
Cooperative Multi-Agent Planning with Adaptive Skill Synthesis

Zhiyuan Li

Department of Electrical Engineering and Automation
Aalto University
zhiyuan.li@aalto.fi

Wenshuai Zhao

Department of Electrical Engineering and Automation
Aalto University
wenshuai.zhao@aalto.fi

Joni Pajarinen

Department of Electrical Engineering and Automation
Aalto University
joni.pajarinen@aalto.fi

Abstract

Despite much progress in training distributed artificial intelligence (AI), building cooperative multi-agent systems with multi-agent reinforcement learning (MARL) faces challenges in sample efficiency, interpretability, and transferability. Unlike traditional learning-based methods that require extensive interaction with the environment, large language models (LLMs) demonstrate remarkable capabilities in zero-shot planning and complex reasoning. However, existing LLM-based approaches heavily rely on text-based observations and struggle with the non-Markovian nature of multi-agent interactions under partial observability. We present COMPASS, a novel multi-agent architecture that integrates vision-language models (VLMs) with a dynamic skill library and structured communication for decentralized closed-loop decision-making. The skill library, bootstrapped from demonstrations, evolves via planner-guided tasks to enable adaptive strategies. COMPASS propagates entity information through multi-hop communication under partial observability. Evaluations on the improved StarCraft Multi-Agent Challenge (SMACv2) demonstrate COMPASS's strong performance against state-of-the-art MARL baselines across both symmetric and asymmetric scenarios. Notably, in the symmetric Protoss 5v5 task, COMPASS achieved a 57% win rate, representing a 30 percentage point advantage over QMIX (27%). Project page can be found at <https://stellar-entremet-1720bb.netlify.app/>.

1 Introduction

A major long-term goal for the field of cooperative multi-agent systems (MAS), e.g. multi-robot control Gao et al. (2024); Feng et al. (2024), power management Monroc et al. (2024), and multi-agent games Samvelyan et al. (2019); Kurach et al. (2020), is to build a protocol of collaboration among the agents. Multi-agent reinforcement learning (MARL), proven as an advanced paradigm of distributed artificial intelligence (AI), holds promise for discovering collective behavior from interactions. One line of works follows centralized training decentralized execution (CTDE) paradigm Rashid et al.

(2020); Yu et al. (2022); Liu et al. (2024); Lowe et al. (2017); Zhu et al. (2025); Li et al. (2024b). CTDE assumes a central controller that exploits global information, while the individual policies are designed to allow for decentralized execution. However, in many real-world scenarios, the central controller becomes unfeasible due to the communication overhead that exponentially scales with the number of agents, thereby compromising the scalability of MAS. In contrast, the decentralized training decentralized execution paradigm (DTDE) discards this assumption and is scalable to large-scale systems Su & Lu (2024); Zhang et al. (2018); Ma et al. (2024a). However, DTDE requires complicated learning and planning under uncertainty, as partial observability magnifies the discrepancy between each agent's local observation and global information. Although much progress has been made, MARL suffers from compromised sample efficiency, interpretability, and transferability.

The emergence of Large Language Models (LLMs) has revitalized this field. LLM-based multi-agents have been proposed to leverage their remarkable capacity to perform task-oriented collective behaviors Mandi et al. (2024); Zhang et al. (2024); Gong et al. (2024); Zhang et al. (2025); Nayak et al. (2025). The LLMs are used for high-level planning to generate centralized Gong et al. (2024); Nayak et al. (2025); Deng et al. (2024) or decentralized plans Mandi et al. (2024); Zhang et al. (2024), often adopting a hierarchical decision-making structure in conjunction with a pre-defined low-level controller. While these methods have succeeded in a set of multi-agent problems including Overcooked-AI Carroll et al. (2019), SMAC Samvelyan et al. (2019), and VirtualHome Puig et al. (2018), heavy reliance on text-based observation prevents them from learning from multi-modal information. Moreover, they ignore the non-Markovian nature of MAS, where learning and planning necessitate a decentralized closed-loop solution.

In this paper, we mitigate the existing research limitations and advance general decision-making for cooperative multi-agent systems. At a high level, COMPASS combines a vision language models (VLMs)-based planner with a dynamic skill library for storing and retrieving complex behaviors, along with a structured communication protocol. A diagram of COMPASS is provided in Figure. 1. Inspired by Cradle Tan et al. (2024), the VLM-based planner perceives the visual and textual observation and suggests the most suitable executable code from the skill library. We adopt the code-as-policy paradigm Wang et al. (2024d) instead of task-specific primitive actions, as it constrains generalizability and fails to fully leverage foundation models' extensive world knowledge and sophisticated reasoning capabilities.

Traditional open-loop methods struggle to produce effective plans that adapt to dynamics in stochastic, partially observable environments. To address this challenge, the VLM-based planer attempts to solve challenging and ambiguous final tasks, such as "Defeat all enemy units in the StarCraft multi-agent combat scenario while coordinating with allied units", by progressively proposing a sequence of clear, manageable sub-tasks while incorporating environmental feedback and task progress. COMPASS generates Python scripts through LLMs as semantic-level skills to accomplish sub-tasks, incrementally building a skill library throughout the task progress. Each skill is indexed through its documentation embeddings, enabling retrieval based on task-skill relevance. However, developing the skill library from scratch requires extensive exploration to discover viable strategies. In contrast, we pre-collect demonstration videos and introduce a "warm start" by initializing the skill library with strategies derived from the expert-level dataset.

Moreover, building autonomous agents to cooperate in completing tasks under partial observation requires an efficient communication protocol. However, naive communication leads to the risks of hallucination caused by meaningless chatter between agents Li et al. (2024a). Inspired by entity-based MARL Iqbal et al. (2021); Ding et al. (2023), we present a structured communication protocol to formulate the communication among agents along with a global memory that allows all agents to retrieve. The protocol incorporates a multi-hop propagation mechanism, enabling agents to infer information about entities beyond their field of view through information shared by teammates. Similar to previous approaches, each agent maintains a local memory to preserve current and historical experiences.

Empirically, COMPASS demonstrates effective adaptation and skill synthesis in cooperative multi-agent scenarios. Through its dynamic skill library, it creates reusable and interpretable code-based behaviors that evolve during task execution. We evaluate COMPASS systematically in the improved StarCraft Multi-Agent Challenge (SMACv2) using both open-source (Qwen2-VL-72B Wang et al.

(2024c)) and closed-source (GPT-4o-mini¹, Claude-3-Haiku²) VLMs. COMPASS achieves strong results in Protoss scenarios with a 57% win rate, substantially outperforming state-of-the-art MARL algorithms, including QMIX Rashid et al. (2020), MAPPO Yu et al. (2022), HAPPO Kuba et al. (2022), and HASAC Liu et al. (2024). COMPASS maintains moderate performance in Terran scenarios and handles asymmetric settings effectively, though showing limited success in Zerg task. We evaluate the contribution of individual COMPASS components to its overall performance. We further demonstrate COMPASS’s ability to bootstrap effective strategies from expert demonstrations.

2 Related Work

2.1 Agents in the StarCraft Multi-Agent Challenge

SMAC Samvelyan et al. (2019), a predominant cooperative MARL benchmark based on the StarCraft II real-time strategy game Vinyals et al. (2017), focuses on decentralized micromanagement scenarios where each unit operates under decentralized execution with partial observability to defeat enemy units controlled by Starcraft II’s built-in AI opponent. Previous research in SMAC resort to MARL which can be divided into two categories: 1) Online MARL: One line of representative research is value decomposition (VD) Rashid et al. (2020); Wang et al. (2021); Son et al. (2019), which decomposes the centralized action-value function into individual utility functions. On the other hand, multi-agent policy gradient (MAPG) methods Yu et al. (2022); Kuba et al. (2022); Liu et al. (2024); Li et al. (2023); Wen et al. (2022); Hu et al. (2024); Na et al. (2024) extend single-agent policy gradient algorithm to multi-agent with coordination modeling. Researches such as MAPPO Yu et al. (2022), HAPPO Kuba et al. (2022), and HASAC Liu et al. (2024) combine trust region and maximum entropy with MARL in a non-trivial way respectively. To encourage coordination, communication methods Hu et al. (2024); Lo et al. (2024), sequential modeling methods Li et al. (2023); Wen et al. (2022), and cooperative exploration methods Mahajan et al. (2019); Na et al. (2024) have been proposed. 2) Offline MARL: Recent efforts such as MADT Meng et al. (2023), ODIS Zhang et al. (2022), and MADiff Zhu et al. (2025) leverage data-driven training via pre-collected offline datasets to enhance policy training efficiency. However, the near-optimal performance of these existing approaches on SMAC highlights the benchmark’s limited stochasticity and partial observability. To address these limitations, SMACv2 Ellis et al. (2024) introduces more complexity to necessitate decentralized closed-loop control policies. There have been some recent attempts Li et al. (2024b); McClellan et al. (2024); Formanek et al. (2023); Li et al. (2024c) to evaluate MARL algorithms on SMACv2, and the results confirm the complexity. However, current learning-based multi-agent methods are computationally inefficient and non-interpretable. In the quest to find methods that are sample-efficient and interpretable, LLM-SMAC Deng et al. (2024) leverage LLMs to generate centralized decision tree code under global information in an open-loop framework. Unlike prior works, COMPASS integrates a Vision-Language Model (VLM) with each agent in a decentralized closed-loop manner under partial observability, improving both real-world applicability and scalability.

2.2 LLM-based Multi-Agent System

Based on the inspiring capabilities of LLMs, such as zero-shot planning and complex reasoning Kojima et al. (2024); Zhao et al. (2024); Wei et al. (2024); Besta et al. (2024), embodied single-agent researches have demonstrated the effectiveness of LLMs in solving complex long-horizon tasks Wang et al. (2024b); Yao et al. (2023); Shinn et al. (2023); brian ichter et al. (2022); Ma et al. (2024b); Wang et al. (2024d); Tan et al. (2024). Despite significant advances in single-agent applications, developing real-world multi-agent systems with foundation models remains challenging, primarily due to the nature of decentralized control under partial observability in multi-agent settings Pajarinen & Peltonen (2011). Most prior efforts Mandi et al. (2024); Zhang et al. (2024, 2025); Nayak et al. (2025); Gong et al. (2024) leverage a hierarchical framework with components like perception, communication, planning, execution, and memory to build multi-agent systems with collective behaviors. These approaches can be roughly classified into two groups. 1) Centralized plan: MindAgent Gong et al. (2024) adopts a centralized planning scheme with a pre-defined oracle in a fully observable multi-agent game. LLaMAR Nayak et al. (2025) employs LLMs to manage

¹<https://platform.openai.com/docs/models#gpt-4o-mini>

²<https://www.anthropic.com/news/clause-3-haiku>

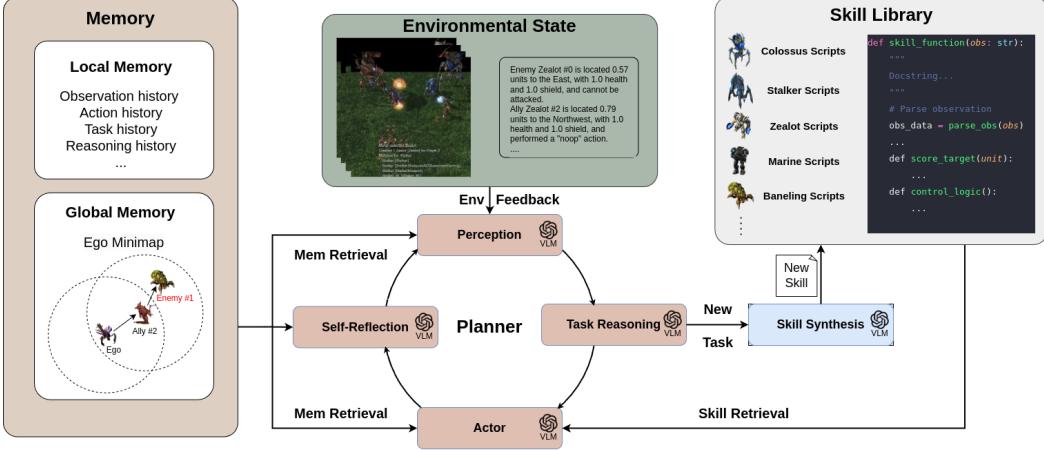


Figure 1: Overview of the COMPASS architecture, a novel framework that advances cooperative multi-agent decision-making through three synergistic components: (1) A VLM-based closed-loop planner that enables decentralized control by continuously processing multi-modal feedback and adapting strategies, addressing the non-Markovian challenge of multi-agent systems; (2) A dynamic skill synthesis mechanism that combines demonstration bootstrapping with incremental skill generation, improving sample efficiency and interpretability; and (3) A structured communication protocol that facilitates efficient information sharing through entity-based multi-hop propagation, enhancing cooperative perception under partial observability.

long-horizon tasks in partially observable environments without assumptions about access to perfect low-level policies. 2) Decentralized plan: ProAgent Zhang et al. (2025) introduces Theory of Mind (ToM), enabling agents to reason about others' mental states. RoCo Mandi et al. (2024) and CoELA Zhang et al. (2024) assign separate LLMs to each embodied agent for collaboration with communication. However, RoCo and CoELA assume a skill library with a low-level heuristic controller, which is impractical in real-world applications. Moreover, RoCo's open-loop plan-and-execute paradigm fails to incorporate environmental feedback during decision-making. In contrast, our work does not assume any pre-defined low-level controller and generates code-based action through VLMs in a closed-loop manner.

3 Preliminaries

We model a fully cooperative multi-agent game with N agents as a *decentralized partially observable Markov decision process* (Dec-POMDP) Oliehoek & Amato (2016), which is formally defined as a tuple $\mathcal{G} = (\mathcal{N}, \mathcal{S}, \mathcal{O}, \mathbb{O}, \mathcal{B}, \mathcal{A}, \mathcal{T}, \Omega, R, \gamma, \rho_0)$. $\mathcal{N} = \{1, \dots, N\}$ is a set of agents, $s \in \mathcal{S}$ denotes the state of the environment and ρ_0 is the distribution of the initial state. $\mathcal{A} = \prod_{i=1}^N A^i$ is the joint action space, $\mathbb{O} = \prod_{i=1}^N O^i$ is the set of joint observations. At time step t , each agent i receives an individual partial observation $o_t^i \in O^i$ given by the observation function $\mathcal{O} : (a_t, s_{t+1}) \mapsto P(o_{t+1}|a_t, s_{t+1})$ where a_t, s_{t+1} and o_{t+1} are the joint actions, states and joint observations respectively. Each agent i uses a stochastic policy $\pi^i(a_t^i|h_t^i, \omega_t^i)$ conditioned on its action-observation history $h_t^i = (o_0^i, a_0^i, \dots, o_{t-1}^i, a_{t-1}^i)$ and a random seed $\omega_t^i \in \Omega_t$ to choose an action $a_t^i \in A^i$. A belief state b_t is a probability distribution over states at time t , where $b_t \in \mathcal{B}$, and \mathcal{B} is the space of all probability distributions over the state space. Actions a_t drawn from joint policy $\pi(a_t|s_t, \omega_t)$ conditioned on state s_t and joint random seed $\omega_t = (\omega_1^1, \dots, \omega_t^N)$ change the state according to transition function $\mathcal{T} : (s_t, a_t^1, \dots, a_t^N) \mapsto P(s_{t+1}|s_t, a_t^1, \dots, a_t^N)$. All agents share the same reward $r_t = R(s_t, a_t^1, \dots, a_t^N)$ based on s_t and a_t . γ is the discount factor for future rewards. Agents try to maximize the expected total reward, $\mathcal{J}(\pi) = \mathbb{E}_{s_0, a_0, \dots} [\sum_{t=0}^{\infty} \gamma^t r_t]$, where $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi(a_t|s_t, \omega_t)$.

4 Methods

COMPASS, illustrated in Figure 1, is a decentralized closed-loop framework for cooperative multi-agent systems that continuously incorporate environmental feedback for strategy refinement. The architecture comprises three core components: 1) a VLM-based closed-loop planner that iteratively perceives, reasons, reflects and acts to adaptively complete tasks (Sec. 4.1); 2) an adaptive skill synthesis mechanism for generating executable codes tailored to proposed sub-tasks (Sec. 4.2); and 3) a structured communication protocol that enables agents to share visible entity information under partial observability (Sec. 4.3). The pseudo-code of COMPASS is shown in Appendix.

4.1 VLM-based Closed-Loop Planner

Inspired by recent advances in cognitive architectures for autonomous systems Tan et al. (2024), COMPASS implements a sophisticated modular planning framework that emulates key aspects of cognitive decision-making. The planner adopts a modular formulation, utilizing four specialized models: Perception, Task Reasoning, Self-Reflection, and Actor. Each model fulfills a distinct yet interconnected role in the decision-making process. The Perception model processes multi-modal inputs, integrating both visual and textual information to build comprehensive environmental understanding. The Task Reasoning model analyzes the perceived information to decompose complex objectives into manageable sub-tasks, ensuring systematic progress toward the final goal. The Self-Reflection model continuously evaluates task execution and outcome quality, enabling adaptive behavior refinement. The Actor model translates plans into actions by selecting and executing the most appropriate skills from the skill library. We next discuss the various components in detail:

Perception forms the foundation of COMPASS’s decision-making capabilities by enabling robust multi-modal understanding of complex environments. Solving complex real-world tasks often involves data of multiple modalities Wang et al. (2024a), each contributing unique and complementary information for decision-making. We leverage the VLMs’ ability to fuse and analyze a broader spectrum of data, including text- and image-based environment feedback, to enable agents to sense the surrounding environment. The system’s perception mechanism operates at two levels: direct observation processing and collaborative information synthesis. At the direct level, VLMs process raw inputs to extract meaningful features and relationships from both visual and textual data. At the collaborative level, COMPASS addresses the inherent challenge of partial observability in multi-agent systems through an innovative multi-hop communication protocol (detailed in Sec. 4.3) that enables agents to construct a more holistic understanding of their environment by sharing and aggregating observations. This dual-level perception architecture ensures that each agent maintains both detailed local awareness and broader contextual understanding, essential for effective decision-making in complex cooperative tasks.

Task Reasoning enables COMPASS to systematically approach complex cooperative challenges through collective task decomposition. Given a simple general final task in the cooperative multi-agent setting, e.g., “defeat all enemy units”, in order to complete the task more efficiently, agents are required to decompose it into multiple sub-tasks and figure out the right one to focus on, while considering alignment among others (See Figure 2). COMPASS harnesses the power of VLMs to

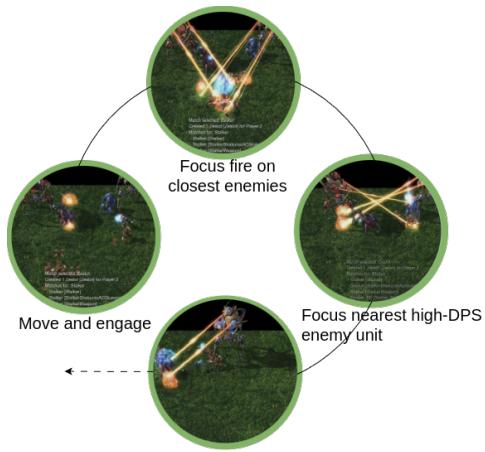


Figure 2: Visualization of COMPASS’s dynamic task reasoning process in the StarCraft Multi-Agent Challenge (SMACv2) environment. The figure demonstrates how the VLM-based planner decomposes a complex final goal (“defeat all enemy units”) into a sequence of concrete, executable sub-tasks that adapt to the changing battlefield conditions. This closed-loop task decomposition enables efficient coordination among multiple agents under partial observability, as each sub-task provides clear, actionable objectives that agents can execute while maintaining overall mission alignment.

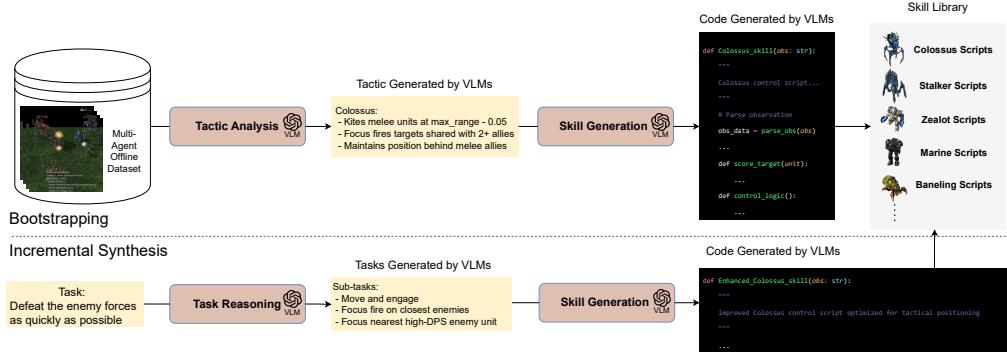


Figure 4: Overview of Adaptive Skill Synthesis. VLMs perform (Top) Bootstrapping by analyzing offline data for initial Tactic Analysis and Skill Generation into a Skill Library. (Bottom) Incremental Synthesis uses Task Reasoning to dynamically generate or enhance code-based skills, evolving the library for new tasks. The skills follow a structured decision-making pipeline with two core components: *score_target(unit)* for dynamic target prioritization and *control_logic()* for coordinating behavior. Textual observations are parsed into structured data (*obs_data*), mapping raw text to attributes, e.g., "Can move North: yes" -> *can_move='north'*: True.

analyze high-level task instructions in conjunction with environmental feedback and team member objectives to generate tractable sub-tasks that collectively advance the overall mission. As agents act under stochastic, partially observable environments, the task reasoning model continuously adapts its plans, proposing and refining sub-tasks based on emerging situations and progress assessment. This dynamic approach enables COMPASS to maintain strategic coherence while adjusting tactical decisions in response to changing circumstances.

Actor serves as the critical bridge between high-level reasoning and concrete action execution. Building upon recent advances in code-writing language models for embodied control Liang et al. (2023); Wang et al. (2024d), the Actor leverages the skill library by first identifying relevant skills for the proposed sub-task, then synthesizes perception and self-reflection inputs to select the optimal skill for execution. This streamlined approach ensures efficient skill selection while maintaining task alignment.

Self-Reflection enables COMPASS to continuously evaluate and refine its decision-making processes through systematic performance analysis (See Figure 3). COMPASS instantiates the Self-Reflection model as a VLM which takes a sequence of visual results from the last skill execution with corresponding descriptions as input to assess the quality of the decision produced by the Actor and whether the task was completed. Additionally, we also request the VLM to generate verbal self-reflections to provide valuable feedback on the completion of the task.

4.2 Adaptive Skill Synthesis

COMPASS employs a dynamic skill library that maintains and evolves a collection of executable behaviors. Each skill is represented as an executable Python function with comprehensive documentation describing its functionality and corresponding

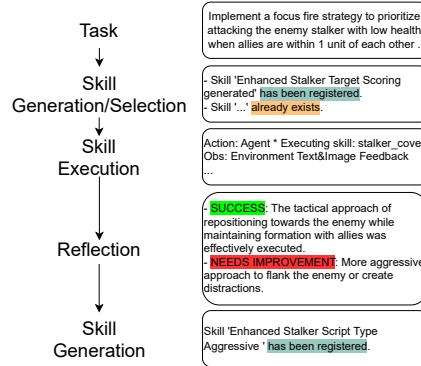


Figure 3: Illustration of self-reflection. Following skill execution (e.g., 'stalker_cover') and feedback, COMPASS assesses performance. This analysis guides further skill generation to refine tactics (e.g., registering an 'Enhanced Stalker Script Type Aggressive').

embedding that enables semantic retrieval. This skill library undergoes continuous refinement through two complementary mechanisms (Figure 4): incremental synthesis, where new skills are generated and existing ones are refined during task execution, and demonstration-based bootstrapping, which initializes the library with behaviors extracted from expert demonstrations. For further details and visualizations of example synthesized tactical behaviors, please refer to the Skill Analysis section in Appendix.

Incremental Synthesis With the Task Reasoning component consistently proposing sub-tasks, COMPASS first attempts to retrieve relevant skills from the library using semantic similarity between the sub-task description and skill documentation embeddings. If no suitable skill exists, or if existing skills prove inadequate, the VLM generates a new Python script specifically tailored to the sub-task.

Bootstrapping However, developing the skill library from scratch requires extensive interactions with environments, which potentially leads to inefficient learning in the early stages. Inspired by offline MARL approaches Meng et al. (2023); Zhang et al. (2022); Zhu et al. (2025), which leverage pre-collected datasets to enhance sample efficiency, we leverage MAPPO as the behavior policy to collect experiences, which are recorded as video sequences. The VLMs then analyze these demonstrations through a multi-stage process: first identifying key strategic patterns and behavioral primitives, then translating these patterns into executable Python functions with appropriate documentation. This initialization methodology establishes a foundational set of validated skills, substantially reducing the exploration overhead typically required for discovering effective behaviors. The resulting baseline skill library enables efficient task execution from the onset while maintaining the flexibility to evolve through incremental synthesis.

4.3 Structured Communication Protocol

To facilitate effective collaboration under partial observability, recent LLM-based multi-agent work Li et al. (2024a); Zhang et al. (2024) employs conversational framework with unconstrained communication protocol. However, while natural language offers flexibility, unrestricted communication can lead to potential hallucinations caused by ambiguous or irrelevant messages between agents. Drawing from advances in structured communication frameworks Hong et al. (2024) and entity-based MARL Iqbal et al. (2021); Ding et al. (2023), COMPASS implements a hierarchical communication protocol that focuses on efficient entity-based information sharing and multi-hop propagation (Figure 5). Each agent maintains an observation buffer containing information about entities in its field of view. At each timestep, agents share their local observations, which are then aggregated into a global entity memory accessible to all. COMPASS employs a multi-hop communication mechanism to propagate information about distant entities, enabling agents to build a more holistic observation of the environment by leveraging the collective knowledge of the team.

5 Experiments

We conducted a comprehensive experimental evaluation of COMPASS to assess its performance and capabilities in complex multi-agent scenarios. Our evaluation focused on the improved StarCraft Multi-Agent Challenge (SMACv2) Ellis et al. (2024), which provides an ideal testbed for examining

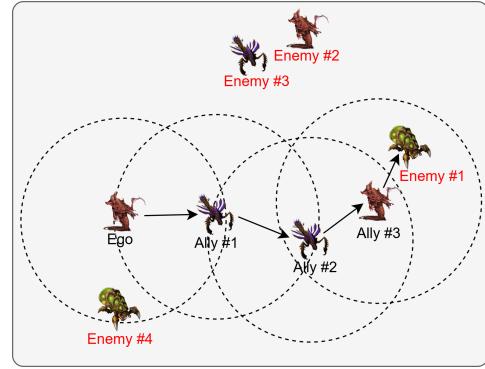


Figure 5: Illustration of COMPASS’s structured multi-hop communication protocol that enables efficient information sharing under partial observability. The figure demonstrates how information about Enemy #1 propagates to the Ego agent through a chain of allied units (Ally #1, #2, #3), despite Enemy #1 being outside Ego’s sight range. Each dashed circle represents an agent’s local observation field, while arrows indicate the flow of entity-based information sharing. This mechanism enables agents to build a more holistic understanding of the environment by propagating critical information (e.g., enemy positions, status) through intermediate allies, effectively addressing the partial observability challenge in decentralized multi-agent systems.

Table 1: Comparative performance of COMPASS (with three VLM variants: G-4o=GPT-4o-mini, C-Hk=Claude-3-Haiku, Q2-VL=Qwen2-VL-72B) and state-of-the-art MARL baselines on SMACv2. Median win rates (%) and standard deviations (subscripts) are reported across Protoss, Terran, and Zerg scenarios in symmetric (5v5) and asymmetric (5v6) categories. Results are averaged over 5 seeds. Bold values denote the best performance in each scenario.

| | QMIX | MAPPO | HAPPO | HASAC | COMPASS | | |
|------------|-----------------------------|-----------------------------|----------------------|----------------------|-----------------------------|----------------------------|----------------------|
| | | | | | G-4o | C-Hk | Q2-VL |
| PROTOSS | | | | | | | |
| SYMMETRIC | 0.27 _{0.03} | 0.32 _{0.067} | 0.34 _{0.07} | 0.20 _{0.08} | 0.57 _{0.08} | 0.49 _{0.06} | 0.45 _{0.04} |
| ASYMMETRIC | 0.01 _{0.01} | 0.04 _{0.04} | 0.02 _{0.03} | 0.01 _{0.02} | 0.08 _{0.04} | 0.06 _{0.05} | 0.06 _{0.03} |
| TERRAN | | | | | | | |
| SYMMETRIC | 0.38 _{0.04} | 0.36 _{0.1} | 0.35 _{0.1} | 0.29 _{0.01} | 0.39 _{0.01} | 0.38 _{0.05} | 0.31 _{0.02} |
| ASYMMETRIC | 0.06 _{0.02} | 0.07 _{0.06} | 0.01 _{0.03} | 0.05 _{0.02} | 0.1 _{0.03} | 0.1 _{0.01} | 0.06 _{0.03} |
| ZERG | | | | | | | |
| SYMMETRIC | 0.21 _{0.01} | 0.27 _{0.04} | 0.2 _{0.11} | 0.24 _{0.07} | 0.16 _{0.07} | 0.18 _{0.02} | 0.14 _{0.03} |
| ASYMMETRIC | 0.18 _{0.03} | 0.13 _{0.09} | 0.09 _{0.02} | 0.08 _{0.05} | 0.03 _{0.01} | 0.04 _{0.01} | 0.02 _{0.01} |

cooperative behavior under partial observability and stochasticity. Through systematic experimentation, we investigated two fundamental questions: (1) How does COMPASS perform compared to state-of-the-art MARL methods? (2) What are the individual contributions of each component in COMPASS? Experiments utilize both open-source (Qwen2-VL-72B) and closed-source VLMs (GPT-4o-mini, Claude-3-Haiku), with Jina AI embeddings for skill retrieval. All results are averaged over 5 seeds to account for environmental stochasticity. Token usage is approximately 0.4 million per episode.

5.1 Experimental Setup

Scenarios Our evaluation scenarios span three distinct race matchups (Protoss, Terran, and Zerg) and two categories (symmetric and asymmetric), as detailed in Appendix. The symmetric scenarios (5v5) test coordination in balanced engagements, while asymmetric scenarios (5v6) evaluate adaptation to numerical disadvantages. Each race combination presents unique tactical challenges due to different unit abilities and constraints. We followed the setting $p=0$ in the SMACv2 original paper (i.e., `prob_obs_enemy: 0.0` in the .yaml file), meaning that only the first agent to initially spot a specific enemy unit can continue observing it, introducing the *Extended Partial Observability Challenge*, which baselines struggled with.

Baselines We compared COMPASS against the state-of-the-art MARL algorithms representing both value-based and policy-gradient approaches:

- Value-Based Methods: QMIX Rashid et al. (2020) uses a mixing network architecture to decompose joint action-values while maintaining monotonicity constraints.
- Policy Gradient Methods: MAPPO Yu et al. (2022) extends PPO to multi-agent settings with the CTDE paradigm. HAPPO Kuba et al. (2022) performs sequential policy updates by utilizing other agents' newest policy under the CTDE framework and provably obtains the monotonic policy improvement guarantee. HASAC Liu et al. (2024) combines the maximum entropy framework with trust region optimization to enhance exploration and coordination.

Datasets To enable effective bootstrapping of the skill library, we constructed a comprehensive demonstration dataset capturing diverse multi-agent strategies and interactions. We employed MAPPO with original hyper-parameters as our behavior policy for data collection, leveraging its strong performance in cooperative multi-agent tasks. Our final dataset comprises over 300 complete game episodes, each recorded as a video sequence capturing the full state-action trajectory. These demonstrations span all symmetric scenario types described in Table 3.

Table 2: Win rates of the initialized skill library (bootstrapped from expert demonstrations) on SMACv2.

| | PROTOSS | TERRAN | ZERG |
|-----|----------------------|----------------------|----------------------|
| 5V5 | 0.35 _{0.06} | 0.24 _{0.04} | 0.06 _{0.01} |
| 5V6 | 0.04 _{0.05} | 0.06 _{0.02} | 0.02 _{0.03} |

5.2 Main Results

Performance As shown in Table 1, COMPASS demonstrates significant performance advantages in SMACv2, particularly excelling in Protoss scenarios where it achieves a 57% win rate in symmetric engagements using GPT-4o-mini, substantially outperforming traditional approaches like QMIX (27%), MAPPO (32%), and HAPPO (34%).

However, performance varies across race matchups. While maintaining strong results in Terran scenarios (39% win rate), COMPASS shows limited effectiveness in Zerg scenarios (16% win rate). This performance disparity can be attributed to the unique mechanics of Zerg combat units, which demand more fine-grained micromanagement due to their shorter attack ranges and reliance on swarm-based tactics.

In asymmetric scenarios (5v6), COMPASS consistently outperforms MARL baselines in Protoss and Terran matchups, demonstrating its ability to execute effective strategies despite being outnumbered. Its success in these settings suggests that COMPASS can adapt dynamically, using coordinated tactics and learned skills to counteract numerical disadvantages. The robust performance holds across different VLM implementations, with GPT-4o-mini consistently achieving the strongest results.

Moreover, COMPASS demonstrates particular advantages in scenarios with sparse reward, where traditional MARL approaches significantly underperform (near-zero win rates).

5.3 Ablation Studies

Skill Initialization To evaluate the impact of our skill initialization, we analyze the performance of COMPASS using only the initialized skill library derived from expert demonstrations. The results in Table 2 demonstrate that skill initialization alone achieves non-trivial performance across different scenarios, particularly in symmetric matchups. Moreover, the gap between initialized skills and COMPASS underscores the necessity of incremental skill synthesis. A script example for skill initialization is in Appendix.

Communication To demonstrate the critical role of communication, we evaluated COMPASS on Protoss 5v5 under the *Extended Partial Observability* setting, using only local information without multi-hop propagation. The resulting win rate with GPT-4o-mini decreased to 0.06_{0.04}, a significant drop from 0.57 with full communication. This degradation occurs because the Extended Partial Observability setting restricts direct enemy visibility to the first agent that initially spots it. The VLMs generated control logic heavily relies on the presence of enemies in the local observation to determine engagement and targeting. Without communication relaying enemy positions, agents other than the discoverer cannot ‘see’ enemies known to teammates, even if within attack range. Consequently, their control logic frequently defaults to ‘no enemy’ behaviors, such as moving towards allies or executing random default actions, preventing effective target engagement and coordinated attacks, thus drastically reducing combat effectiveness and the overall win rate.

Self Reflection In order to show the effectiveness of self-reflection, we evaluate the performance of COMPASS w/o self-reflection on protoss 5 vs 5. Removing the module leads to a drop -10% (0.47_{0.04}).

Visual information We tested visual information contribution by omitting the image inputs. Intuitively, this challenges the agents to rely solely on uni-modal textual information, potentially leading to a loss of spatial understanding. Empirical results show a 10% drop in performance when visuals are omitted. Comparing the VLM outputs shows that without visual information, understanding map boundaries relies on interpreting textual cues (e.g., ‘West (unavailable movement)’), which might be derived indirectly from action availability rather than direct observation. With visual input, the VLM can directly perceive these crucial spatial details like map boundaries from the image itself.

This lack of spatial awareness can result in less informed tactical decisions regarding movement and positioning.

6 Conclusion

We present COMPASS, a novel framework for cooperative multi-agent systems that integrates vision-language models, a dynamic skill library, and structured communication. Through decentralized closed-loop planning, COMPASS enables agents to iteratively decompose tasks and adapt strategies via environmental feedback. Our skill library, initialized from expert demonstrations and refined through execution, provides interpretable code-based behaviors. Our hierarchical communication protocol enhances coordination under partial observability through entity-level information sharing. Evaluations on SMACv2 demonstrate COMPASS’s effectiveness in several scenarios, particularly in Protoss, while highlighting areas for improvement in others like the Zerg setting. These results suggest that COMPASS provides a promising direction for developing interpretable and adaptable multi-agent systems suitable for real-world applications.

References

- Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., and Hoefer, T. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, Mar. 2024. doi: 10.1609/aaai.v38i16.29720. URL <https://ojs.aaai.org/index.php/AAAI/article/view/29720>.
- brian ichter, Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., Kalashnikov, D., Levine, S., Lu, Y., Parada, C., Rao, K., Sermanet, P., Toshev, A. T., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Yan, M., Brown, N., Ahn, M., Cortes, O., Sievers, N., Tan, C., Xu, S., Reyes, D., Rettinghouse, J., Quiambao, J., Pastor, P., Luu, L., Lee, K.-H., Kuang, Y., Jesmonth, S., Jeffrey, K., Ruano, R. J., Hsu, J., Gopalakrishnan, K., David, B., Zeng, A., and Fu, C. K. Do as i can, not as i say: Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=bdHkMjBJG_w.
- Carroll, M., Shah, R., Ho, M. K., Griffiths, T. L., Seshia, S. A., Abbeel, P., and Dragan, A. *On the utility of learning about humans for human-AI coordination*. Curran Associates Inc., Red Hook, NY, USA, 2019.
- Deng, Y., Ma, W., Fan, Y., Zhang, Y., Zhang, H., and Zhao, J. A new approach to solving smac task: Generating decision tree code from large language models, 2024. URL <https://arxiv.org/abs/2410.16024>.
- Ding, Z., Zhang, W., Yue, J., Wang, X., Huang, T., and Lu, Z. Entity divider with language grounding in multi-agent reinforcement learning. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8103–8119. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/ding23d.html>.
- Ellis, B., Cook, J., Moalla, S., Samvelyan, M., Sun, M., Mahajan, A., Foerster, J. N., and Whiteson, S. Smacv2: an improved benchmark for cooperative multi-agent reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS ’23, Red Hook, NY, USA, 2024. Curran Associates Inc.
- Feng, Y., Hong, C., Niu, Y., Liu, S., Yang, Y., Yu, W., Zhang, T., Tan, J., and Zhao, D. Learning multi-agent loco-manipulation for long-horizon quadrupedal pushing, 2024. URL <https://arxiv.org/abs/2411.07104>.
- Formanek, C., Jeewa, A., Shock, J., and Pretorius, A. Off-the-grid marl: Datasets and baselines for offline multi-agent reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’23, pp. 2442–2444, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450394321.

- Gao, J., Wang, Z., Xiao, Z., Wang, J., Wang, T., Cao, J., Hu, X., Liu, S., Dai, J., and Pang, J. Coohoi: Learning cooperative human-object interaction with manipulated object dynamics, 2024. URL <https://arxiv.org/abs/2406.14558>.
- Gong, R., Huang, Q., Ma, X., Noda, Y., Durante, Z., Zheng, Z., Terzopoulos, D., Fei-Fei, L., Gao, J., and Vo, H. MindAgent: Emergent gaming interaction. In Duh, K., Gomez, H., and Bethard, S. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3154–3183, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.200. URL <https://aclanthology.org/2024.findings-naacl.200/>.
- Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., and Schmidhuber, J. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=VtmBAGCN7o>.
- Hu, S., Shen, L., Zhang, Y., and Tao, D. Learning multi-agent communication from graph modeling perspective. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Qox9r00kN0>.
- Iqbal, S., De Witt, C. A. S., Peng, B., Boehmer, W., Whiteson, S., and Sha, F. Randomized entity-wise factorization for multi-agent reinforcement learning. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4596–4606. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/iqbal21a.html>.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS ’22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088.
- Kuba, J. G., Chen, R., Wen, M., Wen, Y., Sun, F., Wang, J., and Yang, Y. Trust region policy optimisation in multi-agent reinforcement learning, 2022. URL <https://arxiv.org/abs/2109.11251>.
- Kurach, K., Raichuk, A., Stanczyk, P., Zajac, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., and Gelly, S. Google research football: A novel reinforcement learning environment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4501–4510, Apr. 2020. doi: 10.1609/aaai.v34i04.5878. URL <https://ojs.aaai.org/index.php/AAAI/article/view/5878>.
- Li, C., Liu, J., Zhang, Y., Wei, Y., Niu, Y., Yang, Y., Liu, Y., and Ouyang, W. Ace: Cooperative multi-agent q-learning with bidirectional action-dependency. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):8536–8544, Jun. 2023. doi: 10.1609/aaai.v37i7.26028. URL <https://ojs.aaai.org/index.php/AAAI/article/view/26028>.
- Li, G., Al Kader Hammoud, H. A., Itani, H., Khizbulin, D., and Ghanem, B. Camel: communicative agents for "mind" exploration of large language model society. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS ’23, Red Hook, NY, USA, 2024a. Curran Associates Inc.
- Li, Z., Zhao, W., Wu, L., and Pajarinen, J. Backpropagation through agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(12):13718–13726, Mar. 2024b. doi: 10.1609/aaai.v38i12.29277. URL <https://ojs.aaai.org/index.php/AAAI/article/view/29277>.
- Li, Z., Zhao, W., Wu, L., and Pajarinen, J. Agentmixer: Multi-agent correlated policy factorization, 2024c. URL <https://arxiv.org/abs/2401.08728>.
- Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control, 2023. URL <https://arxiv.org/abs/2209.07753>.
- Liu, J., Zhong, Y., Hu, S., Fu, H., FU, Q., Chang, X., and Yang, Y. Maximum entropy heterogeneous-agent reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=tmq0hBC4a5>.

- Lo, Y. L., Sengupta, B., Foerster, J. N., and Noukhovitch, M. Learning multi-agent communication with contrastive learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=vZZ4hhniJU>.
- Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6382–6393, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- Ma, C., Li, A., Du, Y., Dong, H., and Yang, Y. Efficient and scalable reinforcement learning for large-scale network control. *Nature Machine Intelligence*, 6(9):1006–1020, 2024a.
- Ma, W., Mi, Q., Zeng, Y., Yan, X., Wu, Y., Lin, R., Zhang, H., and Wang, J. Large language models play starcraft ii: Benchmarks and a chain of summarization approach, 2024b. URL <https://arxiv.org/abs/2312.11865>.
- Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. *MAVEN: multi-agent variational exploration*. Curran Associates Inc., Red Hook, NY, USA, 2019.
- Mandi, Z., Jain, S., and Song, S. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 286–299, 2024. doi: 10.1109/ICRA57147.2024.10610855.
- McClellan, J., Haghani, N., Winder, J., Huang, F., and Tokekar, P. Boosting sample efficiency and generalization in multi-agent reinforcement learning via equivariance, 2024. URL <https://arxiv.org/abs/2410.02581>.
- Meng, L., Wen, M., Le, C., Li, X., Xing, D., Zhang, W., Wen, Y., Zhang, H., Wang, J., Yang, Y., et al. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 20(2): 233–248, 2023.
- Monroc, C. B., Busic, A., Dubuc, D., and Zhu, J. WFCRL: A multi-agent reinforcement learning benchmark for wind farm control. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL <https://openreview.net/forum?id=ZRMAhpZ3ED>.
- Na, H., Seo, Y., and chul Moon, I. Efficient episodic memory utilization of cooperative multi-agent reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=LjivA1SLZ6>.
- Nayak, S., Orozco, A. M., Have, M. T., Thirumalai, V., Zhang, J., Chen, D., Kapoor, A., Robinson, E., Gopalakrishnan, K., Harrison, J., Ichter, B., Mahajan, A., and Balakrishnan, H. Llamar: Long-horizon planning for multi-agent robots in partially observable environments, 2025. URL <https://arxiv.org/abs/2407.10031>.
- Oliehoek, F. A. and Amato, C. *A Concise Introduction to Decentralized POMDPs*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 3319289276.
- Pajarinen, J. and Peltonen, J. Efficient planning for factored infinite-horizon dec-pomdps. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pp. 325–331. AAAI Press, 2011. ISBN 9781577355137.
- Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020. URL <http://jmlr.org/papers/v21/20-081.html>.
- Samvelyan, M., Rashid, T., Schroeder de Witt, C., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pp. 2186–2188, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450363099.

- Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: language agents with verbal reinforcement learning. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.
- Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5887–5896. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/son19a.html>.
- Su, K. and Lu, Z. A fully decentralized surrogate for multi-agent policy optimization. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=MppUW90uU2>.
- Tan, W., Zhang, W., Xu, X., Xia, H., Ding, Z., Li, B., Zhou, B., Yue, J., Jiang, J., Li, Y., An, R., Qin, M., Zong, C., Zheng, L., Wu, Y., Chai, X., Bi, Y., Xie, T., Gu, P., Li, X., Zhang, C., Tian, L., Wang, C., Wang, X., Karlsson, B. F., An, B., Yan, S., and Lu, Z. Cradle: Empowering foundation agents towards general computer control, 2024. URL <https://arxiv.org/abs/2403.03186>.
- Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittweise, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., van Hasselt, H., Silver, D., Lillicrap, T., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekermo, A., Repp, J., and Tsing, R. Starcraft ii: A new challenge for reinforcement learning, 2017. URL <https://arxiv.org/abs/1708.04782>.
- Wang, C., Hasler, S., Tanneberg, D., Ocker, F., Joublin, F., Ceravola, A., Deigmoeller, J., and Gienger, M. Lami: Large language models for multi-modal human-robot interaction. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI EA ’24, New York, NY, USA, 2024a. Association for Computing Machinery. ISBN 9798400703317. doi: 10.1145/3613905.3651029. URL <https://doi.org/10.1145/3613905.3651029>.
- Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024b. ISSN 2835-8856. URL <https://openreview.net/forum?id=ehfRiFOR3a>.
- Wang, J., Ren, Z., Liu, T., Yu, Y., and Zhang, C. {QPLEX}: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=Rcmk0xxIQV>.
- Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., Fan, Y., Dang, K., Du, M., Ren, X., Men, R., Liu, D., Zhou, C., Zhou, J., and Lin, J. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution, 2024c. URL <https://arxiv.org/abs/2409.12191>.
- Wang, X., Chen, Y., Yuan, L., Zhang, Y., Li, Y., Peng, H., and Ji, H. Executable code actions elicit better llm agents, 2024d. URL <https://arxiv.org/abs/2402.01030>.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS ’22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088.
- Wen, M., Kuba, J., Lin, R., Zhang, W., Wen, Y., Wang, J., and Yang, Y. Multi-agent reinforcement learning is a sequence modeling problem. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 16509–16521. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/69413f87e5a34897cd010ca698097d0a-Paper-Conference.pdf.
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and WU, Y. The surprising effectiveness of ppo in cooperative multi-agent games. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24611–24624. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf.

Zhang, C., Yang, K., Hu, S., Wang, Z., Li, G., Sun, Y., Zhang, C., Zhang, Z., Liu, A., Zhu, S.-C., Chang, X., Zhang, J., Yin, F., Liang, Y., and Yang, Y. Proagent: building proactive cooperative agents with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence, AAAI’24/IAAI’24/EAAI’24*. AAAI Press, 2025. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i16.29710. URL <https://doi.org/10.1609/aaai.v38i16.29710>.

Zhang, F., Jia, C., Li, Y.-C., Yuan, L., Yu, Y., and Zhang, Z. Discovering generalizable multi-agent coordination skills from multi-task offline data. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhang, H., Du, W., Shan, J., Zhou, Q., Du, Y., Tenenbaum, J. B., Shu, T., and Gan, C. Building cooperative embodied agents modularly with large language models, 2024. URL <https://arxiv.org/abs/2307.02485>.

Zhang, K., Yang, Z., Liu, H., Zhang, T., and Basar, T. Fully decentralized multi-agent reinforcement learning with networked agents. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5872–5881. PMLR, 10–15 Jul 2018. URL <https://proceedings.mlr.press/v80/zhang18n.html>.

Zhao, Z., Lee, W. S., and Hsu, D. Large language models as commonsense knowledge for large-scale task planning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, Red Hook, NY, USA, 2024. Curran Associates Inc.

Zhu, Z., Liu, M., Mao, L., Kang, B., Xu, M., Yu, Y., Ermon, S., and Zhang, W. Madiff: Offline multi-agent learning with diffusion models, 2025. URL <https://arxiv.org/abs/2305.17330>.

A Pseudocode

The pseudo-code of the COMPASS algorithm is shown in Pseudocode 1.

Algorithm 1: COMPASS Agent Decision-Making Loop

Initialize:

```
skill_manager.bootstrap(demonstration_data)
agent_state ← environment.reset(agent_id)
local_memory ← initialize_local_memory()
global_memory ← initialize_global_memory()
previous_action_result ← None
while True do
    // 1. Communication Phase
    local_observations ← agent_state.get_observations()
    communication_protocol.share_local_observations(agent_id, local_observations,
                                                    global_memory)
    global_entity_info ← communication_protocol.get_global_memory.update(global_memory)
    // 2. Perception Phase
    processed_state ← vlm_perception.process( raw_observation=agent_state,
                                              communication_data=global_entity_info, local_memory=local_memory )
    local_memory.update(processed_state)
    // 3. Self-Reflection Phase
    if previous_action_result ≠ None then
        reflection_feedback ← vlm_self_reflection.reflect( previous_action_result )
    // 4. Task Reasoning Phase
    sub_task ← vlm_task_reasoning.props_subtask( processed_state, overall_goal,
                                                reflection_feedback )
    // 5. Skill Generation Phase
    new_skill_code ← vlm_skill_generator.generate_skill(sub_task, processed_state)
    skill_manager.add_skill(new_skill_code, sub_task)
    // 6. Actor Phase
    relevant_skills ← skill_manager.retrieve_skills(sub_task)
    chosen_skill_code ← vlm_actor.select_skill( sub_task, relevant_skills, processed_state )
    // 7. Execution Phase
    (next_agent_state, reward, done, info) ← environment.step( agent_id, chosen_skill_code )
    agent_state ← next_agent_state
    previous_action_result ← (chosen_skill_code, info, reward, done)
    local_memory.add_action(chosen_skill_code)
    if done then
        break
```

B Implementation Details

COMPASS integrates VLMs to process multi-modal inputs and generate executable skills in two stages. Each skill follows a standardized interface:

```
1 def skill_template(obs: str):
2     obs_data = parse_obs(obs)
3     def score_target(unit):
4         ...
5             return score
6     def control_logic():
7         ...
8             return atomic_action
```

Listing 1: Interface of Generated skills.

The skills follow a structured decision-making pipeline with two core components: `score_target(unit)` and `control_logic()`:

- `score_target(unit)`: Dynamically calculates a threat/priority score by evaluating unit type, health, distance, formations, and matchups to guide optimal attack/heal targeting decisions.
- `control_logic()`: Dynamically coordinates unit behavior by integrating observations, target priorities, and pathfinding to execute role-optimized strategies (e.g., stalkers attack colossus at max range while moving away from zealots).

COMPASS evolves skills through iterative refinement and task-guided synthesis:

- **iterative refinement**: When errors occur during skill execution, VLMs analyze the error messages and attempt to fix the bugs.
- **task-guided synthesis**: When a new task is proposed, VLMs first determine whether a new skill needs to be generated to align with the task. If necessary, VLMs generate new `score_target` or `control_logic` components to fulfill the task requirements and integrate them with the existing code to construct a new skill.

For example, if the task is: "*Implement an aggressive advance movement pattern for the colossus unit when enemy stalkers are within sight range and allies are positioned to provide covering fire,*" and the current skills in the library are not aggressive enough, the VLMs will refine the `control_logic` to implement a more aggressive behavior pattern.

COMPASS maintains a global entity memory, where agents act as nodes connected if within sight range. Each node stores information about visible allies and enemies. When Agent A observes entities, it propagates updates through adjacent nodes recursively, up to `max_hops` (default = 3). This allows agents to infer off-screen threats via ally intermediaries. For implementation details, see `/common/memory/global_memory.py`.

C Skill Analysis

We now analyze COMPASS’s capability to synthesize and execute diverse tactical behaviors. COMPASS develops four key tactical patterns: (1) An exponentially-scaled focus fire implementation that coordinates multiple units’ target selection based on allied attacker density (Figure 6), (2) A position-aware kiting mechanism that maintains optimal engagement ranges while managing unit positioning relative to threats (Figure 7), (3) A formation-based isolation tactic that enables systematic target elimination through coordinated unit movements (Figure 8), and (4) An area-of-effect (AOE) tactic that maximizes splash damage through cluster density calculation (Figure 9). These synthesized skills exhibit clear strategic intent while maintaining interpretability.

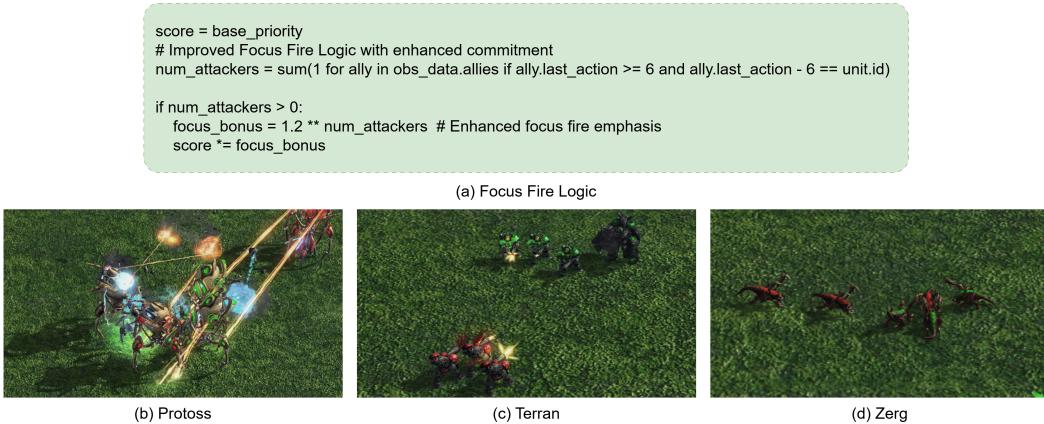


Figure 6: Focus Fire Logic Implementation. (a) VLM-generated Python code snippet implementing dynamic focus fire logic. The code prioritizes enemy units based on the number of allied attackers, scaling the attack bonus exponentially. (b–d) Visualizations of focus fire execution across Protoss, Terran, and Zerg.

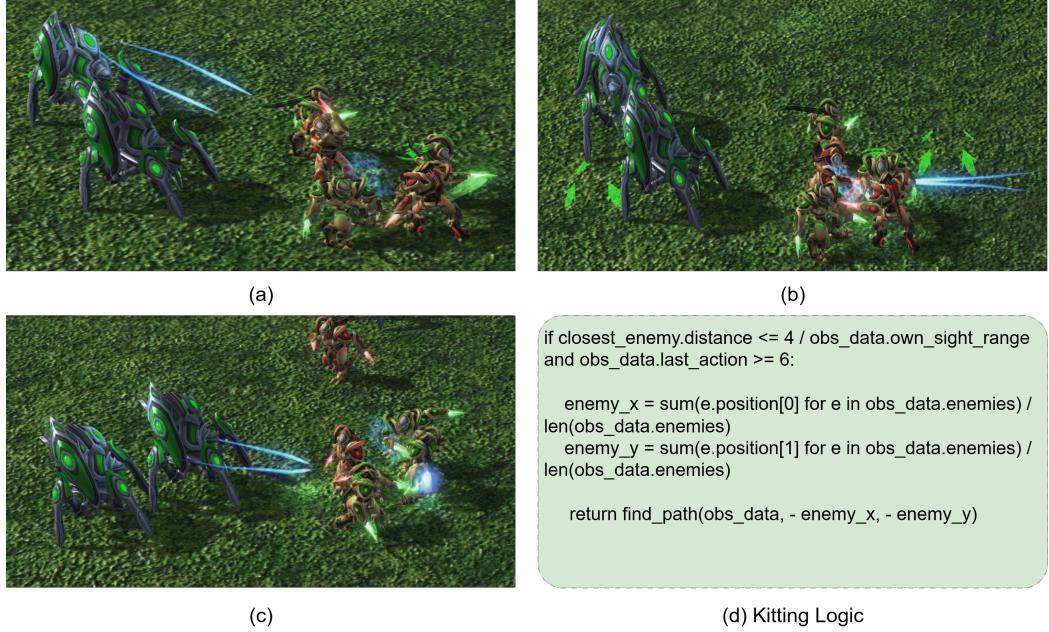


Figure 7: Illustration and implementation of Kitting logic. (a)-(c) demonstrate progressive stages of the kitting tactic where allied units strategically maintain optimal attack range while retreating from melee enemies. (d) shows the corresponding Python code snippet generated by the VLMs.

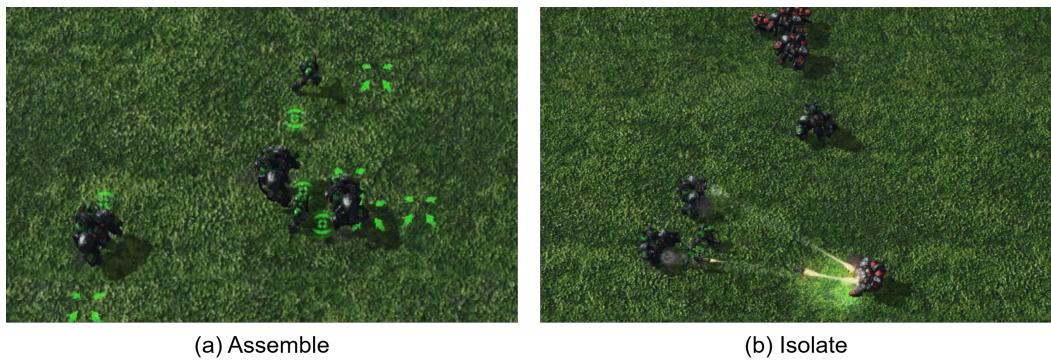


Figure 8: Illustration of Isolating logic. (a) Allied units strategically assemble into a cohesive formation. (b) The assembled units execute a rapid engagement against an isolated enemy unit, eliminating it before reinforcements can arrive, thus creating a numerical advantage.

```

enemy_clusters = {}
for enemy in obs_data.enemies:
    nearby_enemies = []
    # Dynamic cluster radius based on unit type
    cluster_radius = 0.3 if obs_data.own_unit_type.lower() == 'baneling' else 0.2
    for other in obs_data.enemies:
        distance = ((other.position[0] - enemy.position[0])**2 + (other.position[1] - enemy.position[1])**2)**0.5
        if distance <= cluster_radius:
            nearby_enemies.append(other)
    enemy_clusters[enemy.id] = len(nearby_enemies)

```

(a) Baneling Cluster

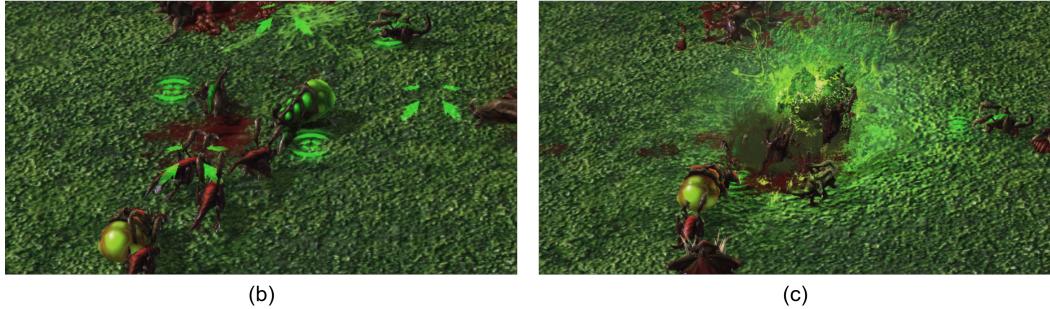


Figure 9: Demonstration of area-of-effect (AOE) optimization for Baneling units in SMACv2. (a) The VLM-generated Python code calculates optimal detonation positions by analyzing enemy cluster density and positions. (b-c) Visual sequence showing Baneling execution, where the unit identifies a dense cluster of enemy units and detonates for maximum AOE damage.

D Baseline training results

For QMIX, we utilized Epmarl³, and for others, we used HARL⁴. Task difficulty was set to 5v5 for SYMMETRIC tasks and 5v6 for ASYMMETRIC tasks; the 5v6 scenario introduces a significant force disadvantage (20% outnumbered vs 10% in 10v11).

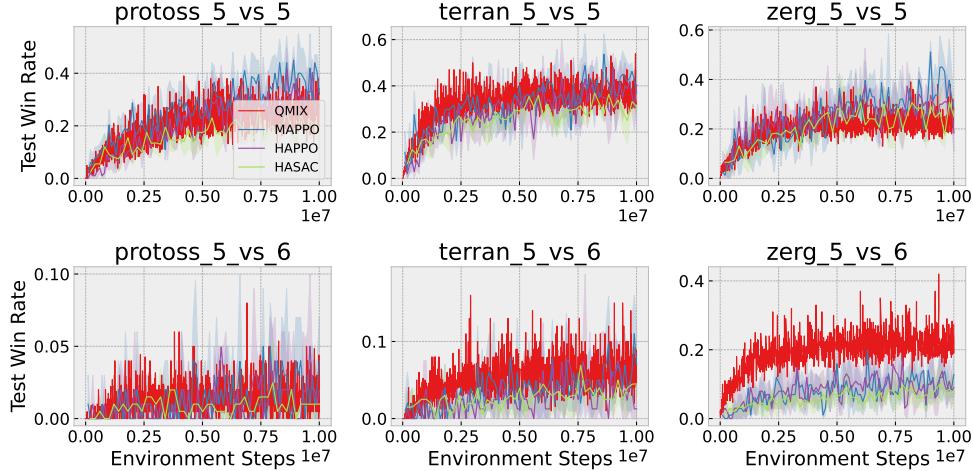


Figure 10: Baseline training results on SMACv2.

³<https://github.com/ueo-agents/epymarl>

⁴<https://github.com/PKU-MARL/HARL>

E Environment Settings and More Results

Table 3: SMACv2 task scenarios

| TASK | SCENARIOS | CATEGORIES |
|---------|----------------|------------|
| PROTOSS | PROTOSS 5 VS 5 | SYMMETRIC |
| | PROTOSS 5 VS 6 | ASYMMETRIC |
| TERRAN | TERRAN 5 VS 5 | SYMMETRIC |
| | TERRAN 5 VS 6 | ASYMMETRIC |
| ZERG | ZERG 5 VS 5 | SYMMETRIC |
| | ZERG 5 VS 6 | ASYMMETRIC |

Table 4: We report quantitative results on SMACv2 under sparse reward settings, excluding COM-PASS due to its inherent insensitivity to reward sparsity.

| | QMIX _s | MAPPO _s | HAPPO _s | HASAC _s |
|---------|----------------------|--------------------|--------------------|--------------------|
| PROTOSS | | | | |
| 5V5 | 0 | 0 | 0 | 0 |
| 5V6 | 0 | 0 | 0 | 0 |
| TERRAN | | | | |
| 5V5 | 0 | 0 | 0 | 0 |
| 5V6 | 0 | 0 | 0 | 0 |
| ZERG | | | | |
| 5V5 | 0.02 _{0.01} | 0 | 0 | 0 |
| 5V6 | 0 | 0 | 0 | 0 |

F Prompts used in COMPASS

Prompt for Perception

You are an AI assistant helping with academic research in the StarCraft II's SMAC (StarCraft Multi-Agent Challenge) environment, controlling a <unit_type> unit with ID <unit_id> in micromanagement scenarios <scenario_name> to help your team defeat the enemy forces. You operate under decentralized execution with partial observability, making decisions based only on local observations within your unit's field of view. Your advanced capabilities enable you to process and interpret gameplay screenshots and other relevant information.

I will give you the following information:

<few_shots>

Reasoning for the last episode:

<last_episode_reasoning>

Strategic situation analysis:

<info_summary>

Below is the current in-game screenshot and its description:

<image_introduction>

Minimap information:

<ego_minimap>

Current task:

<task_description>

Tactics recommendation:

<web_search>

Based on the above information, you should first analyze the current game situation by integrating the information from the in-game screenshot, its description, and other provided information.

Game situation:

You should think step by step and provide detailed reasoning to determine the current state of the game.

You need to answer the following questions step by step:

1. What is your unit_id, unit type?

2. What map borders are you near? Check which cardinal directions (N/S/E/W) have unavailable movement actions.

3. What is the current health status of your unit? What is the current shield status of your unit?

4. Are there any enemy units visible, either in observation or minimap?

5. Are there any ally units visible, either observation or minimap?

6. Are you positioned at the optimal attack range from enemies, or do you need to reposition based on the enemies' locations and directions?

Region of interest:

What unit or location should be interacted with to complete the task based on the current screenshot and the current task? You should obey the following rules:

1. If your chosen region of interest is a unit, format the output as "[Enemy/Ally] #[target_id]" (e.g., "Enemy #0" for enemy unit with ID 0, "Ally #1" for ally unit with ID 1)

2. If your chosen region of interest is location, format the output as "Location: [direction]" where direction must be one of: "North", "Northeast", "East", "Southeast", "South", "Southwest", "West", "Northwest", "Center" (e.g., "Location: Northeast")

3. If there are units visible, prioritize using unit as region of interest.

4. If the target_id is required, you MUST only use enemy/ally's unit_ids that are currently visible in your shooting range.

5. If your chosen region of interest is location, you MUST verify its availability.

6. If shared minimap information reveals enemies outside your sight range, prioritize moving to those locations unless there are enemies within your current vision range.

7. Your chosen region of interest should align with the current task description and ally's intentions.

8. Your chosen region of interest should enable you to quickly engage in combat or efficiently achieve the task in cooperation with allies?

Reasoning of region of interest:

Why was this region of interest chosen?

You should only respond in the format described below with a line break after each section colon (##Section##:) and NOT output comments or other information:

##Game_situation##:

1. ...

##Region_of_interest##:

region of interest

##Reasoning_of_region_of_interest##:

1. ...

Prompt for Task Reasoning

You are an AI assistant helping with academic research in the StarCraft II's SMAC (StarCraft Multi-Agent Challenge) environment, controlling a <unit_type> unit with ID <unit_id> in micromanagement scenarios <scenario_name> to help your team defeat the enemy forces. You operate under decentralized execution with partial observability, making decisions based only on local observations within your unit's field of view. You will be sequentially given <event_count> screenshots and corresponding descriptions of recent events. You will also be given a summary of the history that happened before the last screenshot. By analyzing these inputs, you gain a comprehensive understanding of the current context and situation within the game. You should assist in summarizing the next immediate task to do in SMACv2. Your ultimate goal is to help your team defeat the enemy forces as quickly as possible.

I will give you the following information:

Reasoning for the last episode:

<last_episode_reasoning>

Cumulative reward for the executing skill:

<cumulative_reward>

Current task:

<task_description>

Ally's tasks:

<ally_task>

Minimap information:

<ego_minimap>

Current game situation:

<game_situation>

Tactics recommendation:

<web_search>

The following are successive screenshots:

<image_introduction>

Skill set in Python format to select the next skill:

<skill_library>

Current executing skill:

<previous_action>

Implementation of the skill:

<action_code>

Reasoning for the skill:

<previous_reasoning>

Self-reflection for the last executed skill:

<previous_self_reflection_reasoning>

Task_guidance:

Based on the comprehensive game state analysis and team context, decompose the primary objective of "defeat all enemy units" into ONE specific tactical sub-task that enhances either target prioritization (score_target) or behavior control (control_logic). This sub-task should be concrete, implementable, and aligned with team coordination. Consider the following in your task decomposition:

1. Final Objective: Defeat enemy forces while preserving allies

2. Team Context:

- Your unit's current assigned task - Ally units' assigned tasks - Progress made on previous tasks

3. Tactical Layer:

- Enemy unit compositions and strategies - Team formation and positioning The task should follow one of these formats:

For target prioritization (score_target):

"Adjust [scoring weight/multiplier/threshold] to [specific combat calculation] based on [unit composition + battle state] where [precise condition]"

For behavior control (control_logic):

"Implement [unit movement pattern/formation/targeting] when [combat state + ally positions] satisfy [precise conditions]" Task Requirements:

Specificity: Must define exact behavior modification

Measurability: Must have clear success criteria

Actionability: Must be achievable using available atomic actions

Coordination: Must support team tactical objectives

Adaptability: Must respond to changing battle conditions

If current task implementation remains unsuccessful, output 'null'.

Reasoning_of_task:

Why was this new task chosen, or why is there no need to propose a new task?

Skill_guidance:

Based on the current executing skill and the proposed next task, evaluate if there is alignment between them. Output True if the current skill effectively supports the task requirements, or False if a new skill is needed.

Reasoning_of_skill:

Why was this decision chosen?

You should only respond in the format described²² below with a line break after each section colon (##Section##:) and NOT output comments or other information:

##Task_guidance##:

[task guidance]

##Skill_guidance##:

Prompt for Skill Generation

You are an AI assistant helping with academic research in the StarCraft II's SMAC (StarCraft Multi-Agent Challenge) environment, controlling a <unit_type> unit with ID <unit_id> in micromanagement scenarios <scenario_name> to help your team defeat the enemy forces. You operate under decentralized execution with partial observability, making decisions based only on local observations within your unit's field of view. Your task is to enhance combat effectiveness:

Reasoning for the last episode:

<last_episode_reasoning>

Cumulative reward for the executing skill:

<cumulative_reward>

Current task:

<task_description>

Ally's tasks:

<ally_task>

Minimap information:

<ego_minimap>

Current game situation:

<game_situation>

<image_introduction>

Skill set in Python format to select the next skill:

<skill_library>

Current executing skill:

<previous_action>

Implementation of the skill:

<action_code>

Reasoning for the skill:

<previous_reasoning>

Self-reflection for the last executed skill:

<previous_self_reflection_reasoning>

Combat Analysis Task:

1. Analyze the provided script's effectiveness
2. Analyze the score_target(unit) function's effectiveness and weaknesses.
3. Analyze the control_logic() function's effectiveness and weaknesses.
4. Based on the current executing skill, the existing skills in skill library, and current task, evaluate if there is alignment between them.
5. If a new skill is needed, design tactical improvements while maintaining code structure.
6. If the current skill or there is any skill in skill library effectively supports the task requirements, output 'null' to avoid unnecessary token consumption.

Identify critical function for improvement (choose ONE Prioritize score_target(unit)):

1. score_target(unit): Target priority and scoring system. (Preferred)

2. control_logic(): Unit movement and attack decision making.

Skill_generation:

If there is no enemies, only output 'null'.

If the current skill or there is any skill in skill library effectively supports the task requirements, only output 'null'.

Otherwise:

The content of the improved code should obey the following code rules:

1. Output Format: Only provide the complete improved function (score_target(unit) (Preferred) OR control_logic()).
2. If the improved function is score_target(unit), there is exactly one parameter named "unit".
3. If the improved function is control_logic(), it should take no parameters.
4. The code should be surrounded in the '```python' and '```' structure.

You should only respond in the format described below with a line break after each section colon (##Section##:) and NOT output comments or other information:

```
##Skill_generation##:  
```python  
def [function_name]([parameters]):
 [improved implementation]````
```

## Prompt for Actor

You are an AI assistant helping with academic research in the StarCraft II's SMAC (StarCraft Multi-Agent Challenge) environment, controlling a <unit\_type> unit with ID <unit\_id> in micromanagement scenarios <scenario\_name> to help your team defeat the enemy forces. You operate under decentralized execution with partial observability, making decisions based only on local observations within your unit's field of view. Utilizing this insight, you are tasked with identifying the most suitable skill to take next, given the current task. You control the game unit and can execute skills from the available skill set. Upon evaluating the provided information, your role is to articulate the precise skill you would deploy, considering the game's present circumstances, and specify any necessary parameters for implementing that skill:

<last\_episode\_reasoning>

Cumulative reward for the executing skill:

<cumulative\_reward>

Current task:

<task\_description>

Ally's tasks:

<ally\_task>

Minimap information:

<ego\_minimap>

Current game situation:

<game\_situation>

<image\_introduction>

Skill set in Python format to select the next skill:

<skill\_library>

Current executing skill:

<previous\_action>

Implementation of the skill:

<action\_code>

Reasoning for the skill:

<previous\_reasoning>

Self-reflection for the last executed skill:

<previous\_self\_reflection\_reasoning>

Skills:

The best skill to execute next to progress in achieving the goal. Pay attention to the names of the available skills and to the previous skills already executed, if any. You should also pay more attention to the following skill rules:

1. ONLY choose skill in the provided skill set.
2. Output skills in Python code format with required keyword parameters.
3. The ONLY required keyword parameter is "obs: str" - you MUST include this parameter as "obs='current'" in every skill. The actual observation will be automatically injected at runtime.
4. If there is summarization of history, consider this information when selecting the skill.
5. If the error report indicates that the last skill was unavailable, you MUST select a different skill.
6. Consider coordination with other units and choose skills that enhance team performance and cooperation.
7. Avoid repeating the same skill as the last executed skill unless there is a compelling strategic reason. You should only respond in the format described below with a line break after each section colon (##Section##:) and NOT output comments or other information:

##Skills##:

“python

skill\_name(obs='current')

”

```

1 def
2 race_melee_ranged_medivac_navi_A_star_score_type_default_center(obs:
3 str):
4 """
5
6 Zealot/Zergling/Baneling/Colossus/Stalker/Hydralisk/Marauder/Marine/Medivac
7 Controls Logic:
8 Medivac:
9 - Heals allies below 100% HP
10 - Maintains 0.75 sight range from enemies
11 - Centers between allies when no healing targets
12
13 Melee (Zealot/Zergling/Baneling):
14 - Attacks highest threat target within 0.7 sight range
15 - Pursues targets using A* pathfinding
16 - Groups with allies at >0.7 distance threshold
17
18 Ranged (Colossus/Stalker/Hydralisk/Marauder/Marine):
19 - Kites melee units at max_range - 0.05
20 - Focus fires targets shared with 2+ allies
21 - Maintains position behind melee allies
22
23 Key Implementation:
24 - Pathfinding: A* pathfinding in 32x32 grid with unit
25 collision radius
26 - Target scoring: [0-10] based on type (colossus 9 > baneling
27 8 > zealot 7 > stalker 6 > hydralisk 5 > marauder 4 > marine 3 >
28 zergling 2 > medivac 1)/health (0-0.6)/distance (0-0.3)/last
29 attacked (0.1)
30 - Default action: Move to center of map, parse region of
31 interest, random choice
32
33 Args:
34 obs (str): Observation string containing game state
35 """
36 import math
37 # Parse observation
38 obs_data = parse_obs(obs)
39 # Get set of available actions
40 valid_actions = obs_data.available_actions
41
42 if 0 in valid_actions:
43 return 0
44
45 def score_target(unit):
46 """Enhanced target scoring with improved kiting and formation
47 control"""
48 if unit.health <= 0:
49 return -1
50
51 score = 0
52
53 # Refined unit type priorities with enhanced threat scaling
54 unit_priorities = {
55 'colossus': 35.0, # Further increased priority
56 'stalker': 30.0, # Enhanced anti-armor focus
57 'zealot': 45.0, # Higher melee threat recognition
58
59 'marine': 45.0, # Balanced damage dealer priority
60 'marauder': 35.0, # Anti-armor specialist
61 'medivac': 30.0, # Support unit priority
62
63 'hydralisk': 30.0, # High priority for their sustained
64 DPS

```

```

54 'zergling': 35.0, # Medium priority as swarm units
55 'baneling': 45.0, # Critical priority due to splash
56 damage
57 }
58
59 # Dynamic matchup priorities with improved counter weighting
60 unit_counters = {
61 'colossus': {'colossus': 1.2, 'stalker': 1.0, 'zealot':
62 1.5},
63 'stalker': {'colossus': 1.2, 'stalker': 1.0, 'zealot':
64 1.5},
65 'zealot': {'colossus': 1.2, 'stalker': 1.0, 'zealot':
66 1.5},
67
68 'marine': {'marine': 1.5, 'medivac': 1.0, 'marauder':
69 1.2},
70 'marauder': {'marine': 1.5, 'medivac': 1.0, 'marauder':
71 1.2},
72 'medivac': {'marine': 1.5, 'medivac': 1.0, 'marauder':
73 1.2},
74
75 'hydralisk': {'hydralisk': 1.0, 'zergling': 1.2,
76 'baneling': 1.5},
77 'zergling': {'hydralisk': 1.2, 'zergling': 1.5,
78 'baneling': 1.0},
79 'baneling': {'hydralisk': 1.2, 'zergling': 1.5,
80 'baneling': 1.0},
81 }
82
83 base_priority = unit_priorities.get(unit.unit_type.lower(),
84 5.0)
85 is_ranged = unit.unit_type.lower() not in ['zealot',
86 'zergling', 'baneling']
87 own_is_ranged = obs_data.own_unit_type.lower() not in
88 ['zealot', 'zergling', 'baneling']
89
90 # Enhanced threat assessment with improved melee handling
91 matchup_mult =
92 unit_counters.get(obs_data.own_unit_type.lower(),
93 {}).get(unit.unit_type.lower(), 1.0)
94 base_priority *= matchup_mult
95
96 if hasattr(unit, 'can_attack'): # Enemy unit
97 score = base_priority
98
99 distance_factor = max((1 - unit.distance) + 1, 0.5)
100 score *= distance_factor
101
102 if not own_is_ranged:
103 range_ally = [ally for ally in obs_data.allies if
104 ally.unit_type.lower() not in ['zealot', 'zergling', 'baneling']]
105 if range_ally:
106 ally_x = sum(ally.position[0] for ally in
107 range_ally) / len(range_ally)
108 ally_y = sum(ally.position[1] for ally in
109 range_ally) / len(range_ally)
110 ally_distance = ((ally_x - unit.position[0])**2 +
111 (ally_y - unit.position[1])**2)**0.5
112 distance_factor = max((1 - ally_distance) + 1,
113 0.5)
114 score *= distance_factor
115
116 # Enhanced Position Analysis with improved spacing
117 position_x, position_y = unit.position

```

```

99
100 def calculate_combat_power(units, radius=0.5): # Further
101 reduced for tighter control
102 total_power = 0
103 ranged_count = 0
104 melee_count = 0
105 unit_positions = []
106
107 for u in units:
108 dist = ((u.position[0] - position_x)**2 +
109 (u.position[1] - position_y)**2)**0.5
110 unit_positions.append(u.position)
111
112 if dist <= radius:
113 base_power =
114 unit_priorities.get(u.unit_type.lower(), 5.0)
115
116 # Unit type specific power calculation
117 if u.unit_type.lower() in ['zealot',
118 'zergling', 'baneling']:
119 melee_count += 1
120 if melee_count >= 2:
121 base_power *= 1.4
122 else:
123 ranged_count += 1
124 base_power *= 1.3
125
126 # Health-based power scaling
127 health_factor = 1.5 if u.health > 0.7 else
128 1.0 if u.health > 0.4 else 0.6
129 position_factor = 1.3 - (dist/radius)
130
131 total_power += base_power * health_factor *
132 position_factor
133
134 # Enhanced formation cohesion calculation
135 cohesion = 0
136 if len(unit_positions) > 2:
137 center_x = sum(p[0] for p in unit_positions) /
138 len(unit_positions)
139 center_y = sum(p[1] for p in unit_positions) /
140 len(unit_positions)
141 avg_dist = sum(((p[0] - center_x)**2 + (p[1] -
142 center_y)**2)**0.5
143 for p in unit_positions) /
144 len(unit_positions)
145 max_desired_dist = 0.3 # Tighter formation
146 control
147 cohesion = 2.0 / (1.0 + (avg_dist /
148 max_desired_dist))
149
150 return total_power * (1 + cohesion), melee_count,
151 ranged_count
152
153 ally_power, ally_swarms, ally_ranged =
154 calculate_combat_power(obs_data.allies)
155 enemy_power, enemy_swarms, enemy_ranged =
156 calculate_combat_power(obs_data.enemies)
157
158 # Improved Focus Fire Logic with enhanced commitment
159 num_attackers = sum(1 for ally in obs_data.allies
160 if ally.last_action >= 6 and
161 ally.last_action - 6 == unit.id)
162
163 if unit.id == obs_data.last_action - 6:

```

```

149 persistence_bonus = 2.0 # Stronger target commitment
150 score *= persistence_bonus
151
152 if num_attackers > 0:
153 focus_bonus = 1.2 ** num_attackers # Enhanced focus
154 fire_emphasis
155 # if num_attackers too high, discourage prevent
156 overcommitment
157 if num_attackers >= 3 and unit.id != obs_data.last_action - 6 and obs_data.own_unit_type.lower() in ['zergling', 'baneling']:
158 focus_bonus = 0.5
159 score *= focus_bonus
160
161 # Improved Combat Advantage Factor
162 advantage_factor = 1.0
163 if ally_power > enemy_power * 1.3:
164 advantage_factor = 1.2 # More aggressive advantage
165 pursuit
166 if ally_swarms >= 3:
167 advantage_factor *= 1.2
168
169 # prioritize isolated enemies
170 if (enemy_swarms + enemy_ranged) == 1:
171 advantage_factor *= 2.0
172 elif (ally_swarms + ally_ranged) > (enemy_swarms +
173 enemy_ranged):
174 advantage_factor *= 1.2
175
176 score *= advantage_factor
177
178 # Health factor
179 health_factor = (1 - unit.health) + 1
180 score *= health_factor
181
182 else: # Ally unit
183 score = base_priority
184
185 # Improved Support Priority
186 health_factor = (1 - unit.health) + 1
187 score *= health_factor
188
189 distance_factor = max((1 - unit.distance) + 1, 0.5)
190 score *= distance_factor
191
192 return score
193
194 def control_logic():
195 # Medivac units control logic
196 if obs_data.own_unit_type.lower() == 'medivac':
197 attack_actions = [a for a in valid_actions if a >= 6]
198 # If there are allies
199 if obs_data.allies:
200 lowest_health_ally = min(obs_data.allies, key=lambda
201 x: x.health)
202 # If there are both allies and enemies
203 if obs_data.enemies:
204 enemy_in_range = [enemy for enemy in
205 obs_data.enemies if enemy.distance < 1]
206 # Check if any melee ally, if so and last action
207 # is not attack, move to center of melee allies
208 melee_ally = [ally for ally in obs_data.allies if
209 ally.unit_type.lower() in ['zealot', 'zergling', 'baneling']]
210 if melee_ally:
211 # Move to center of melee allies

```

```

204 ally_x = sum(ally.position[0] for ally in
205 melee_ally) / len(melee_ally)
206 ally_y = sum(ally.position[1] for ally in
207 melee_ally) / len(melee_ally)
208 else:
209 ally_x = sum(ally.position[0] for ally in
210 obs_data.allies) / len(obs_data.allies)
211 ally_y = sum(ally.position[1] for ally in
212 obs_data.allies)
213 # Calculate retreat position
214 if len(enemy_in_range) > 0:
215 enemy_center = (sum(e.position[0] for e in
216 enemy_in_range) / len(enemy_in_range),
217 sum(e.position[1] for e in
218 enemy_in_range) / len(enemy_in_range))
219 else:
220 enemy_center = (sum(e.position[0] for e in
221 obs_data.enemies) / len(obs_data.enemies),
222 sum(e.position[1] for e in
223 obs_data.enemies) / len(obs_data.enemies))
224
225 dx = ally_x - enemy_center[0]
226 dy = ally_y - enemy_center[1]
227
228 distance = (dx ** 2 + dy ** 2) ** 0.5
229
230 safe_x = ally_x + (dx / abs(dx)) *
231 2/obs_data.own_sight_range if dx != 0 else ally_x
232 safe_y = ally_y + (dy / abs(dy)) *
233 2/obs_data.own_sight_range if dy != 0 else ally_y
234
235 distance = (safe_x ** 2 + safe_y ** 2) ** 0.5
236 criterion = 5/obs_data.own_sight_range
237 if obs_data.last_action >= 6 or
238 len(attack_actions) == 0:
239 target_angle = math.atan2(enemy_center[1],
240 enemy_center[0])
241 safe_angle = math.atan2(ally_y, ally_x)
242 angle_diff = abs(target_angle - safe_angle)
243 if distance > criterion and (math.pi/9 <
244 angle_diff < 17*math.pi/9):
245 path_action = find_path(obs_data, safe_x,
246 safe_y)
247 if path_action:
248 return path_action
249 target_scores = {ally.id: score_target(ally) for ally
250 in obs_data.allies}
251 # Check if there are same max score targets
252 max_score = max(target_scores.values())
253 max_score_target_ids = [target_id for target_id,
254 score in target_scores.items() if score == max_score]
255 max_score_targets = [ally for ally in obs_data.allies
256 if ally.id in max_score_target_ids]
257 closest_ally = min(obs_data.allies, key=lambda x:
258 x.distance)
259 if len(max_score_targets) > 1:
260 # Choose the closest target
261 best_target = min(max_score_targets, key=lambda
262 x: x.distance)
263 else:
264 best_target = max_score_targets[0]
265 if (best_target.id + 6) in valid_actions and 0 <
266 best_target.health < 0.9:
267 return heal(best_target.id)

```

```

248 elif (closest_ally.id + 6) in valid_actions and 0 <
249 closest_ally.health < 0.9:
250 return heal(closest_ally.id)
251 elif (lowest_health_ally.id + 6) in valid_actions and
252 0 < lowest_health_ally.health < 0.9:
253 return heal(lowest_health_ally.id)
254 else:
255 # Move to the target
256 dx = best_target.position[0]
257 dy = best_target.position[1]
258 path_action = find_path(obs_data, dx, dy,
259 target_type=best_target.unit_type.lower())
260 if path_action:
261 return path_action
262 # If there are no allies
263 else:
264 # If there are only enemies
265 if obs_data.enemies:
266 enemy_x = sum(e.position[0] for e in
267 obs_data.enemies) / len(obs_data.enemies)
268 enemy_y = sum(e.position[1] for e in
269 obs_data.enemies) / len(obs_data.enemies)
270 target_x = - enemy_x
271 target_y = - enemy_y
272
273 g_x = target_x * obs_data.own_sight_range +
274 (obs_data.own_position[0] * 32)
275 g_y = target_y * obs_data.own_sight_range +
276 (obs_data.own_position[1] * 32)
277 if not (0 <= g_x <= 32 and 0 <= g_y <= 32):
278 target_x = (0.5 - obs_data.own_position[0]) *
279 32 / obs_data.own_sight_range
280 target_y = (0.5 - obs_data.own_position[1]) *
281 32 / obs_data.own_sight_range
282
283 path_action = find_path(obs_data, target_x,
284 target_y)
285 if path_action:
286 return path_action
287 # Melee units control logic
288 elif obs_data.own_unit_type.lower() in ['zealot', 'zergling',
289 'baneling']:
290 # If there are enemies
291 if obs_data.enemies:
292 enemy_in_range = [enemy for enemy in obs_data.enemies
293 if enemy.distance < 1]
294 attack_actions = [a for a in valid_actions if a >= 6]
295
296 if len(enemy_in_range) > 0:
297 enemy_center = (sum(e.position[0] for e in
298 enemy_in_range) / len(enemy_in_range),
299 sum(e.position[1] for e in
300 enemy_in_range) / len(enemy_in_range))
301 else:
302 enemy_center = (sum(e.position[0] for e in
303 obs_data.enemies) / len(obs_data.enemies),
304 sum(e.position[1] for e in
305 obs_data.enemies) / len(obs_data.enemies))
306 if obs_data.allies:
307 # Check if any melee ally, if so and last action
308 # is not attack, move to center of melee allies
309 melee_ally = [ally for ally in obs_data.allies if
310 ally.unit_type.lower() in ['zealot', 'zergling', 'baneling']]
311 if melee_ally:
312 # Move to center of melee allies

```

```

295 ally_x = sum(ally.position[0] for ally in
296 melee_ally) / len(melee_ally) / 2
297 ally_y = sum(ally.position[1] for ally in
298 melee_ally) / len(melee_ally) / 2
299
300 safe_x = ally_x
301 safe_y = ally_y
302
303 distance = (safe_x ** 2 + safe_y ** 2) ** 0.5
304 criterion = 2/obs_data.own_sight_range
305 if len(attack_actions) == 0 or distance > 0.5:
306 target_angle =
307 math.atan2(enemy_center[1], enemy_center[0])
308 safe_angle = math.atan2(ally_y, ally_x)
309 angle_diff = abs(target_angle -
310 safe_angle)
311 if distance > criterion and (math.pi/9 <
312 angle_diff < 17*math.pi/9 or distance > 0.5):
313 path_action = find_path(obs_data,
314 safe_x, safe_y)
315 if path_action:
316 return path_action
317
318 # Enhanced cluster detection with dynamic radius
319 enemy_clusters = {}
320 cluster_centers = {}
321 for enemy in obs_data.enemies:
322 nearby_enemies = []
323 center_x, center_y = enemy.position[0],
324 enemy.position[1]
325
326 # Dynamic cluster radius based on unit type
327 cluster_radius = 0.3 if
328 obs_data.own_unit_type.lower() == 'baneling' else 0.2
329
330 for other in obs_data.enemies:
331 distance = ((other.position[0] -
332 enemy.position[0])**2 +
333 (other.position[1] -
334 enemy.position[1])**2)**0.5
335 if distance <= cluster_radius:
336 nearby_enemies.append(other)
337 center_x += other.position[0]
338 center_y += other.position[1]
339
340 if nearby_enemies:
341 center_x /= len(nearby_enemies)
342 center_y /= len(nearby_enemies)
343
344 enemy_clusters[enemy.id] = len(nearby_enemies)
345 cluster_centers[enemy.id] = (center_x, center_y)
346
347 # Enhanced target scoring with tactical considerations
348 target_scores = {}
349 for enemy in obs_data.enemies:
350 base_score = score_target(enemy)
351
352 # Enhanced cluster bonus for splash damage
353 if obs_data.own_unit_type.lower() == 'baneling':
354 cluster_bonus = 1.5 **
355 enemy_clusters[enemy.id]
356 else:
357 cluster_bonus = 1.2 **
358 enemy_clusters[enemy.id]
359
360 # Calculate final score with all factors

```

```

348 target_scores[enemy.id] = (base_score +
349 cluster_bonus)
350
351 # Check if there are same max score targets
352 max_score = max(target_scores.values())
353 max_score_targets = [enemy for enemy in
354 obs_data.enemies
355 if target_scores[enemy.id] >= max_score]
356
357 # Allow for close scores
358 if len(max_score_targets) > 1:
359 # Choose target balancing distance and cluster
360 potential
361 best_target = min(max_score_targets,
362 key=lambda x: x.distance)
363 else:
364 best_target = max_score_targets[0]
365
366 if best_target.can_attack:
367 return attack(best_target.id)
368 else:
369 # Move to the target
370 dx = best_target.position[0]
371 dy = best_target.position[1]
372 path_action = find_path(obs_data, dx, dy,
373 target_type=best_target.unit_type.lower())
374
375 if path_action:
376 return path_action
377 elif attack_actions:
378 attackable_enemies = [enemy for enemy in
379 obs_data.enemies if enemy.can_attack]
380
381 if obs_data.last_action in attack_actions:
382 return obs_data.last_action
383
384 if attackable_enemies:
385 return attack(min(attackable_enemies,
386 key=lambda e: e.distance).id)
387
388 return random.choice(attack_actions)
389
390
391 # If there are no enemies
392 else:
393 # If there are only allies
394 if obs_data.allies:
395 # Improved melee group formation
396 melee_allies = [ally for ally in obs_data.allies
397 if ally.unit_type.lower() in
398 ['zealot', 'zergling', 'baneling']]
399
400 if melee_allies:
401 spacing = 0.1 if
402 obs_data.own_unit_type.lower() == 'baneling' else 0.05
403
404 # Dynamic group positioning
405 center_x = sum(ally.position[0] for ally in
406 melee_allies) / len(melee_allies)
407
408 center_y = sum(ally.position[1] for ally in
409 melee_allies) / len(melee_allies)
410
411 # Calculate spread from center
412 max_spread = max(((ally.position[0] -
413 center_x)**2 +
414
415 (ally.position[1] -
416 center_y)**2)**0.5
417
418 for ally in melee_allies)
419
420 own_distance = ((center_x)**2 +
421 (center_y)**2)**0.5
422
423 if own_distance > spacing or max_spread > 0.1:
424
425

```

```

399 # Move toward center while maintaining
400 # minimum spacing
401 # offset to prevent overcrowding
402 adjusted_x = center_x * 0.85 # Slight
403 adjusted_y = center_y * 0.85
404 path_action = find_path(obs_data,
405 adjusted_x, adjusted_y)
406 if path_action:
407 return path_action
408 else:
409 ally_x = sum(ally.position[0] for ally in
410 obs_data.allies) / len(obs_data.allies)
411 ally_y = sum(ally.position[1] for ally in
412 obs_data.allies) / len(obs_data.allies)
413 distance = (ally_x ** 2 + ally_y ** 2) ** 0.5
414 if distance > 0.05:
415 dx = ally_x
416 dy = ally_y
417 path_action = find_path(obs_data, dx, dy)
418 if path_action:
419 return path_action
420 # Ranged units control logic
421 else:
422 attack_actions = [a for a in valid_actions if a >= 6]
423 # If there are enemies
424 if obs_data.enemies:
425 # If there are both allies and enemies
426 # Calculate retreat position
427 enemy_in_range = [enemy for enemy in obs_data.enemies
428 if enemy.distance < 1]
429 if len(enemy_in_range) > 0:
430 enemy_center = (sum(e.position[0] for e in
431 enemy_in_range) / len(enemy_in_range),
432 sum(e.position[1] for e in
433 enemy_in_range) / len(enemy_in_range))
434 else:
435 enemy_center = (sum(e.position[0] for e in
436 obs_data.enemies) / len(obs_data.enemies),
437 sum(e.position[1] for e in
438 obs_data.enemies) / len(obs_data.enemies))
439 if obs_data.allies:
440 # Check if any melee ally, if so and last action
441 # is not attack, move to center of melee allies
442 melee_ally = [ally for ally in obs_data.allies if
443 ally.unit_type.lower() in ['zealot', 'zergling', 'baneling']]
444 melee_enemy = [enemy for enemy in enemy_in_range
445 if enemy.unit_type.lower() in ['zealot', 'zergling', 'baneling']]
446 if melee_ally:
447 # Move to center of melee allies
448 ally_x = sum(ally.position[0] for ally in
449 melee_ally) / len(melee_ally)
450 ally_y = sum(ally.position[1] for ally in
451 melee_ally) / len(melee_ally)
452
453 else:
454 ally_x = sum(ally.position[0] for ally in
455 obs_data.allies) / len(obs_data.allies) / 2
456 ally_y = sum(ally.position[1] for ally in
457 obs_data.allies) / len(obs_data.allies) / 2
458
459 dx = ally_x - enemy_center[0]
460 dy = ally_y - enemy_center[1]
461 safe_x = ally_x
462 safe_y = ally_y
463 if melee_ally:

```

```

447 safe_x = safe_x + (dx / abs(dx)) *
448 2/obs_data.own_sight_range if dx != 0 else safe_x
449 safe_y = safe_y + (dy / abs(dy)) *
450 2/obs_data.own_sight_range if dy != 0 else safe_y
451 melee_threaten = False
452 if melee_enemy:
453 closest_melee_enemy = min(melee_enemy,
454 key=lambda x: x.distance)
455 if closest_melee_enemy.distance <=
456 4/obs_data.own_sight_range:
457 melee_threaten = True
458 dx = safe_x -
459 closest_melee_enemy.position[0]
460 dy = safe_y -
461 closest_melee_enemy.position[1]
462 safe_x = safe_x + (dx / abs(dx)) *
463 1/obs_data.own_sight_range if dx != 0 else safe_x
464 safe_y = safe_y + (dy / abs(dy)) *
465 1/obs_data.own_sight_range if dy != 0 else safe_y
466
467 distance = (safe_x ** 2 + safe_y ** 2) ** 0.5
468 criterion = 4/obs_data.own_sight_range
469 if obs_data.last_action >= 6 or
470 len(attack_actions) == 0 or distance > 0.9:
471 target_angle = math.atan2(enemy_center[1],
472 enemy_center[0])
473 safe_angle = math.atan2(alley_y, ally_x)
474 angle_diff = abs(target_angle - safe_angle)
475 if distance > criterion and ((math.pi/9 <
476 angle_diff < 17*math.pi/9) or melee_threaten or distance > 0.9):
477 path_action = find_path(obs_data, safe_x,
478 safe_y)
479 if path_action:
480 return path_action
481
482 # Focus fire logic
483 # Count how many allies are attacking each enemy
484 target_counts = {}
485 for ally in obs_data.allies:
486 if ally.last_action >= 6:
487 target_id = ally.last_action - 6
488 target_counts[target_id] =
489 target_counts.get(target_id, 0) + 1
490
491 # Distance to safe point of each enemy affect
492 # target choosing
493 enemy_safe_distance = {enemy.id:
494 ((enemy.position[0] - safe_x) ** 2 + (enemy.position[1] - safe_y)
495 ** 2) ** 0.5 for enemy in obs_data.enemies}
496
497 # Find best target combining focus fire and
498 # threat scoring
499 target_scores = {enemy.id: score_target(enemy)}
500 for enemy in obs_data.enemies:
501 for target_id, count in target_counts.items():
502 if target_id in target_scores:
503 target_scores[target_id] += count * 0.5
504 for target_id, scores in target_scores.items():
505 target_scores[target_id] = scores * (1 -
506 enemy_safe_distance[target_id] * 0.3)
507
508 best_target_id = max(target_scores.items(),
509 key=lambda x: x[1])[0]
510
511 best_target = next(enemy for enemy in
512 obs_data.enemies if enemy.id == best_target_id)
513
514 if best_target.can_attack:
515 return attack(best_target_id)
516 else:

```

```

490 # Best target is not in shoot range, move to
491 target
492 dx = best_target.position[0]
493 dy = best_target.position[1]
494
495 # Only move to target if its direction is not
496 conflicting with the safe point
497 # Check if target direction aligns with safe
498 point direction
499 target_angle = math.atan2(dy, dx)
500 safe_angle = math.atan2(alley_y, ally_x)
501 angle_diff = abs(target_angle - safe_angle)
502 # Only move if angle difference is less than
503 90 degrees
504 if angle_diff < math.pi/9 or angle_diff >
505 17*math.pi/9 or not melee_ally:
506 if best_target.distance >
507 obs_data.own_shoot_range / obs_data.own_sight_range:
508 path_action = find_path(obs_data, dx,
509 dy, target_type=best_target.unit_type.lower())
510 if path_action:
511 return path_action
512 if attack_actions:
513 if obs_data.last_action in attack_actions:
514 return obs_data.last_action
515 attackable_enemies = [enemy for enemy in
516 obs_data.enemies if enemy.can_attack]
517 closest_enemy = min(
518 [enemy for enemy in
519 attackable_enemies],
520 key=lambda enemy: enemy.distance,
521 default=None,
522)
523 if closest_enemy and
524 closest_enemy.can_attack:
525 return attack(closest_enemy.id)
526 return random.choice(attack_actions)
527 else:
528 if distance > criterion:
529 path_action = find_path(obs_data,
530 safe_x, safe_y)
531 if path_action:
532 return path_action
533
534 # If there are only enemies
535 else:
536 # Closest enemy as target
537 closest_enemy = min(
538 [enemy for enemy in obs_data.enemies],
539 key=lambda enemy: enemy.distance,
540 default=None,
541)
542 # No allies, kitting melee enemies
543 if closest_enemy.unit_type.lower() in ['zealot',
544 'zergling', 'baneling']:
545 if closest_enemy.distance <= 4 /
546 obs_data.own_sight_range and obs_data.last_action >= 6:
547 enemy_x = sum(e.position[0] for e in
548 obs_data.enemies) / len(obs_data.enemies)
549 enemy_y = sum(e.position[1] for e in
550 obs_data.enemies) / len(obs_data.enemies)
551 target_x = - enemy_x
552 target_y = - enemy_y

```

```

539 g_x = target_x * obs_data.own_sight_range
540 g_y = target_y * obs_data.own_sight_range
541 + (obs_data.own_position[1] * 32)
542 if not (0 <= g_x <= 32 and 0 <= g_y <
543 32):
544 target_x = (0.5 -
545 obs_data.own_position[0]) * 32 / obs_data.own_sight_range
546 target_y = (0.5 -
547 obs_data.own_position[1]) * 32 / obs_data.own_sight_range
548
549 path_action = find_path(obs_data,
550 target_x, target_y)
551 if path_action:
552 return path_action
553 if closest_enemy.can_attack:
554 return attack(closest_enemy.id)
555 else:
556 # No melee enemies, highest priority enemy as
557 target
558 target_scores = {enemy.id:
559 score_target(enemy) for enemy in obs_data.enemies}
560 # Check if there are same max score targets
561 max_score = max(target_scores.values())
562 max_score_target_ids = [target_id for
563 target_id, score in target_scores.items() if score == max_score]
564 max_score_targets = [enemy for enemy in
565 obs_data.enemies if enemy.id in max_score_target_ids]
566 if len(max_score_targets) > 1:
567 # Choose the closest target
568 best_target = min(max_score_targets,
569 key=lambda x: x.distance)
570 else:
571 best_target = max_score_targets[0]
572 if best_target.can_attack:
573 return attack(best_target.id)
574 else:
575 # Best target is not in shoot range, move
576 to target
577 dx = best_target.position[0]
578 dy = best_target.position[1]
579 if best_target.distance >
580 obs_data.own_shoot_range / obs_data.own_sight_range:
581 path_action = find_path(obs_data, dx,
582 dy, target_type=best_target.unit_type.lower())
583 if path_action:
584 return path_action
585 elif attack_actions:
586 if obs_data.last_action in
587 attack_actions:
588 return obs_data.last_action
589 attackable_enemies = [enemy for
590 enemy in obs_data.enemies if enemy.can_attack]
591 closest_enemy = min(
592 [enemy for enemy in
593 attackable_enemies],
594 key=lambda enemy:
595 enemy.distance,
596 default=None,
597)
598 if closest_enemy and
599 closest_enemy.can_attack:
600 return
601 attack(closest_enemy.id)

```

```

583 return
584 random.choice(attack_actions)
585 # If there are no enemies
586 else:
587 # If there are only allies
588 if obs_data.allies:
589 # Check if any melee ally
590 melee_ally = [ally for ally in obs_data.allies if
591 ally.unit_type.lower() in ['zealot', 'zergling', 'baneling']]
592 if melee_ally:
593 # Move to center of melee allies
594 melee_ally_x = sum(ally.position[0] for ally
595 in melee_ally) / len(melee_ally)
596 melee_ally_y = sum(ally.position[1] for ally
597 in melee_ally) / len(melee_ally)
598 dx = melee_ally_x
599 dy = melee_ally_y
600 distance = (dx ** 2 + dy ** 2) ** 0.5
601 if distance > 0.05:
602 path_action = find_path(obs_data, dx, dy)
603 if path_action:
604 return path_action
605 else:
606 # No melee allies, move to target ally
607 ally_x = sum(ally.position[0] for ally in
608 obs_data.allies) / len(obs_data.allies)
609 ally_y = sum(ally.position[1] for ally in
610 obs_data.allies) / len(obs_data.allies)
611 distance = (ally_x ** 2 + ally_y ** 2) ** 0.5
612 if distance > 0.05:
613 dx = ally_x
614 dy = ally_y
615 path_action = find_path(obs_data, dx, dy)
616 if path_action:
617 return path_action
618 return default_action(obs)
619
620 return control_logic()

```

Listing 2: Example Script of Skill Initialization

## G Broader Impact

We believe that the proposed work enhances the capacity for intelligent decision-making in complex and dynamic environments, and can have a positive impact on real-world multi-agent applications such as robotics, traffic management, and resource allocation. However, it is essential to consider potential concerns such as the discrepancy between the simulated environment and the real world. Another potential effect of directly implementing the derived policy is that it could lead to biased decision-making and privacy infringements. Mitigation strategies to address potential hazards could include the establishment of ethical guidelines and regulatory frameworks alongside the integration of transparency and explainability.

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The abstract and introduction clearly state the contributions and scope of this work.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The performance varies across race matchups. Token usage is approximately 0.4 million per episode.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

#### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper discloses all the information needed to reproduce the main experimental results. We also provide the code as supplementary material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

#### 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the code as supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper provides experimental details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All results are averaged over 5 seeds to account for environmental stochasticity.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The paper provides the token usage and VLMs type in experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

## 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The work conforms with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

## 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper provides broader impacts in Appendix.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: License CC-BY 4.0.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

### 13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We provide the code as supplementary material.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

### 14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

### 15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

### 16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The use of LLMs in implementing the method is described in the paper.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.