

---

# LLM-PySC2: STARCRAFT II LEARNING ENVIRONMENT FOR LARGE LANGUAGE MODELS <sup>\*†</sup>

---

**Zongyuan Li<sup>1</sup>, Yanan Ni<sup>2</sup>, Runnan Qi<sup>2</sup>, Lumin Jiang<sup>2</sup>, Chang Lu<sup>1</sup>, Xiaojie Xu<sup>1</sup>,  
Xiangbei Liu<sup>1</sup>, Pengfei Li<sup>1</sup>, Yunzheng Guo<sup>1</sup>, Zhe Ma<sup>1</sup>,  
Xian Guo<sup>1,\*</sup>, Kuihua Huang<sup>2,\*</sup>, Xuebo Zhang<sup>1,\*</sup>**

<sup>1</sup> College of Artificial Intelligence, Nankai University

<sup>2</sup> Laboratory for Big Data and Decision, National University of Defense Technology

## ABSTRACT

This paper introduces a new environment LLM-PySC2 (the Large Language Model StarCraft II Learning Environment), a platform derived from DeepMind’s StarCraft II Learning Environment that serves to develop Large Language Models (LLMs) based decision-making methodologies. This environment is the first to offer the complete StarCraft II action space, multi-modal observation interfaces, and a structured game knowledge database, which are seamlessly connected with various LLMs to facilitate the research of LLMs-based decision-making. To further support multi-agent research, we developed an LLM collaborative framework that supports multi-agent concurrent queries and multi-agent communication. In our experiments, the LLM-PySC2 environment is adapted to be compatible with the StarCraft Multi-Agent Challenge (SMAC) task group and provided eight new scenarios focused on macro-decision abilities. We evaluated nine mainstream LLMs in the experiments, and results show that sufficient parameters are necessary for LLMs to make decisions, but improving reasoning ability does not directly lead to better decision-making outcomes. Our findings further indicate the importance of enabling large models to learn autonomously in the deployment environment through parameter training or train-free learning techniques. Ultimately, we expect that the LLM-PySC2 environment can promote research on learning methods for LLMs, helping LLM-based methods better adapt to task scenarios.

## 1 Introduction

In 2017, the StarCraft II Learning Environment (SC2LE)[1] was developed by DeepMind and Blizzard Entertainment. It is the first environment that enables various reinforcement learning (RL) agents to compete with each other in the StarCraft II game, and promoted the emergence of decision-making methods such as QMix[2], Weighted QMIX[3], MAPPO[4], and the household name AlphaStar[5]. However, RL-trained agents typically require a substantial amount of data and prolonged interactions, but still lack generalization capabilities in most scenarios due to the task-relevant reward function. Consequently, it is urgent to develop new decision-making methods at the present.

Additionally, impressive research efforts such as Stanford Town[6], LLM plays MineCraft[7] and the game of Diplomacy[8] have demonstrated great potential in LLM-based decision-making in recent years. Considering that large models exhibit greater interactivity, interpretability, and reasoning capabilities, it is quite natural to apply large models in complex decision-making environments. However, there is no sufficiently comprehensive platform to support research on LLM decision-making methods in complex environments. Notably, the mainstream platform SC2LE environment does not yet support research on decision-making with large models.

In order to leverage the advantages of large models and circumvent the disadvantages of RL, researchers developed the SC2 module into *TextStarCraft II* (TSC2)[9], enabling LLMs to interact with the StarCraft II environment for the first time. However, there are some restrictions in the environment. The LLM based agent can not use micro-operations and

---

<sup>\*</sup> Corresponding author.

<sup>†</sup> Code is available at <https://github.com/NKAI-Decision-Team/LLM-PySC2>

unit skills to defeat enemy units due to the scale-cropped discrete action space. While observation only contains unit counts and upgrade status that are not sufficient for the implementation of complex strategies. What's more, multi-agent collaboration is not available because TSC2 is a single-agent framework.

To address these issues, we developed *LLM-PySC2*, an environment derived from SC2LE, based on PySC2 module. This environment provides agents with comprehensive observations, including global information and agent-specific local combat information (in text form or multimodal form) and a structured game knowledge database. We also expanded the action space to the full StarCraft II action space, enabling agents to perform fine-grained operations and unit skills. To support multi-agent research, we built a multi-agent framework with a communication system that allows both point-to-point and domain communication.

In experiments, eight new scenarios were proposed. Unlike the SMAC[10] tasks, these tasks emphasize not only micro-operations but also task understanding and macro-decision abilities. We tested nine mainstream LLMs in both the SMAC tasks and the new proposed scenarios. The results indicate that pre-trained LLMs *have* possess decision-making ability but *lack the ability* to make consistently effective decisions. Pre-trained LLMs without task-specific training may be unable to analyze the key elements for achieving victory. They often fail to identify the important part of the game knowledge for the most times, making mistakes in analysis or even dealing damage to allies sometimes.

In summary, there remains much to be done to raise the ability of LLMs in the domain of multi-agent decision making. We hope that the LLM-PySC2 environment will advance research on LLM learning techniques, helping LLM-based methods better adapt to task scenarios.

## 2 Related Works

### 2.1 Starcraft II

Starcraft II is a classic platform for evaluating algorithms. Specially, as a real-time strategy game, StarCraft II features a high-dimensional partially observable state space and a huge continuous action space. With three species and more than 120 types of units, it is widely regarded as one of the most complex and challenging environments and is commonly used for evaluating advanced decision-making methods.

To support the research of learning methods, DeepMind and Blizzard Entertainment developed SC2LE, a comprehensive environment for RL research. This environment is designed to improve research in learning algorithms within complex strategy games. It provides RL interfaces such as the observation, actions, and reward function, considered as one of the most significant environments in the field of artificial intelligence.

Consequently, after the introduction of SC2LE, more and more StarCraft II environments have emerged. Among those environments, SMAC[10] and PyMARL are the most famous. SMAC is a benchmark comprising 23 tasks specifically designed for multi-agent RL, mostly focusing on distributed multi-agent decision-making. To evaluate MARL algorithms, the SMAC team also developed PyMARL as their training platform. In the PyMARL framework, over five algorithms are integrated, and the framework is gradually expanded into a multi-environment available RL platform.

Overall, their work effectively advanced the research on multi-agent learning methods, made significant contributions to the field of intelligent decision-making, and motivated us to develop an environment for LLM-based methods.

### 2.2 LLM Decision-Making and Text StarCraft II

In recent years, the decision-making ability of LLMs has started to attract attention. In 2023, an LLM-based agent called the Ghost in Minecraft achieved 67.5% success in Minecraft's diamond challenge. After that, Agent-Pro[11], an LLM agent capable of using strategies like bluffing in Poker, was developed. Additionally, researchers deployed LLM agents in Werewolf[12], a game with deception and counter-deception through communication, and developed LLM agents in the game of Diplomacy, a game of collaboration and competition.

These works inspire researchers to develop LLM-based decision-making methods in games. As one of the most famous real-time strategy games, StarCraft II was first developed into an LLM-interactable environment called TSC2. This environment enables LLMs to make macro-decisions in StarCraft II and proves that LLMs can make decisions and defeat build-in bots at level-4 in StarCraft-II. However, TSC2 does not support micro-operations on units and multi-agent collaboration and faces limitations in observation and action space.

Under these circumstances, we constructed the LLM-PySC2 environment, aiming to solve these problems and provide a new StarCraft-II environment. We also make our environment compatible with SMAC tasks, facilitating comparisons with algorithms developed in the StarCraft environment.

### 3 LLM-PySC2 environment

#### 3.1 Framework

The LLM-PySC2 environment is built on the PySC2 module's agent level. In Figure 1, the MainAgent plays the role of controlling the camera, selecting units, collecting observations, and executing actions, while the LLM agent plays the role of the actual decision maker that observes game situations, analyzes, and gives actions. Each LLM agent connects to an LLM, getting a text or multimodal observations from a wrapper, querying the LLM in an independent thread, and finally getting game analysis and actions.

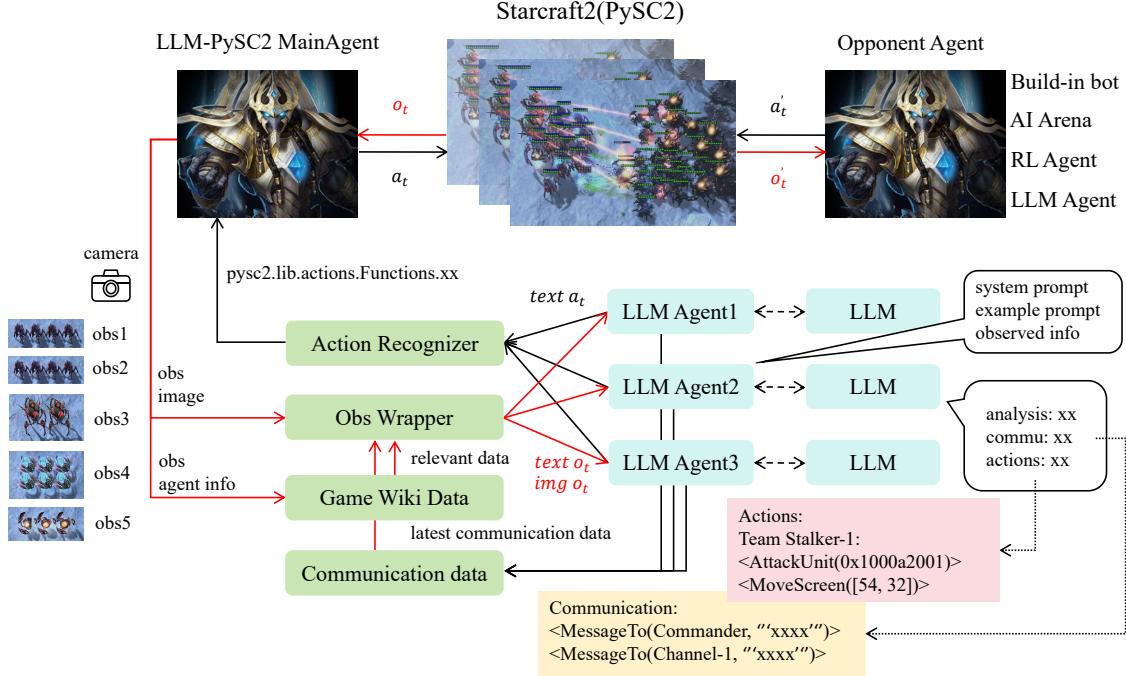


Figure 1: **LLM-PySC2 framework.** In LLM-PySC2, the original PySC2 observation will transform into a text-form observation. LLM-generated text action can be recognized and transformed into PySC2 action functions, enabling LLMs to interact with the StarCraft II environment.

##### 3.1.1 Interact with environment

An interaction step consists of two phases: auxiliary management and decision-making (and it consists of many game steps). In the auxiliary management phase, no LLM will be involved. The MainAgent will control the PySC2 camera and finish works like grouping newly trained units and managing idle workers to avoid excessive involvement of large models in simple and repetitive labour.

Observations for each agent's unit teams will be collected in the decision-making phase. After all teams' observations are collected, the agents use the Observation Wrapper to translate the structured observations into a text observation. Then, all agents query remote or local LLMs concurrently, waiting until all the agents get the response.

After all agents get the response, they will use the Action Recognizer to detect valid actions and translate the text actions into a structured form. Then, the MainAgent moves the camera to the same position when collecting observations and executes each agent's stored actions. After executing all the actions, the LLM-PySC2 environment will enter the next interaction step and repeat the work mentioned above.

##### 3.1.2 Multi-agent communication

Considering that LLMs have inherent advantages in interaction, we designed a communication system for the multi-agent framework. In the communication system, agents communicate with each other using 'communication actions', a kind of text action similar to unit control actions shown in Figure 1.

In the communication system. An LLM agent can send a message to another agent or send information to a channel. If the message is sent to an agent, only the designated receiver can get the information. If the message is sent to a channel, all agents that listen to the channel share the information. Through these communication actions, multi-agent collaboration frameworks such as centralized decision-making and distributed decision-making can be easily built.

### 3.2 Observation

Observation is indispensable for decision making. Different types of information are necessary for agents with different tasks. Roughly, we categorize observational information into two types: local observations for micro-level operations and global observations for macro-level decision-making. These observations can be divided into text and image observations according to form.

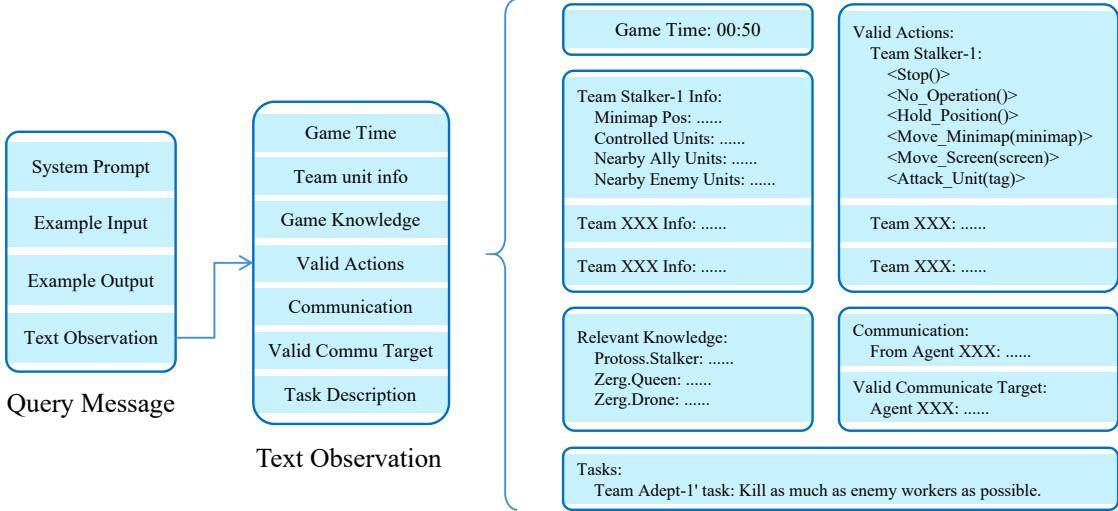


Figure 2: **Text observation for micro-operation LLM.** Text observation is a part of the query message. It contains many paragraphs, including team unit info, relevant game Knowledge, and valid actions. Semantic information is added when the observation wrapper processes the original obs object.

#### 3.2.1 Text Observation

**Observation Wrapper for micro-operations** This wrapper focuses on local observations. It provides detailed information of controlled unit, nearby ally and nearby enemy unit for an agent. It extracts unit information from PySC2 obs object and the relevant game knowledge from the knowledge base. As shown in Figure 2 The text observation generated by the wrapper includes game time, unit information, unit knowledge, valid actions, short-term memory, communication data, and task descriptions. Agents using the wrapper are designed for micro-operations like fighting with enemy units or constructing buildings in a specific position.

**Observation Wrapper for macro-decisions** This wrapper focuses on global observations. It provides deployment information, unit counts, and upgrade status that are similar to the text observation of the TSC2 environment. For the agent responsible for military deployment, text observation generated from the wrapper is used for supporting overall strategy. For the agent responsible for development, the generated global observation will make the agent aware of the current economy and technology situation, supporting the planning of future works of development.

#### 3.2.2 Image Observation

In the complex environment of StarCraft II, relying solely on textual observations may prevent agents from fully comprehending the battlefield dynamics. To enhance situational awareness, the LLM-PySC2 environment provides multimodal observation. This feature enables multimodal large models to integrate visual information, leading to a more accurate understanding of the situation. Figure 3 highlights two primary types of image observations: game image observation and feature map observation. These visual inputs provide the agent with critical battlefield information, facilitating tactical analysis and strategy development.

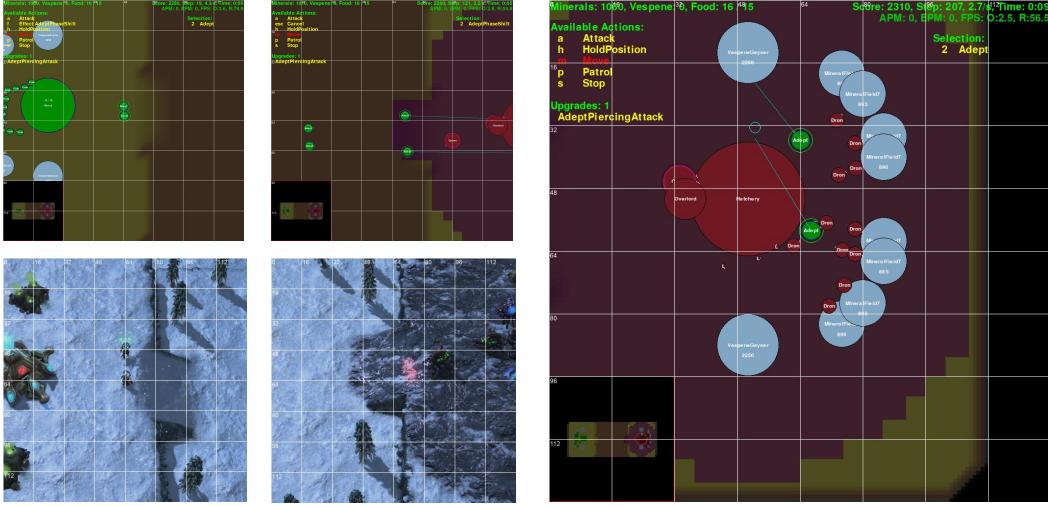


Figure 3: **Image observations.** LLM-PySC2 directly extracts PySC2’s image observation, including the game image and the feature map. It enhances these images by incorporating auxiliary lines, which assist LLMs in accurately determining the coordinates of various positions. These images, interpretable by multimodal LLMs, provide decision-makers with complex information, such as terrain layout and unit distribution on the map.

Figure 3 demonstrates the game image and feature map, which are directly extracted from the PySC2 interface. These images are enhanced with auxiliary lines to provide coordinate information for large models. This approach enables the agent to accurately perceive crucial battlefield elements, such as unit count and distribution, while also conveying information that is challenging to express through text, such as terrain features and relative spatial relationships.

### 3.3 Action

In decision-making environments, the concept of "action" is pivotal to enable interactions between the agent and the environment. In our framework, LLMs engage with the environment through text-based actions, which must adhere to a specific format to be recognized and converted into PySC2 action functions. The process of processing text action into PySC2 functions can be seen in Figure 4.

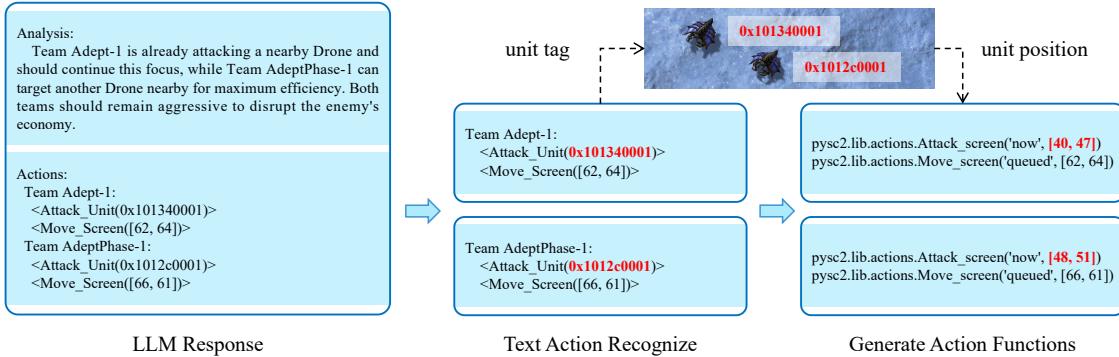


Figure 4: **Text action recognition.** The default action recognizer recognizes text actions by searching the "Actions" part in LLM’s response, extracting action names and arguments, searching for corresponding PySC2 functions, and generating the callback form of the function.

**Text Actions** These actions are expressed in a syntax that is intuitive and descriptive, allowing the LLM to comprehend the intended operation without additional context. A standard text action is encapsulated in angle brackets and several arguments, shaped as `<ActionName()>`, `<ActionName(arg0)>`, or `<ActionName(arg0, arg1)>`. The arguments can represent various elements, such as a unit tag, a screen coordinate, or a minimap position, allowing these actions to encompass the complete continuous action space of PySC2.

In the decision-making phase, LLM will be informed of currently available actions, such as <Attack\_Unit(tag)>, <Move\_Screen(screen)> and <Select\_Unit\_Attack\_Unit(tag, tag)>. The LLM can generate actions like <Attack\_Unit(0x100030001)> or <Move\_Screen([23, 37])> according to observed information and its purpose. If LLM generated multiple text actions, the first action will be executed immediately, and the remaining actions will be added to the action sequence waiting for execution.

**Action Space** All kinds of actions in PySC2 are available in our environment, however, each agent does not have to face all the actions of its race. In our environment, the action space is agent-specific, allowing each agent to define a unique set of actions. For the agent that controls units such as Stalkers, the action space consists of text actions like <Stop(), <No\_Op(), <Move\_Screen(screen)>, <Move\_Minimap(minimap)>, <Attack\_Unit(tag)>, and do not consist of actions like training units or research.

## 4 Experiments

### 4.1 Experiment Scenarios

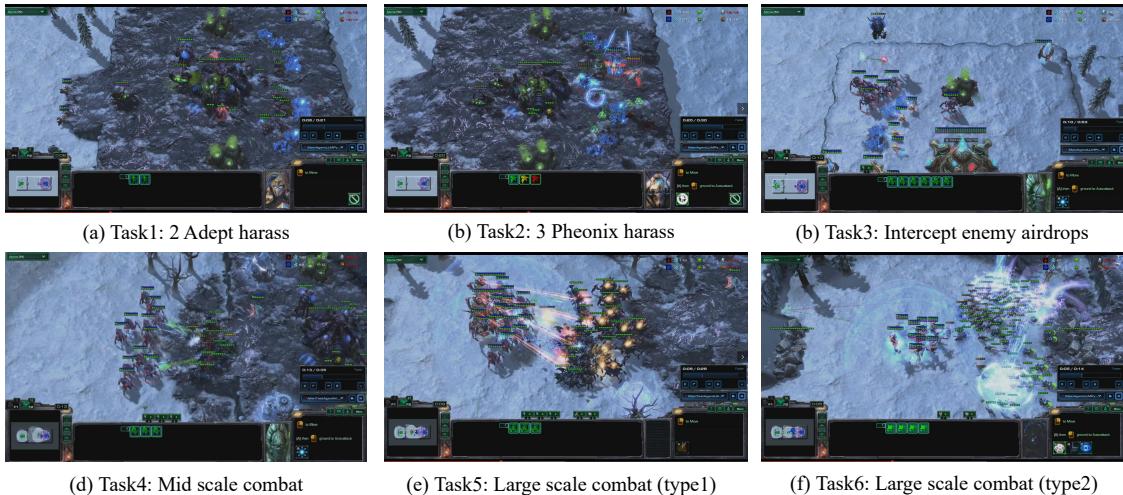
To facilitate research in LLM-based decision-making, we have provided two sets of experiments: LLM-SMAC tasks and LLM-PySC2 tasks. The LLM-SMAC tasks are the same as standard SMAC experiments, which serve as an excellent bridge for comparing with RL-based methods. LLM-PySC2 tasks are new scenarios, which, compared to the SMAC tasks designed specifically for micro-operations, place more emphasis on the large model’s ability to understand the task scenario and make macro-level decisions.

#### 4.1.1 LLM-SMAC tasks

LLM-SMAC tasks share the same settings as the original SMAC tasks. These tasks initialize units for both sides and automatically raise attacks for enemy units. In these scenarios, the key to victory lies in concentrating firepower, controlling combat distance, and, sometimes, in interaction frequency. They are good scenarios for comparing the training data efficiency with RL-based methods but not good scenarios for utilizing the multitasking and macro decision-making capabilities of LLMs.

#### 4.1.2 LLM-PySC2 tasks

LLM-PySC2 tasks are the newly proposed experiment scenarios, a task group that tests agents’ situation analysis capabilities, planning abilities, the application of knowledge, communication, and collaboration. Some of the tasks are shown in Figure 5



**Figure 5: LLM-PySC2 task group.** The LLM-PySC task group contains eight tasks, with three difficulties for each task. Compared to SMAC tasks, they place more emphasis on macro decision-making, situation analysis, and skill use. These scenarios are common in professional competitions. Winning these small tasks is beneficial for winning complete games in the future.

In these tasks, LLMs need to plan an infiltration route into the enemy base and kill enemy workers, or use unit skills to implement specific tactics in a battle. In addition, these tasks are more suitable for researching multi-agent collaboration methods, and implementing centralized or distributed decision-making for LLMs.

There are eight tasks in the LLM-PySC2 task group. Half of the task scenarios are single-agent decision-making scenarios (from tasks 1 to 4), where one LLM agent controls multiple units, while the other half (from tasks 5 to 8) tests the cooperation between agents, with multiple agents controlling multiple units with different tactical roles. In LLM-PySC2 task group, image observation and multi-agent communication are available and can be easily disabled if needed.

To avoid the situation where methods in SMAC can always reach the 100% winning rate of most tasks, we set three different difficulty levels for our experiment group. From level 1 to level 3, the forces of the enemy gradually increase. At a higher level, more units or upgrades will be added to the enemy side, ensuring these tasks can still be effective even after the LLM-based decision-making methods have been well developed.

## 4.2 Experiment Results

To facilitate subsequent research, we tested the decision-making ability of various large models. All experiments were conducted in StarCraft II of Version 5.0.13 (92440), LLM-PySC2 v0.1. We recorded a ratio of resources of the killed unit over the dead unit (K/D rate) and the winning rate (WR, i.e. task completion rate). The combination of K/D rate and WR reflects the performance of LLM in decision-making scenarios.

In the LLM-PySC2 environment, we provide series of LLMs, such as GPT-3, GPT-4, GPT-o1, GLM-4, Claude-3, Llama-3.1. We tested some representative models among them, tested the performance of models with different reasoning abilities in decision-making tasks (GPT-3.5, GPT-4o-mini, GPT-4o), and tested the performance of models with different parameters based on the same architecture (Llama3.1-8b, Llama3.1-70b, Llama3.1-405b).

All experiments use the default configuration of the open-sourced codes. As a benchmark, we do not specially design prompts to promote decision quality or instruct the LLMs to obtain victories, and all the LLMs are not fine-tuned in the LLM-PySC2 environment. Results show that large models can make decisions and generate text actions in the correct form. However, when the task is complex enough or requires a lot of micro-operations, large models may not perform well, suggesting that training or other technical methods are necessary for improving their decision quality.

### 4.2.1 Experiment Results in LLM-SMAC tasks

In the LLM-SMAC tasks, we conducted 20 repeated experiments for 6 LLMs in each scenario. For scenarios where decisions were made by GPT-3.5-turbo, we raised the number to 50 due to its good concurrency support and friendly cost. In these experiments, all large models used textual observations. This setting is completely sufficient for scenarios other than 2c\_vs\_64zg, as they basically did not need to utilize terrain information. Results are shown in Table 1.

Table 1: Kill/Death Rates and Winning Rates of LLMs in LLM-SMAC Tasks.

Model Name	2s3z	3s5z	1c3s5z	3s5z_vs_3s6z	2s_vs_1sc	2c_vs_64zg	3s_vs_3z
Gpt-3.5-turbo	0.60 (22%)	0.43 (4%)	0.91 (44%)	0.29 (0%)	<b>0.01 (2%)</b>	0.52 (0%)	0.05 (0%)
Gpt-4o-mini	0.66 (20%)	0.39 (0%)	<b>1.01 (50%)</b>	0.29 (0%)	0.00 (0%)	0.54(0%)	0.09 (0%)
Gpt-4o	0.76 (20%)	0.47 (0%)	0.80 (30%)	<b>0.35 (0%)</b>	0.00 (0%)	<b>0.56 (0%)</b>	<b>0.15 (0%)</b>
Claude3-haiku	0.58 (5%)	<b>0.48 (0%)</b>	0.48 (0%)	0.32 (0%)	0.00 (0%)	0.52 (0%)	0.10 (0%)
Llama3.1-8b	0.19 (0%)	0.23 (0%)	0.18 (0%)	0.14 (0%)	0.00 (0%)	0.49 (0%)	0.00 (0%)
Glm-4-plus	<b>0.81(25%)</b>	0.46 (0%)	0.47 (0%)	0.33 (0%)	0.00 (0%)	0.54 (5%)	<b>0.15 (0%)</b>

We found that, although large models can analyze the observation information and output actions in the correct form, they performed poorly in SMAC tasks. On the one hand, due to LLM hallucinations and the lack of task-specific knowledge, they can not deduce the principle that concentrated fire is the key to victory. On the other hand, even if the observation provided the knowledge that Zealots have a higher attack efficiency than Stalkers, the large models sometimes still chose to attack the enemy Stalkers first in tasks like 2s3z and 3s5z.

#### 4.2.2 Experiment Results in LLM-PySC2 tasks

The same as the experiments in SMAC, we conducted 20 repeated experiments for each large model in each scenario and 50 for GPT-3.5-turbo. Considering that all models cannot complete the multi-line attack in Task 8, we only listed the data from Task 1 to Task 7. Results are shown in Table 2 and 3.

Table 2: Kill/Death Rates and Winning Rates of Gpt-3.5-turbo in LLM-PySC2 Tasks (level-1/2/3).

Task level	task1	task2	task3	task4	task5	task6	task7
task-level-1	1.23 (58%)	0.13 (4%)	6.63 (38%)	0.38 (0%)	0.61 (8%)	0.28 (0%)	1.29 (72%)
task-level-2	0.56 (5%)	0.04 (0%)	3.31 (5%)	0.34 (0%)	0.52 (0%)	0.20 (0%)	0.98 (25%)
task-level-3	0.39 (0%)	0.05 (0%)	1.99 (0%)	0.31 (0%)	0.40 (0%)	0.26 (0%)	0.62 (0%)

In table 2, we tested Gpt-3.5-turbo’s performance in all the levels of each task. These data can serve as benchmark values for future research. These three levels of difficulty not only serve as validation scenarios for developed decision-making methods in the future but can also be applied to out-of-distribution (OOD) tasks, such as training on level 2 and validating on level 3.

Table 3: Kill/Death Rates and Winning Rates of LLMs in LLM-PySC2 Tasks (level-1).

Model Name	task1	task2	task3	task4	task5	task6	task7
Gpt-3.5-turbo	1.23 (58%)	0.13 (4%)	6.63 (38%)	0.38 (0%)	0.61 (8%)	0.28 (0%)	<b>1.29 (72%)</b>
Gpt-4o-mini	1.67 (70%)	0.16 (0%)	3.46 (0%)	0.39 (0%)	0.62 (20%)	0.30 (0%)	1.02 (40%)
Gpt-4o	<b>2.27</b> (80%)	0.16 ( <b>10%</b> )	<b>Inf (100%)</b>	<b>0.46</b> (0%)	TBD	TBD	TBD
Gpt-o1-mini	1.36 (60%)	0.04 (0%)	TBD	TBD	TBD	TBD	TBD
Claude3-haiku	2.19 ( <b>90%</b> )	0.19 ( <b>10%</b> )	5.25 (40%)	0.34 (0%)	<b>0.75 (25%)</b>	<b>0.33</b> (0%)	0.93 (45%)
Llama3.1-8b	0.28 (5%)	0.12 (5%)	14.9 (75%)	0.18 (0%)	0.48 (5%)	0.14 (0%)	0.71 (25%)
Llama3.1-70b	0.36 (15%)	0.14 (0%)	58.9 (95%)	0.33 (0%)	0.59 (15%)	0.31 (0%)	0.71 (30%)
Llama3.1-405b	0.70 (30%)	0.10 (0%)	3.0k( <b>100%</b> )	0.28 (0%)	0.56 (10%)	0.32 (0%)	0.47 (15%)
Glm-4-plus	0.78 (30%)	<b>0.21</b> (5%)	<b>153 (100%)</b>	0.38 (0%)	0.60 (10%)	0.30 (0%)	1.03 (55%)

Based on the data presented in Table 3, two conclusions can be extrapolated. First, adequate parameters for the large model are necessary for decision-making. Llama-3.1-8b, the model with minimum parameters, performs nearly the worst among all the models we tested, while the 70b and 405b models perform better than the 8b model. Second, improving reasoning ability does not lead to a linear improvement in decision-making ability. Although GPT-4o performed the best in most experiments, it still had a zero winning rate in some tasks that can easily be completed, such as task 4. These results lead to a conclusion: pre-trained large models cannot directly undertake complex decision-making tasks, and learning in deployment scenarios is almost inevitable.

## 5 Discussion

In the experiments, we found that there are several deficiencies in LLM-based decision making.

**Hallucinations.** Hallucination is the first problem that leads to bad decisions. Sometimes, LLMs confuse screen coordinates with minimap coordinates (as shown in Figure 6), or use unmentioned actions in the Valid Actions part of the observation. Sometimes, LLMs even damage teammate units. Hallucination has become an urgent problem in LLM-based decision making.

**Poor knowledge utilization.** Large models generally exhibit a significant deficiency in leveraging game-related knowledge. In task 2, game knowledge shows that Phoenix’s GravitonBeam ability will prevent the unit from moving and attacking. However, this ability is still overused, failing to obtain victories in task 2. In task 5, even the LLM knows the PurificationNova of Disruptor deals a lot of damage, they use the skill on injured units, causing a large amount of spillover damage.

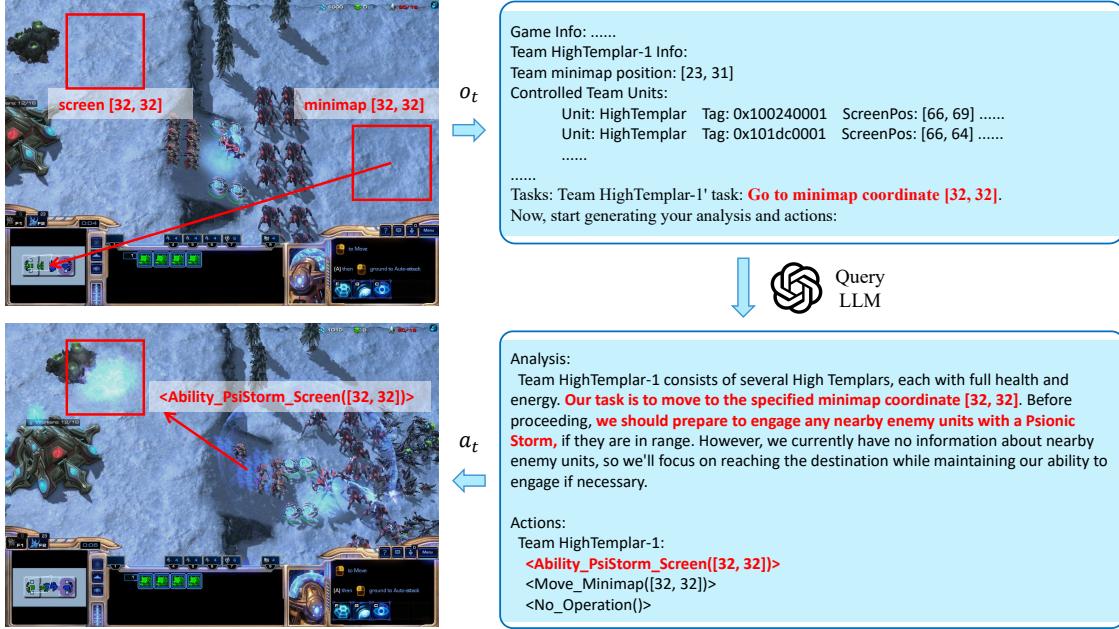


Figure 6: **LLM hallucination in decision-making.** This is an example that LLM confuse the screen coordinates and minimap coordinates, wasting a PsiStorm skill and 75 energy. This is just an instance of the hallucination of large models. In fact, there are many other forms of expression, such as attacking teammates and incorrectly choosing priority targets.

**Poor understanding of the world.** Lack of world understanding is a kind of lack of knowledge. Pre-trained LLMs are generally not trained in decision-making tasks. They do not know how to win in each task. In task 4, for example, the large model should use Stalker’s Blink ability to transfer injured units to the rear. However, this ability is rarely used, resulting in the unit’s death and a zero winning rate in task 4, even though the LLM is told that Blink is commonly used to pursue the enemy or retreat injured units.

**Low quality collaboration.** In the multi-agent tasks like task 5 to 8, LLM agents should collaborate with others and defeat the enemy together. However, we found it difficult for these agents to reasonably allocate targets, coordinate attack timing, and coordinate retreat timing, no matter whether they collaborate with or without a leadership/commander. How to improve the collaboration performance of LLM agents is important in building a high-level multi-agent decision-making system.

These issues hinder the application of LLMs in decision-making scenarios. Fortunately, there are many ways to improve the decision-making ability of large models. For example, directly providing knowledge to LLMs may directly improve their ability. However, providing LLMs knowledge or precisely annotated datasets usually demands quite a lot of resources. Self-supervised learning is still the most attractive way to enhance decision-making ability, either through reward-based or reward-free methods (such as LLM reflection), and either through parameter training or training-free techniques.

## 6 Conclusion

In this paper, we introduce a new environment for LLM decision-making, the first environment that accommodates continuous PySC2 actions, and the first LLM StarCraft II environment with a multi-agent framework and communication system. In experiments, we test mainstream LLMs’ performance in both the LLM-SMAC and LLM-PySC2 task groups, among which the LLM-PySC2 task group is a brand-new experimental scenario that we designed for large models. Results of baseline tests show that LLMs can make decisions, generating actions in the correct form. Still, the decision quality is relatively low and there are several problems like hallucinations, poor utilization of game knowledge, and lack of world understanding. Results indicate that learning in the deployment environment is necessary for LLM-based decision-making. We hope the LLM-PySC2 environment can promote research on LLM learning methods, helping LLM-based decision-making methods better adapt to task scenarios.

## References

- [1] Oriol Vinyals, Tim Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, and Demis Hassabis. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.
- [2] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv e-prints*, page arXiv:1803.11485, March 2018.
- [3] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv e-prints*, page arXiv:2006.10800, June 2020.
- [4] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. *arXiv e-prints*, page arXiv:2103.01955, March 2021.
- [5] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350 – 354, 2019.
- [6] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. *arXiv e-prints*, page arXiv:2304.03442, April 2023.
- [7] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023.
- [8] Meta Fundamental AI Research Diplomacy Team (FAIR)†, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.
- [9] Anonymous. Large language models play starcraft ii: Benchmarks and a chain of summarization approach. *arXiv preprint arXiv:2312.11865*, 2023. Accessed: 2024-10-21.
- [10] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 2186–2188, 2019.
- [11] Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yuetong Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. *arXiv preprint arXiv:2402.17574*, 2024.
- [12] Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. Exploring large language models for communication games: An empirical study on werewolf. *arXiv preprint arXiv:2309.04658*, 2023.