Instruction Following with Goal-Conditioned Reinforcement Learning in Virtual Environments

Zoya Volovikova[1,2,*]**, Alexey Skrynnik**[1,3]**, Petr Kuderov**[1,2] **and Aleksandr I. Panov**[1,2,3]

[1]AIRI, Moscow, Russia
[2]MIPT, Moscow, Russia
[3]FRC CSC RAS, Moscow, Russia

Abstract. In this study, we address the issue of enabling an artificial intelligence agent to execute complex language instructions within virtual environments. In our framework, we assume that these instructions involve intricate linguistic structures and multiple interdependent tasks that must be navigated successfully to achieve the desired outcomes. To effectively manage these complexities, we propose a hierarchical framework that combines the deep language comprehension of large language models with the adaptive action-execution capabilities of reinforcement learning agents: the language module (based on LLM) translates the language instruction into a high-level action plan, which is then executed by a pre-trained reinforcement learning agent.We have demonstrated the effectiveness of our approach in two different environments: in IGLU, where agents are instructed to build structures, and in Crafter, where agents perform tasks and interact with objects in the surrounding environment according to language commands.

1 Introduction

The ability to solve complex tasks, formulated in natural language, that require a long sequence of actions in the environment is a fundamental property of human intelligence. The recent significant success of large language models (LLMs) in instruction following and explanation generation demonstrates their powerful capabilities in solving commonsense, general knowledge, and code generation problems within the verbal domain. However, success rate of multi-step task completion for autonomous agents driven by general purpose LLMs is still low [19]. Moreover, LLMs are often trained solely on textual data, which limits their ability to understand and perform actions in real-world-like environments. Consequently, even ChatGPT [25] exhibits poor spatial reasoning [2]. On the other hand, reinforcement learning (RL) has proven effective in learning sequences of fine-grained actions for specific tasks within an environment. Thus, investigating the combination of LLMs for natural language understanding and high-level planning, along with RL for learning environmental manipulation, represents a promising research direction.

LLMs can be regarded as universal knowledge bases that allow human users to interact in natural language and solve complex tasks [12, 39]. Recent studies have shown that pre-trained LLMs can construct high-level action plans in both simulated and real environments [3, 22, 23]. However, these LLM-based approaches necessitate manual prompt engineering, handcrafted translation of language
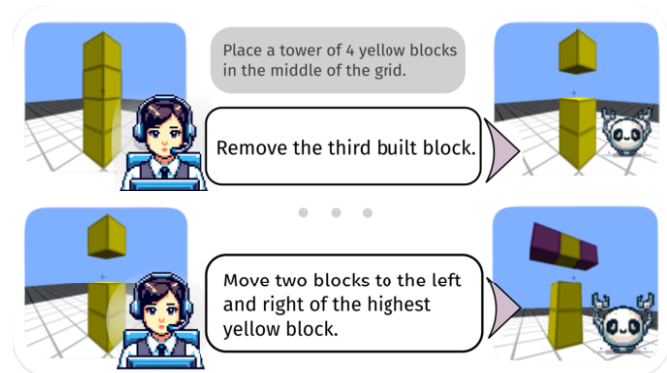

Figure 1: The task of collaborative interaction between the agent, the environment, and the user involves the following: the user provides instructions to the agent, and the agent executes actions within the environment to accomplish the task based on these instructions.

commands into embodied actions, and a strategy for goal-aware action selection from the distribution of potential options generated by language. In this context, several studies [1, 25] have demonstrated that the rough action plans extracted from language models can be refined using RL.

In our research, we present the hierarchical framework **IGOR (Instruction Following with Goal-Conditioned RL)**, which combines the capabilities of large language models for understanding complex natural language constructions with RL-based policies to develop effective behavior in the embodied environment. The framework relies on two main components: a **Language Module**, which translates instructions in natural language into a high-level action plan with generated subgoals, and a **Policy Module**, tasked with executing this plan and achieving subgoals.

Furthermore, we introduce independent learning strategies for each module. We have developed efficient learning strategies for the LLM for limited datasets. Some of these strategies are based on data augmentation using the ChatGPT model, while others rely on subdividing subtasks by altering data formats and decomposing these subtasks into primitives. We set a learning task for the RL agents based on goals and curriculum. This approach promotes highly efficient task execution in dynamic environments beyond the training sample.

The effectiveness of our approach was rigorously tested in two embodied environments. The first is IGLU [41], where the 'Builder'

* Corresponding Author. Email: volovikova@airi.net

agent constructs structures based on natural language instructions from the 'Architect' agent (see an example in Fig. 1). The second is the Modified Crafter environment, where the instruction-following agent needs to adapt to dynamically changing environments. Our results demonstrate that the application of our method not only outperforms the algorithms presented in the NeurIPS IGLU competition[1] [14, 13, 15], but also surpasses the baselines based on Dreamer-v3 in Crafter environment [10].

The **main contributions**[2] of our paper are:

1. We proposed a novel task decomposition approach that facilitates the incorporation of augmentations, curriculum learning, and potentially other techniques to multi-modal setups involving LLM and RL learning for virtual environments.
2. In addition to the known IGLU environment, we presented a modified Crafter environment by introducing a textual modality and prepared a corresponding dataset to support this enhancement.
3. We conducted extensive experiments to compare our approach with other methods in both the Crafter and IGLU environments, demonstrating significant improvements.

2 Related work

Planing with LLM. Recent studies are actively exploring the generation of action plans using language models. Some works focus on prompt engineering for effective plan generation [32], while others address the challenge of translating the language model's output into executable actions [12]. In [1], models are trained with RL for selecting feasible actions and executing elementary actions. Many of these works [31, 20] aim to control robots interactively in real time. Maintaining a dialogue with humans is an essential area of research for robotics [26, 9].

Language Grounding Problem. The language grounding problem in intelligent agents involves linking objects across modalities, such as matching textual instructions with objects in virtual environments. Methods to address this include using CLIP for visual-textual links [28], cross-attention mechanisms, and compressing data into hidden subspaces, exemplified by Dynalang [18]. Some strategies integrate language processing with reinforcement learning, using text embeddings as observations [11, 18]. Others connect textual descriptions to environmental entities using transformer models like Emma and EmBERT [37, 33]. Additionally, some approaches use multiple modules trained independently, with pre-trained language models aiding in planning and adapting actions, addressing the lack of real-world experience [12, 5, 17]. Innovatively, models like JARVIS-1 combine pre-trained memory blocks with tools like CLIP, enhancing multimodal memory and scheduling [38, 8].

Embodied environments. In the field of embodied reinforcement learning, several platforms have been developed to train agents based on text instructions. Among these, AI2Thor [16] and Habitat [34] , offer tasks that are simple and adhere to strict rules, which simplifies the process of linking actions to text using straightforward syntactic structures (Messenger [37], HomeGrid [18], TWOSOME [35]).

Furthermore, advancements have been made to enhance the Crafter [10] environment, resulting in the creation of the Text-Crafter [6] version. Similarly, the MineDojo [8] platform, which is based on Minecraft, has been introduced. These platforms are designed for more intricate linguistic and planning tasks. Additionally, the IGLU [15] environment stands out for its complexity. In IGLU,

agents must follow detailed instructions to construct structures within a virtual world. These environments are characterized by a vast state space and involve complex tasks that are formulated by humans.

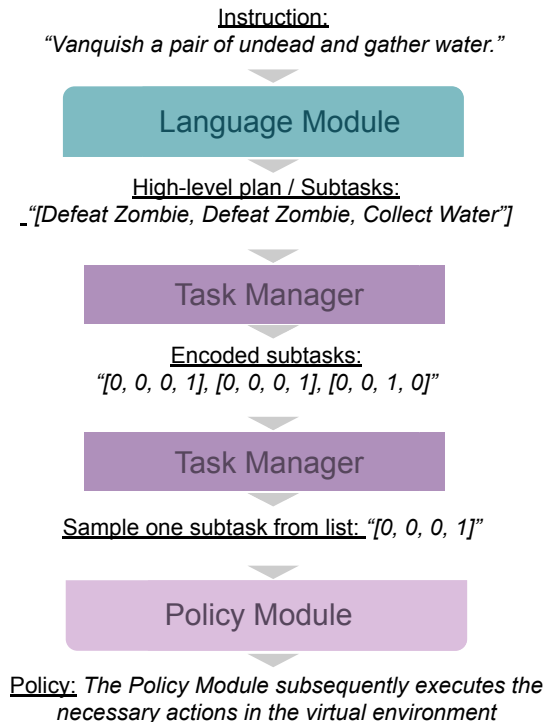## 3 IGOR: Follow Instruction with Goal-based RL

Figure 2: The IGOR framework has three modules: a Language module that solves language understanding problems and provides a high-level plan of subtasks, a Task Manager that encodes the subtasks for the Policy module, and a Policy module that executes actions in the environment based on visual observations and subtasks.

The IGOR framework is designed to solve the challenges of natural language understanding and instruction execution within virtual environments, enabling the processing of instructions that contain complex linguistic structures and specific terminologies. The virtual environments in which the intelligent agent operates are characterized by extensive observation areas and require the execution of multiple interconnected tasks. The framework is composed of three key modules.

The **Language Module**, implemented using Large Language Models (LLM), analyzes incoming instructions and converts them into a high-level plan consisting of a set of specific subtasks that need to be executed.

The **Policy Module**, implemented using reinforcement learning methods based on the Proximal Policy Optimization (PPO) algorithm, is responsible for the strategic interaction in the environment, including the execution of the interconnected tasks.

The **Task Manager** acts as a wrapper over the virtual environment, transforming the list of subtasks provided by the Language Module into a format understandable to the Policy Module. This module also ensures that the tasks specified in the instructions are completed and concludes the episode after their execution.

Thus, the **inference pipeline** operates by initially receiving instructions which are processed by the Language Module, where they

---

[1] https://www.iglu-contest.net/
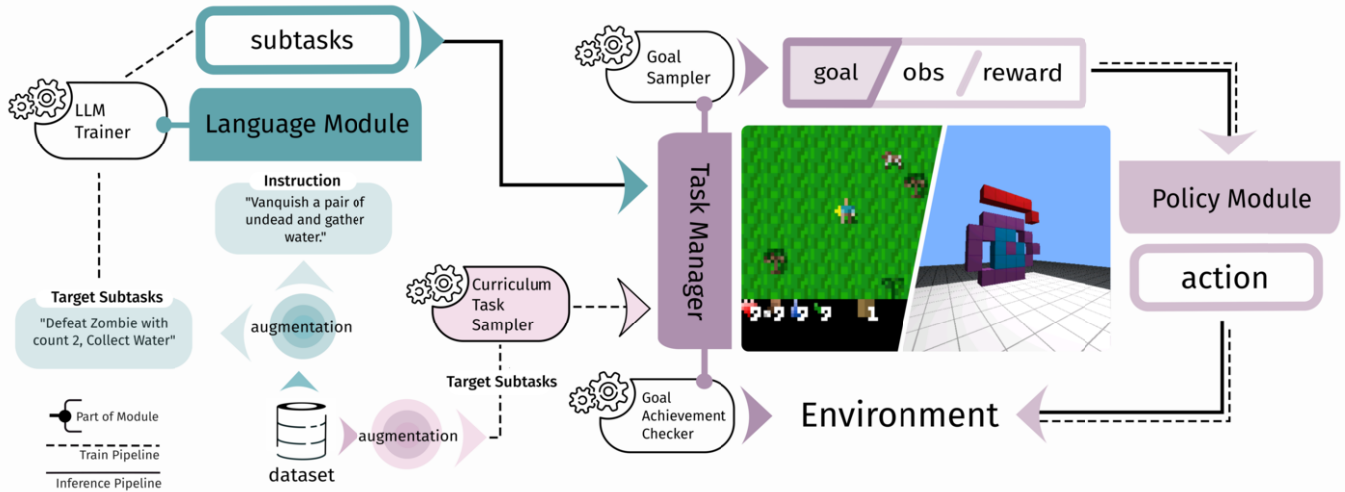[2] Our code is available at https://github.com/AIRI-Institute/IGOR

Figure 3: The diagram displays the IGOR system, where the "Language Module" transforms text instructions into subtasks. The "Task Manager" coordinates the subtasks and monitors their execution. The "Policy Module" operates in a virtual environment based on the subtasks. Dotted lines indicate the training process of the modules, while solid lines show how the modules interact during inference.

are translated into a series of subtasks — a high-level execution plan. Subsequently, the Policy Module executes the necessary actions in the virtual environmrnt to achieve the objectives outlined in the instructions. An example of using our pipeline can be found in the Figure (2)

Training for each learnable module is conducted separately, which provides flexibility in the integration of training methods and techniques. The training of the Language Module involves various augmentations and modifications to the dataset to prevent overfitting. The Policy Module is trained using a goal-based approach, which allows for training on a broader set of potential objectives than those available in the initial data. The inclusion of a curriculum in the training process also significantly enhances the quality of the final agent.

3.1   Language Module Training Techniques

The core of our language module utilizes a large pre-trained language model, which has been further finetuned on a dataset specific to the environment. This dataset contains information on how instructions translate into specific subtasks, enabling the model to understand and decompose complex commands effectively. The training leverages a specific format that maps instructions to their corresponding subtasks (e.g., Instruction -> Subtask 1, Subtask 2, Subtask 3).

Due to the difficulty of obtaining comprehensive datasets for training models to translate language instructions into commands, we face additional challenges. Typically, datasets for training large language models (LLMs) on such tasks are manually curated, which is a labor-intensive process. This often limits both the size and the quality of the datasets available. In response, we employ techniques to prevent overfitting, especially when working with these limited datasets. Our experiments demonstrate that these methods effectively enhance training quality by ensuring the model can generalize well from smaller, varied linguistic datasets, leading to a more robust understanding of instructions.

Augmentation with LLM. In this technique, we begin by understanding the structure and specific terminology of the dataset. Our approach involves iteratively modifying the list of subtasks necessary to execute a given instruction for each dataset element. Subsequently,

ChatGPT or another LLM is tasked with rewriting the instruction to incorporate these modified subtasks.

The prompt request is structured as follows:

1) Description of the Environment: Provide the LLM with a detailed understanding of the setting by explaining the overarching themes and key specific concepts. Use succinct descriptions for familiar ideas and detailed explanations for unique aspects.

2) Few-shot Example: Introduce the original instruction alongside its required subtasks. Optionally, you can also provide an example of how the instruction might change if the subtasks are altered.

3) Task Modification Request: Specify new subtasks for the target instruction and request the LLM to revise the instruction accordingly.

It is important to emphasize that the LLM is not creating the instruction from scratch. We aim to start with the existing instruction and suggest modifications, ensuring that the style of the original instruction is preserved as much as possible.

Subtasks decomposition. The second technique entails decomposing original subtasks into "primitives", which involves modifying the structure and format of subtasks in the dataset. Essentially, it suggests consolidating frequently co-occurring subtasks into a single common subtask, thereby reducing the data volume required for processing by the language model. These aggregated subtasks are termed "primitives".

We explore two methods for creating primitives. One method utilizes unique tokens, such as emojis, to encode each primitive. Emojis are chosen for their diverse range, making them a convenient means of representing a broad array of subtasks. The second method involves crafting primitives in a manner that aligns logically with the content of the instructions. This approach aims to enhance the coherence between the instructional context and the subtask structure and is the approach employed in our experiments.

3.2   Task Manager

The Task Manager serves as an intermediary to connect the Language Module and the Policy Module during execution and to allocate subtasks from the dataset to the Policy Module throughout its training phase. It retrieves a list of subtasks and systematically supplies these

to the RL agent as components of its observational input. Based on observations collected during the agent's interaction with the environment, the Task Manager determines whether a subtask has been successfully completed and decides whether to proceed to the next task or to conclude the episode. Upon successful completion of a subtask, the Task Manager assigns a positive reward ($r = +1$) to reinforce the agent's behavior.

3.3 Training the Policy Module

Instead of training a RL agent to tackle all subtasks required by a complex instruction simultaneously, we propose a **goal-based** learning approach. In this approach, each episode presents the agent with an observation and only one of the subtasks from the instruction. By doing this, we simplify the agent's task, reducing complexity of the task and facilitating more focused learning on specific aspects of the overall problem.

We examine a visual-based reinforcement learning environments characterized as a Partially Observable Markov Decision Process (POMDP). In this framework, the observation function is augmented by subtask encodings provided by the Task Manager. The primary objective of the agent is to develop a policy π that selects actions to maximize the expected cumulative reward. To achieve this, we employ policy gradient methods, specifically the Proximal Policy Optimization (PPO) algorithm. PPO has been demonstrated to offer substantial robustness across various tasks, attributed to its effective balancing of exploration and exploitation by optimizing a clipped surrogate objective function $\mathbb{E}\left[\min(\rho_t(\theta)\hat{A}_t, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$, where $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$ and $\hat{A}_t$ denotes the advantage estimate at time t [30].

During training we utilize curriculum task sampler that dynamically adjusts the probability of selecting subtasks based on their performance, inspired by curriculum learning approaches [21, 24]. Specifically, the selection probability for each task i in $\mathcal{T}$ is modified according to:

$$q_i = \begin{cases} \frac{1}{d} & \text{if } r_i \geq \tau \\ 1 + (\delta_i \cdot d) & \text{if } r_i < \tau \end{cases}$$

where r_i is the task's average reward, δ_i measures the variability in reward, d is a scaling coefficient, and τ is a success threshold. Probabilities are normalized using a softmax function to form a distribution from which tasks are sampled. A detailed description of the algorithm, its pseudocode and ablation study can be found in the Appendix of the full paper version [36].

4 Experimental Setup

To investigate and test the capabilities of intelligent agents, we have chosen environments with high combinatorial complexity. These environments allow us to assess how agents cope with tasks requiring the execution of many interrelated subtasks to achieve a target state.

The first environment is IGLU, where the agent's task is to build three-dimensional constructions based on textual descriptions. It is important to note that the complexity of the environment largely lies in the fact that, depending on the instructions, there can be a vast number of potential target states. To be successful in such an environment, an agent must possess advanced text interpretation skills, as well as the ability to think spatially and model ordering to adequately recreate the required structures.

The second environment is Crafter, where the agent needs to follow textual instructions to perform a variety of tasks, such as gathering resources and crafting items. This environment tests the agent's ability to understand natural language and effectively plan sequences of actions in response to changing conditions.

Below is the general pipeline for applying our approach to these virtual environments:

1. **Fine-tune the LLM**: Fine-tune the large language model (LLM) with an environment-specific dataset to translate instructions into subtasks for the reinforcement learning (RL) agent. Add environment-specific techniques if needed: augmentation, data modification, and dataset expansion.
2. **Train the RL agent**: Train the RL agent in a goal-based mode on environment-relevant subtasks. If needed, add environment-specific techniques: curriculum learning, hierarchical RL, reward shaping, and curiosity-driven exploration.
3. **Measure performance**: Measure performance on a test dataset for each environment. The LLM predicts subtasks from instructions, which the trained RL agent then executes.

4.1 IGLU Environment

Environment. IGLU is an environment[3] where an embodied agent can build spatial structures (figures) of blocks of different colors. The agent's goal is to complete a task expressed as an instruction written in natural language.

The observation space consists of point-of-view (POV) image $(64, 64, 3)$, inventory item counts (6), and the pitch and yaw angles (5). The agent can navigate over the building zone, place, and break blocks, and switch between block types. Additionally, the environment provides a textual observation in the form of a dialogue from the dataset, which defines a building task. The examples of such target tasks is presented in Fig. 4.
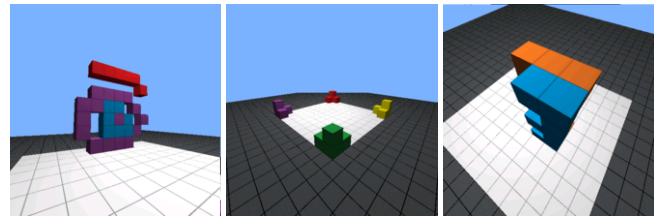

Figure 4: IGLU is a 3D environment where agents are tasked with constructing structures in a designated area, guided by descriptions provided in natural language and the agent's first person perspective.

Target utterances define the rest of the blocks needed to be added. The environment provides two modes of action for choice: walking and flying. In our experiments, we create agents for use in both flying and walking modes. The action space combines discrete and continuous subspaces: a discrete space of 13 actions (*noop, four camera rotation actions, break block, place block, choose block type 1-6*) and a set of 6 continuous actions for movement in all three directions.

Metrics. We employed the *F1* metric, as proposed in the IGLU competition, to evaluate the quality of the approaches.

Baselines. We have selected the three best solutions from the IGLU 2022 competition for comparison. The second and third-place solutions, which utilized T5 [29] and Pegasus [40] models respectively,

---

[3] https://github.com/iglu-contest/gridworld

are based on the organizers' proposed solution but differ in the NLP models employed. We also include the solution of the first-place team, called BrainAgent[4], whose approach differs significantly from the others. They used end-to-end learning, with the RL agent taking the embedding of a frozen NLP model as input, along with environment information and manually added features.

Dataset. The training and testing datasets contain 109 and 41 English instructions, respectively, with corresponding grids that need to be constructed. Each instruction consists of individual building steps with a corresponding voxel for each step.

For training the language model, we transform voxel shapes into text — a list of block coordinates that need to be placed. Grids are converted into a sequence of block coordinates and colors in the form of a string of tuples. For this dataset, a single block (its coordinates and a color) represents a single subtask in our terminology.

The order of the blocks in the sequence corresponds to the appearance of the blocks in the grid and have the following ordering rule: first by x, then by z, and then by y, where x and z are horizontal axes and y is vertical axis. In many instructions, the horizontal position of the first block is not specified, such that it can appear anywhere along x and z randomly. This is detrimental to the model during training as it will lead to high loss values for tokens that cannot be derived from the input data. Therefore, we change the sequence so that the first block of the first step of the instruction is placed at the center of the space, and the other blocks are shifted accordingly.

Figures Classification

Based on the capabilities an AI agent needs to construct figures in a dataset, we have developed our own rule-based classification of figures. This helps to more precisely determine which types of figures pose challenges and which descriptions of figure types might cause difficulties in construction by a language model. For visual examples, see Appendix [36]. The classes, defined by their spatial characteristics, are presented below:

- `flat` figures are characterized primarily by their length and width, with minimal emphasis on height.
- `tall` figures prioritize height over both length and width.
- `air` figures are distinguished by their lack of contact with the ground, signifying that they are airborne.
- `floor` figures are defined by their complete contact with the ground, resting entirely on the surface.

Training details. The *Language Module* for the IGLU environment was trained to predict, based on dialogue instructions, a list of coordinates of blocks that need to be placed or removed, where each coordinate corresponds to a subtask. We created a second dataset with a modified format of subtasks, grouping blocks that occur together in instructions into parallelepipeds using the technique described in 3.1. An example of the subtasks format for the IGLU dataset can be found in 1. Additionally, we augmented the dataset using ChatGPT, where we tasked the model to rotate the object by 180 degrees. The modification we applied for ChatGPT augmentation involved rotating the object by 180 degrees. We requested the LLM to adjust the building instructions to match this modification, ensuring the textual description aligned with the altered figure orientation.

As the base for the Language Module, the Flan-T5 base model was used. The LLM models are trained in a seq2seq manner, converting the instruction into a sequence of blocks as described above. Training hyperparameters and hardware parameters are listed in the appendix [36].

---

[4] https://github.com/kakaobrain/brain-agent

Table 1: Example of **coords** and **prims** subtask formats in the IGLU environment, corresponding to the instruction: *"<Architect> place 5 red blocks in a row, one row north of center."*

Format	Description	Example
coords	(x, y, z, colorID): • **x, y, z**: coordinates • **colorID**: block color id	*(0, 5, 5, 3), (0, 6, 5, 3), (0, 7, 5, 3), (0, 8, 5, 3), (0, 9, 5, 3)*
prims	(start), (size), rotation, color: • **(start)**: initial block (x, y, z) • **(size)**: dimensions (x, y, z) • **rotation**: alignment • **color**: block color name	*(0, 5, 5), (1, 1, 5), eastsky, red*

For training *Policy Module* in the IGLU environment, our objective is to train the agent to either place or remove a single block on a virtual field. At the beginning of an episode, a list of subtasks is randomly generated or taken from a training dataset. A specific block from this list is then selected as the target. Depending on this selection, we generate the initial state of the environment by positioning the blocks leading up to the target block.

The reward mechanism used during training is the same as proposed by the competition's authors.

To enhance the quality of training for the RL agent, we incorporate a curriculum that controls the complexity of the list of subtasks. This structured approach helps in progressively challenging the agent, thus improving its learning efficiency and capability to handle complex tasks.

Solving visual goal-based tasks can be accomplished by a wide range of modern RL approaches, both model-free [4, 7, 27] and model-based [10, 11]. In this work, we used an open-source implementation of asynchronous PPO[5] [27], which demonstrated better performance in terms of GPU hours to final score based on our preliminary experiments.

4.2 Crafter Environment

Figure 5: Crafter is a 2D environment reminiscent of Minecraft, where players must gather food and water, acquire resources, fend off creatures, and construct tools.

Environment. In the original Crafter environment[6], agents are primarily tasked with exploration, gathering resources, and battling monsters. We have adapted this framework so that each episode now provides a free-form textual instruction, directing the agent to perform the number of specific tasks. In our modified version, each task given to the agent corresponds exactly to one achievement in

---

[5] https://github.com/alex-petrenko/sample-factory
[6] https://github.com/danijar/crafter

the original game. Thus, within a single episode, following these instructions means completing individual achievements directly tied to the game's original goals. The examples of the agent in Crafter environment is presented in Fig. 5.

Dataset. To create a dataset of textual tasks for the agent in the Crafter environment, we followed a specific pipeline: First, based on the known list of tasks and achievements in the crafter environment, we randomly generated a list of subtasks. Second, to obtain descriptive instructions for these tasks, we used Mistral-7B-Instruct[7]. The example of such prompt and response is presented in Fig.6. Third, to obtain more varied descriptions of instructions, using the same Mistral model to stylize the received instructions (for example of such reformulation see Fig. 7).

Example of prompt request:

```
Rewrite the list of subtasks into one
instructive sentence for an agent in a virtual
environment, replacing words from the list with
synonyms. Subtasks: ["Defeat Zombie", "Collect
Iron with count 1", "Make Iron Sword"]
```

Example of response:

```
Vanquish the undead foe, gather a single unit
of metallic mineral, and forge an iron weapon.
```

Figure 6: The example depict prompt and response interactions used to generate the dataset for the modified Crafter environment.

Example of prompt request:

```
Rewrite the instruction in the style of Tolkien
with the same SHORT sentence size adding
causality. Instruction: Vanquish the undead
foe, gather a single unit of metallic mineral,
and forge an iron weapon.
```

Example of response:

```
Slay the wights, gather a nugget of mithril,
and forge a blade of iron.
```

Figure 7: This figure shows an examples prompt and response for the reformulation of an instruction generated by the Mistral-7B-Instruct.

Metrics. To assess the quality of an agent's performance in an environment where it follows language instructions, we used a metric called "Success Rate." This metric measures whether all the subtasks required by the given instructions to the agent were completed. Accordingly, for one instruction, the metric can be 1 if the agent completed all the tasks, or 0 if any task was not completed.

Baselines. We compared our IGOR Agent with Dynalang [18]. To run Dynalang on a text-adapted Crafter, we employ the T5 model to encode the text, mirroring the approach taken by the paper's authors. In each episode, we sequentially transmit all instructions and images to Dynalang, one token at a time. This process allows us to evaluate Dynalang's ability to identify the necessary subtasks from the instructions, aligning with our methodology. We adopt the same reward system used in our experiments.

Training details. A Language module for the crafting environment was trained on the provided data without any modifications. Following the instructions within this environment, the module is tasked with determining which achievements the agent should collect in the

---

virtual setting. Additionally, the Flan-T5-base model was utilized as a reference.

For Policy Module we use self-supervised goal sampling to train the agent on these subtasks efficiently. In this approach, each training episode for the agent targets a distinct goal selected randomly, aligning with the goals possibly derived from the LLM. The reward function for the RL agent is designed to provide positive feedback for the successful completion of subtasks, motivating the agent to achieve its goals. This function uses both the original Crafter environment reward and an extra reward for the completion of a subtask by the agent. The original reward is scaled by a factor of 0.1, and the extra reward for subtasks accomplishing is set to 1. Through this process, the agent learns to solve individual subtasks, contributing to the overall action plan devised by the LLM.

5 Experimental Results

In this section, we compare our approach with other state-of-the-art methods. The section is organized as follows: First, we present the results in the modified Crafter environment. Second, we show the results for the IGLU environment, alongside ablation experiments for different data processing techniques.

5.1 Crafter Environment

The results of IGOR and Dynalang in the Crafter environment are shown in the bar chart presented in Fig. 8. The x-axis displays various categories, where the **Total** category represents the overall success rate across all test instructions. The other categories illustrate the agents' performance on specific subtasks within the instructions, highlighting how effectively each agent solves individual subtasks when attempting to complete the entire instruction. For instance, a low value for a particular subtask on the plot indicates that the agent often fails to complete the instructions when that subtask is involved.
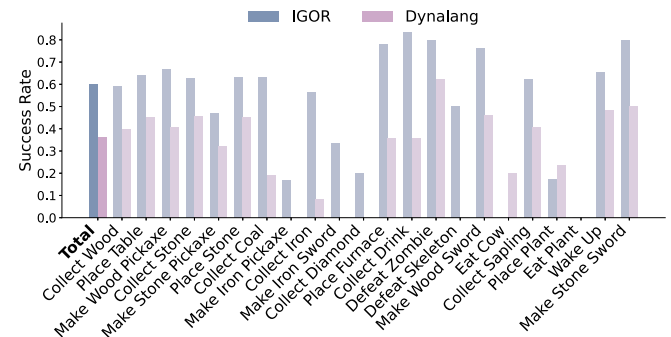

Figure 8: A comparison of the performance of IGOR and Dynalang approaches, on the Crafter environment with textual tasks. The success rate metric for each subtask is averaged across instructions requiring that specific subtask. *Total* bar represents the overall success rate for all instructions of test dataset.

Overall, the total score of IGOR is 0.6, indicating that the agent successfully completes all required subtasks in 60% of cases, significantly outperforming Dynalang, which only achieves 36.4%. Our approach demonstrates superior performance in 19 out of 22 subtasks. Furthermore, it achieves results that are at least twice as good in 8 out of 22 types of instructions. Additionally, our approach, with the integration of curriculum learning, shows strong capabilities in managing tasks that are especially challenging, such as *Collect Diamond*, *Collect Iron*, *Make Iron Sword*, and *Make Iron Pickaxe*.

5.2 IGLU Environemnt

The comparative results in the IGLU environment are presented in Table 2. We ran the IGOR approach in both Flying and Walking modes. Additionally, we conducted experiments with different data variants for training the LLM agent, utilizing both primitives and coordinates as subtasks. Across all configurations, our method consistently outperforms the competitors, achieving significantly better results in all figure categories. Notably, IGOR's lowest score is 0.45, which surpasses the overall performance of the winning team, BrainAgent, which scored 0.36. For simpler figures such as Floor using primitives, our agent scores 0.75 compared to BrainAgent's 0.53. A clear advantage is also observed for more complex tasks such as Air, where IGOR scores 0.46 versus 0.25. Even when using coordinates as subtasks, our agent's advantage, although reduced, persists across all figure types.

Table 2: A comparative analysis of the F1 score (higher is better) for the IGOR approach versus the IGLU-2022 competition winners on the test dataset.

Approach	Prim	Flying	Floor	Flat	Tall	Air	Total
IGOR	✓	✓	0.72	**0.51**	**0.46**	**0.40**	**0.52**
IGOR	✗	✓	0.68	0.44	0.36	0.34	0.45
IGOR	✓	✗	**0.75**	0.46	0.32	0.33	0.46
IGOR	✗	✗	0.68	0.42	0.29	0.31	0.45
BrainAgent	n\a	✗	0.53	0.36	0.23	0.25	0.36
MHB	n\a	✗	0.08	0.05	0.03	0.04	0.05
Pegasus	n\a	✗	0.08	0.06	0.02	0.04	0.06

Comparing Data Processing Technics.   We evaluated various data processing strategies for training NLP models, as detailed in Table 3. These strategies included expanding the original dataset (strategy "Coords"), employing the ChatGPT augmentation technique outlined previously (strategy "ChatGPT"), and modifying the dataset's format (strategy "Prim"). We also enhanced the dataset with color augmentation by altering the color of the block and its corresponding descriptor in the instructions (strategy "Color").

Table 3: A comparative analysis of various data processing strategies for LLM training. The table displays the F1 scores, which measure discrepancy between LLM-predicted figures and target figures. The highest scores in each category are highlighted in bold.

Prim	Color	ChatGPT	Floor	Flat	Tall	Air	Total
✓	✓	✓	**0.75**	0.51	**0.44**	**0.36**	**0.50**
✓	✓	✗	0.73	**0.52**	0.44	0.35	**0.50**
✓	✗	✗	0.69	0.42	0.34	0.25	0.42
✗	✓	✓	0.53	0.40	0.33	0.29	0.39
✗	✓	✗	0.39	0.36	0.33	0.29	0.35
✗	✗	✗	0.45	0.35	0.30	0.28	0.34

It can be observed that adaptation to primitives, even in the absence of augmentations, improves the F1 metric by 0.08 (from 0.34 to 0.42). When combined with augmentations, the F1 score sees a 0.16 boost (0.50) versus 0.01 increase via augmentations alone on coordinates (0.35).

While the ChatGPT augmentations don't yield a significant increase on primitives, an analysis of the F1 scores across different shape categories reveals their effectiveness. On coordinates, the ChatGPT augmentation outperforms color augmentations, showing a 0.4 relative improvement (0.35 vs 0.39).

This suggests that the augmentation strategies have different impact depending on subtask format. Additionally, it is apparent that the primary challenge for the language model arises from instructions requiring the prediction of blocks suspended in the air—these tasks exhibit the highest error rates. This may be due to the limited number and specific nature of such instructions in the training dataset.

6 Limitations

Exploring previous studies in this area reveals that creating a general approach for multimodal environments presents a significant challenge [42]. While many approaches show potential, they often underperform when applied outside their original domains. Aiming for a universally applicable method is an admirable goal, yet it currently has limited practical use.

Our work introduces a new framework that separates the training of RL and LLM components. The proposed decomposition aims to achieve architectural flexibility, facilitating the easy integration of various techniques such as curriculum learning and reward shaping for RL, as well as data augmentation and human feedback for LLM training. However, our approach requires the identification of known subtasks, which may not be present in some domains, thus necessitating additional data collection to identify them.

7 Conclusion

In this paper, we introduce the Instruction Following with Goal-Conditioned Reinforcement Learning in Virtual Environments (IGOR) method, a novel approach that translates natural language instructions into a sequence of executable subtasks. The IGOR method integrates two independently trainable modules along with an intermediary Task Manager module. The Language Module is responsible for converting textual instructions into a defined sequence of subtasks. Meanwhile, the RL agent is structured in a goal-conditioned manner, making it adept at handling a wide array of tasks across various environmental contexts.

The modular decomposition of the IGOR approach allows for the incorporation of additional training techniques such as data augmentation, goal-based strategies, and curriculum learning in RL. A detailed analysis of the results shows that this approach not only allows for more flexible training customization but also yields a significant improvement in performance.

We demonstrate these advantages through experiments in two environments. In the IGLU environment, where the agent is required to construct a figure in a virtual setting based on dialogue with a human, our method surpasses the winners of the IGLU 2022 competition. Additionally, we have adapted the Crafter environment to require the agent to achieve specific achievements based on instructions, showing that our approach outperforms the Dynalang method based on Dreamer V3.

References

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, Q. V. Do, Y. Xu, and P. Fung. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity, 2023.

[3] C.-P. Bara, C.-W. Sky, and J. Chai. Mindcraft: Theory of mind modeling for situated dialogue in collaborative tasks. In *Proceedings of the*

2021 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2021.

[4] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[5] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. PMLR, 2023.

[6] Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR, 2023.

[7] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

[8] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.

[9] X. Gao, Q. Gao, R. Gong, K. Lin, G. Thattai, and G. S. Sukhatme. Dialfred: Dialogue-enabled agents for embodied instruction following. *IEEE Robotics and Automation Letters*, 7(4):10049–10056, 2022.

[10] D. Hafner. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1W0z96MFEoH.

[11] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[12] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

[13] J. Kiseleva, Z. Li, M. Aliannejadi, S. Mohanty, M. ter Hoeve, M. Burtsev, A. Skrynnik, A. Zholus, A. Panov, K. Srinet, et al. Interactive grounded language understanding in a collaborative environment: Iglu 2021. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 146–161. PMLR, 2022.

[14] J. Kiseleva, A. Skrynnik, A. Zholus, S. Mohanty, N. Arabzadeh, M.-A. Côté, M. Aliannejadi, M. Teruel, Z. Li, M. Burtsev, et al. Iglu 2022: Interactive grounded language understanding in a collaborative environment at neurips 2022. *arXiv preprint arXiv:2205.13771*, 2022.

[15] J. Kiseleva, A. Skrynnik, A. Zholus, S. Mohanty, N. Arabzadeh, M.-A. Côté, M. Aliannejadi, M. Teruel, Z. Li, M. Burtsev, et al. Interactive grounded language understanding in a collaborative environment: Retrospective on iglu 2022 competition. In *NeurIPS 2022 Competition Track*, pages 204–216. PMLR, 2023.

[16] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[17] S. Li, X. Puig, C. Paxton, Y. Du, C. Wang, L. Fan, T. Chen, D.-A. Huang, E. Akyürek, A. Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

[18] J. Lin, Y. Du, O. Watkins, D. Hafner, P. Abbeel, D. Klein, and A. D. Dragan. Learning to model the world with language. *ArXiv*, abs/2308.01399, 2023. URL https://api.semanticscholar.org/CorpusID:260438420.

[19] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

[20] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

[21] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[22] S. Y. Min, D. S. Chaplot, P. K. Ravikumar, Y. Bisk, and R. Salakhutdinov. FILM: Following instructions in language with modular methods. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=qI4542Y2s1D.

[23] M. Murray and M. Cakmak. Following natural language instructions for household tasks with landmark guided search and reinforced pose adjustment. *IEEE Robotics and Automation Letters*, 7(3):6870–6877, 2022.

[24] M. Nesterova, A. Skrynnik, and A. Panov. Reinforcement learning with success induced task prioritization. In *Mexican International Conference on Artificial Intelligence*, pages 97–107. Springer, 2022.

[25] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[26] A. Padmakumar, J. Thomason, A. Shrivastava, P. Lange, A. Narayan-Chen, S. Gella, R. Piramuthu, G. Tur, and D. Hakkani-Tur. Teach: Task-driven embodied agents that chat. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2017–2025, 2022.

[27] A. Petrenko, Z. Huang, T. Kumar, G. Sukhatme, and V. Koltun. Sample factory: Egocentric 3d control from pixels at 100000 fps with asynchronous reinforcement learning. In *International Conference on Machine Learning*, pages 7652–7662. PMLR, 2020.

[28] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:231591445.

[29] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140): 1–67, 2020.

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

[31] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[32] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

[33] A. Suglia, Q. Gao, J. Thomason, G. Thattai, and G. S. Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion. *ArXiv*, abs/2108.04927, 2021. URL https://api.semanticscholar.org/CorpusID:236975859.

[34] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[35] W. Tan, W. Zhang, S. Liu, L. Zheng, X. Wang, and B. An. True knowledge comes from practice: Aligning llms with embodied environments via reinforcement learning. *ArXiv*, abs/2401.14151, 2024. URL https://api.semanticscholar.org/CorpusID:267212097.

[36] Z. Volovikova, A. Skrynnik, P. Kuderov, and A. I. Panov. Instruction following with goal-conditioned reinforcement learning in virtual environments. 2024. URL https://arxiv.org/abs/2407.09287.

[37] H. J. A. Wang and K. Narasimhan. Grounding language to entities and dynamics for generalization in reinforcement learning. *ArXiv*, abs/2101.07393, 2021. URL https://api.semanticscholar.org/CorpusID:231639188.

[38] Z. Wang, S. Cai, A. Liu, Y. Jin, J. Hou, B. Zhang, H. Lin, Z. He, Z. Zheng, Y. Yang, X. Ma, and Y. Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv preprint arXiv: 2311.05997*, 2023.

[39] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[40] J. Zhang, Y. Zhao, M. Saleh, and P. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

[41] A. Zholus, A. Skrynnik, S. Mohanty, Z. Volovikova, J. Kiseleva, A. Szlam, M.-A. Coté, and A. I. Panov. IGLU Gridworld: Simple and Fast Environment for Embodied Dialog Agents. In *CVPR 2022 Workshop on Embodied AI*, 2022. URL http://arxiv.org/abs/2206.00142https://embodied-ai.org/papers/2022/12.pdf.

[42] V. Zhong, A. W. Hanjie, S. I. Wang, K. Narasimhan, and L. Zettlemoyer. Silg: The multi-environment symbolic interactive language grounding benchmark. *arXiv preprint arXiv:2110.10661*, 2021.