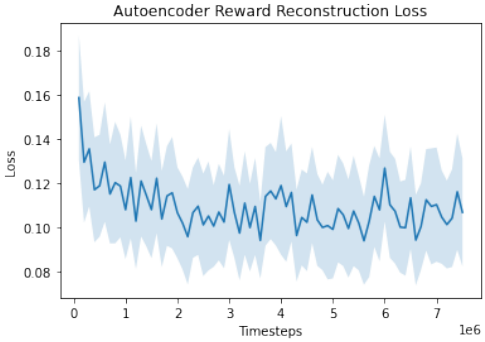

Figure B.2: Reward Reconstruction Loss over Time

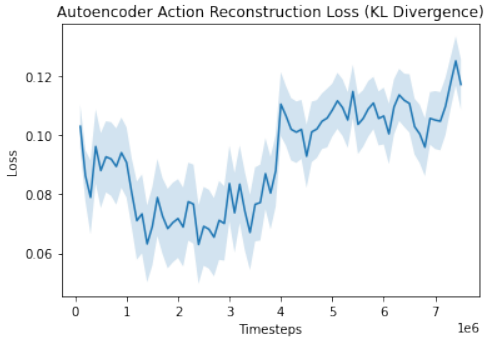

Figure B.3: Action Reconstruction Loss over Time

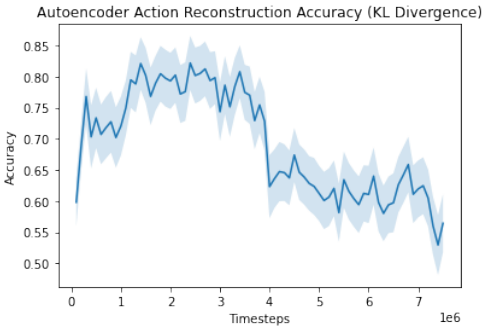

Figure B.4: Action Reconstruction Accuracy over Time

Appendix C

Computational Overhead Analysis

The following section presents an analysis of the computational overhead associated with integrating various forms of opponent modelling on the MAPPO algorithm, tested on the MPE Simple Spread environment. The figures below display the mean execution time, task clock time, CPU utilization, and page faults across different configurations: no agent modelling, reward reconstruction, action reconstruction, observation reconstruction, and combined reconstruction. These metrics help illustrate the relative computational cost of each approach.

Figure C.1 shows the mean time taken for execution, highlighting that while the baseline without agent modelling is the fastest, the differences in execution time for the various opponent modelling methods are minimal. Task clock time, depicted in Figure C.2, increases modestly with more complex reconstructions, particularly in combined reconstruction. Overall, these results demonstrate that the overhead introduced by opponent modelling is manageable and does not impose a significant computational burden.

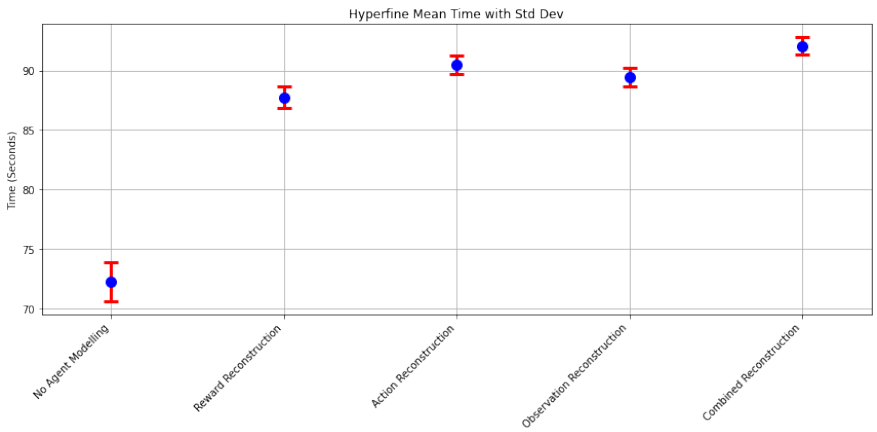

Figure C.1: Hyperfine Mean Time with Standard Deviation across different configurations.

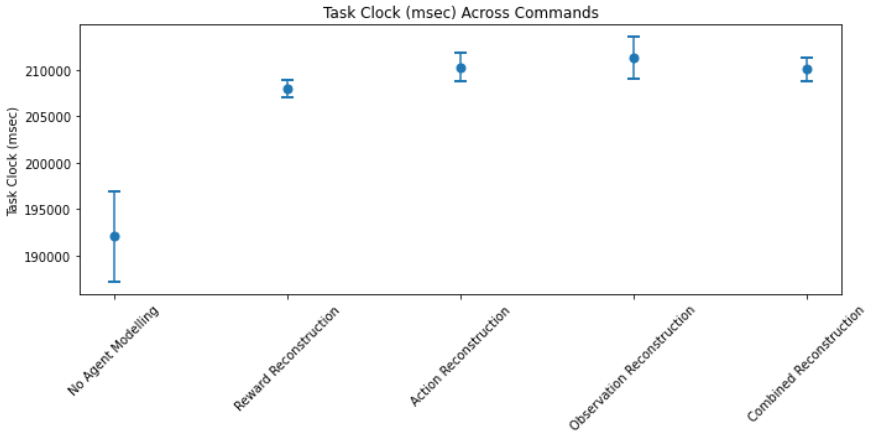

Figure C.2: Task Clock (msec) across different configurations.

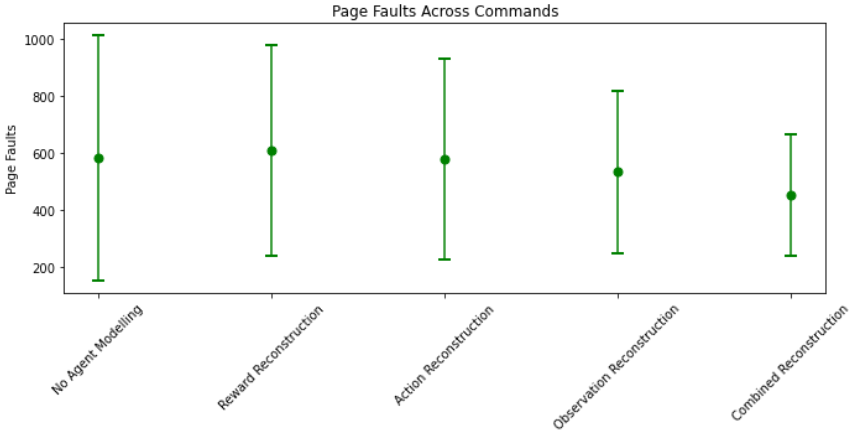

Figure C.3: Page Faults across different configurations.

Appendix D

Hyperparameter Configuration

This appendix provides the detailed hyperparameter settings used for the experiments in this study. The configurations presented here pertain to the various components of the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm, including the opponent modelling settings.

Parameter	**Value**
action_selector	soft_policies
mask_before_softmax	True
buffer_size	20
batch_size_run	20
batch_size	20
target_update_interval_or_tau	0.01
lr	0.0003
hidden_dim	128
obs_agent_id	True
obs_last_action	False
obs_individual_obs	False
agent_output_type	pi_logits
entropy_coef	0.001
use_rnn	True
standardise_returns	False
standardise_rewards	True
q_nstep	5
epochs	4
eps_clip	0.2
t_max	20050000

Table D.1: MAPPO Algorithm Hyperparameters

Parameter	Value
opponent_modelling	True or False
opponent_model_decode_obs	True or False
opponent_model_decode_actions	True or False
opponent_model_decode_rewards	True or False
latent_dims	[0-64]
batch_size_opponent_modelling	64
lr_opponent_modelling	0.0005
opponent_model_epochs	10

Table D.2: Opponent Model Hyperparameters