CLIP4MC: An RL-Friendly Vision-Language Model for Minecraft

Ziluo Ding[12*] Hao Luo[12*] Ke Li[2] Junpeng Yue[23] Tiejun Huang[12] Zongqing Lu[12†]

[1]PKU [2]BAAI [3]TJU

Abstract

One of the essential missions in the AI research community is to build an autonomous embodied agent that can attain high-level performance across a wide spectrum of tasks. However, acquiring reward/penalty in all open-ended tasks is unrealistic, making the Reinforcement Learning (RL) training procedure impossible. In this paper, we propose a novel cross-modal contrastive learning framework architecture, CLIP4MC, aiming to learn an RL-friendly vision-language model that serves as a reward function for open-ended tasks. Therefore, no further task-specific reward design is needed. Intuitively, it is more reasonable for the model to address the similarity between the video snippet and the language prompt at both the action and entity levels. To this end, a motion encoder is proposed to capture the motion embeddings across different intervals. The correlation scores are then used to construct the auxiliary reward signal for RL agents. Moreover, we construct a neat YouTube dataset based on the large-scale YouTube database provided by MineDojo. Specifically, two rounds of filtering operations guarantee that the dataset covers enough essential information and that the video-text pair is highly correlated. Empirically, we show that the proposed method achieves better performance on RL tasks compared with baselines.

1. Introduction

Recently, autonomous agents have had great success in Atari games [19], Starcraft [23], Dota2 [3], and Go [27]. However, these popular works have also been criticized for poor generalization, *i.e.*, agents cannot generalize beyond a very specific set of tasks, unlike humans that continuously learn from open-ended tasks. Thus, building an autonomous embodied agent that can attain high-level performance across a wide spectrum of tasks has been one of the greatest challenges facing the AI research community.

One main challenge is that acquiring reward/penalty in all open-ended tasks is unrealistic, making the Reinforcement Learning (RL) training procedure impossible. To this end, MineDojo [9] has proposed an internet-scale, multi-modal knowledge base of YouTube videos to facilitate learning in open-ended settings. With the advent of a such large-scale database, agents are possible to harvest practical knowledge encoded in large amounts of media like human beings. Moreover, a video-text contrastive framework, MineCLIP [9], is proposed to utilize the internet-scale domain knowledge. In more detail, the learned correlation score between the visual observation and the language prompt can be used effectively as an open-vocabulary, massively multi-task reward function for RL training. Therefore, no further task-specific reward design is needed for open-ended tasks.

Figure 1: Key entities in our YouTube dataset.

However, the autonomous embodied agent requires the contrastive learning framework to provide a more instructive correlation score. Give the partial observations, *e.g.*, video snippet, and the language prompt that describes the task, the agent needs to figure out three non-trivial matters to better evaluate the current state. First, *whether the target entities are present within its field of vision?* MineCLIP has addressed this question. Second, *whether the agent has made the right action toward the right target?* Unlike vision tasks, RL tasks concern how agents ought to take action. Namely, we should measure the similarity between the video snippet and the language prompt at the action level, not only

*These authors contribute equally to this work.

†Correspondence to Zongqing Lu<zongqing.lu@pku.edu.cn>, Ziluo Ding <ziluo@pku.edu.cn>, Hao Luo <lh2000@pku.edu.cn>

at the entity level. Third, *what is the relationship between each video snippet and the degree of completion of the task?* To illustrate, the procedure of completing one single task contains numerous video snippets. However, different video snippets represent different levels of completion of the task even though they may have similar correlation scores with the language prompt. The higher level of completion of the task, the closer the video snippets are to the target state. Thus, we argue it is important to incorporate the level of completion of the task into the reward function. At the current stage, we only attempt to address the second problem so that the contrastive learning framework can be more RL-friendly.

In this paper, we propose an upgraded cross-modal contrastive learning framework, CLIP4MC, to provide a more RL-friendly reward function. Specifically, a motion encoder is proposed to extract the motion features of the video snippet. Intuitively, the atomic actions are captured in the difference between two adjacent frames. A sequence of such actions corresponds to the behavior described in the language prompt. Therefore, we stack the two frames with different intervals as atomic action representations of different amplitudes. After extracting motion features from those representations, we need to summarize a series of motion features at the same interval as the motion embedding and multi-interval motion embeddings as the final embedding. Motion embedding, together with video embedding, is used to measure the correlation score with the language prompt at both entity and action levels. Based on the score, we can render the auxiliary reward signal of the current state to the agent. The RL training is finally possible.

Though a large-scale database is provided by MineDojo, it contains significant noise due to its nature as an online resource. Since MineDojo has not released the training dataset for MineCLIP, we attempt to construct a neat YouTube dataset to facilitate the learning of basic game concepts. In more detail, we have done two rounds of filtering operations to guarantee two things. One is that the dataset covers enough essential information, including key entities and basic semantic events. Another is that there is a stronger content correlation between the video and transcript clips. Note that our proposed method is trained on our YouTube dataset, and we evaluate our method on MineDojo Programmatic tasks, including Harvest and Finding. Results show that our method can provide a more friendly reward signal for the RL training procedure. In addition, we validate various design choices of CLIP4MC through extensive ablation studies.

Our main contributions, among others, are:

- We bring up three key issues we need to solve for learning a vision-language model as the universal reward function for open-ended tasks.
- We build and release a neat vision-language dataset for Minecraft using the YouTube videos from MineDojo.
- We propose an RL-friendly vision-language model, CLIP4MC, which aligns actions implicitly contained in the video and transcript clips in addition to entities.

2. Related Work

Video-Text Retrieval. Video-text retrieval plays an essential role in multi-modal research and has been widely used in many real-world web applications. Recently, the pre-trained models have dominated this line of research with noticeable results on both zero-shot and fine-tuned retrieval. Especially, BERTs [6], ViTs [7], and CLIP [22], are used as the backbones to extract the text or video embedding. The cross-modal embeddings are then matched with specific fusion networks to find the correct video-text pair.

In more detail, CLIP4CLIP [18] proposes three different similarity modules to calculate the correlation between video and text embeddings. HiT [17] performs hierarchical matching at two different levels, *i.e.* semantic level and feature level. Note that semantic level and feature level features are from the transformer network's higher and lower feature layers, respectively. Frozen [1] proposes a dual encoder architecture that utilizes the flexibility of a transformer visual encoder to train from images or video clips with text captions. Moreover, MDMMT [8] adopts several pre-training models as encoders and it shows the CLIP-based model performs the best. Therefore, our model follows this line of research by using the pre-trained model, CLIP [22], to extract the feature embeddings.

Minecraft for AI Research. As an open-world video game with an egocentric vision, Minecraft is a splashy and important domain in RL due to the nature of the sparse reward, large exploration space, and long-term episodes. Since the release of the Malmo simulator [13] and later the MineDojo simulator [9], a series of methods [28, 26, 10, 16] have attempted to train an agent to complete tasks in Minecraft. Approaches such as hierarchical RL, goal-based RL, and reward shaping have been adopted to alleviate the sparse reward and exploration difficulty for the agent. Recently, DreamerV3 [12] succeeds in training agents in Minecraft with a learned world model. More recently, DEPS [29] introduces a pre-trained large language model for sub-goal planning, combined with goal-based bottom-level policy [4] learned from behavior cloning of offline data to successfully complete a series of Minecraft tasks.

In addition, some works attempt to incorporate the human player experience. MineRL [11] collected 60M player demonstrations with action labels, motivating some methods [25, 15] based on behavior cloning. As well-labeled data is limited in content, some works instead attempt to use vast and diverse but unlabelled data from the Internet. MineDojo

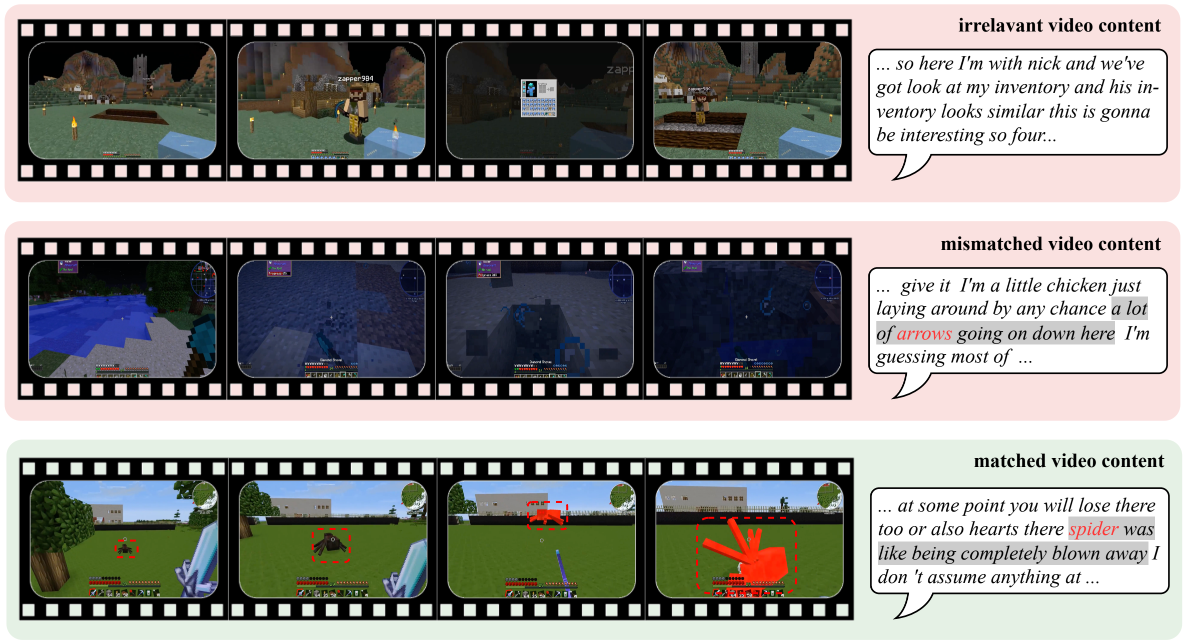

irrelavant video content
... so here I'm with nick and we've got look at my inventory and his inventory looks similar this is gonna be interesting so four...
mismatched video content
... give it I'm a little chicken just laying around by any chance a lot of arrows going on down here I'm guessing most of ...
matched video content
... at some point you will lose there too or also hearts there spider was like being completely blown away I don 't assume anything at ...

Figure 2: Illustration of the YouTube video database. The screenshots of video clips are on the left and key entities are circled in red. The corresponding transcript clips are on the right and key entities are marked in red. We give examples of irrelevant, mismatched, and matched video content in the YouTube video database.

[9] collects 730K+ narrated Minecraft videos from YouTube to learn a vision-language model (MineCLIP) providing auxiliary reward signals. VPT [2] uses action-labeled data to train an inverse dynamic model to label internet-scale data and then conduct behavior cloning.

Unlike these works (except MineCLIP) that incorporate the human experience and require a large number of demonstrations with action labels to train the agent, our work focuses on only using the data without action labels to assist agent learning, which is more friendly with data collection and has the potential to scale in the future.

3. Background

MineDojo tasks. MineDojo has provided thousands of benchmark tasks with its simulator APIs, which can be used to develop generally capable agents in Minecraft. Furthermore, the tasks can be divided into two categories, Programmatic and Creative tasks. The former has ground-truth simulator states to assess whether the task has been completed. The latter, however, do not have well-defined success criteria and tend to be more open-ended, but have to be evaluated by humans.

We mainly focus on Programmatic tasks since they can be automatically assessed. In addition, MineDojo provides 4 categories of programmatic tasks, including Harvest, Combat, Survival, and Tech Tree, with 1,581 template-generated natural language goals to evaluate the agent's different capabilities. Among these tasks, Survival and Tech Tree tasks are harder than Harvest and Combat. At the current stage, MineCLIP only expresses the promising potential in some Harvest and Combat tasks. Harvest means finding, obtaining, cultivating, or manufacturing hundreds of materials and objects. Combat means fighting various monsters and creatures that require fast reflexes and martial skills.

POMDP. We model the programmatic task as a partially observable Markov decision process (POMDP) [14]. At each timestep t, the agent obtains the partial observation o_t from the global state s_t and a language prompt G, takes action a_t following its policy $\pi(a_t|o_t)$, and receives a reward $r_t = \Phi(V_t, G)$, where V_t is the fixed-length sequence of observations till t (thus a video snippet) and Φ maps V_t and G to a scalar value. Then the environment transitions to the next state s_{t+1} given the current state and action according to transition probability function $\mathcal{T}(s_{t+1}|s_t, a_t)$. The agent aims to maximize the expected return $R = \mathbb{E}_\pi \sum_{t=1}^{T} \gamma^{t-1} r_t$, where γ is the discount factor and T is the episode time horizon.

4. YouTube Dataset

MineDojo provides an extensive suite of APIs to interact with the Minecraft environment. However, it is still an arduous challenge for agents to complete long-term tasks with only online experiences, owing to the sparsity of rewards and the vast exploration space. Fortunately, the internet is awash with copious amounts of Minecraft-related data, which harbors a wealth of weak-labeled or even unlabelled Minecraft knowledge, including crucial entities, plausible motions, and common-sense event processes. In MineDojo [9], a total of 640K video clips (8$\sim$16 seconds/clip) sourced from 730K+ YouTube videos and their corresponding transcripts are leveraged to train a vision-language model (MineCLIP) that provides auxiliary reward signals for agents to learn task-specific strategies efficiently. Due to the limitation of computing resources, we cannot fully use 730K+ YouTube videos, which makes a gap between video clips and tasks. Thus, the quality of the vision-language model is highly sensitive to the quality of video-text pairs used for training. Since the specific video-text clip pairs used in MineDojo have not been released, we construct a neat video-text dataset on the basis of the internet-scale database.

Our dataset construction is based on the YouTube video database collected by MineDojo, which comprises over 730K Minecraft videos with a combined duration of 33 years and 2.2B transcript words. The database is massive in scale and contains significant noise, due to its nature as an online resource. As illustrated in Figure 2, on one hand, some videos feature irrelevant game content that may not be conducive to learning basic game concepts. On the other hand, the alignment between the transcripts and videos may not always be precise, leading to temporal or content discrepancies that could hinder the learning of semantic events for the vision-language model. We aim to cover the essential information in the Minecraft environment, including key entities and basic semantic events, with a smaller amount of data. To achieve this, we employed a two-step filtering process to construct a neat video-text dataset.

Content Filtering. To begin with, we employed transcript-based filtering to ensure that the video content in our dataset is relevant to the key entities in Minecraft, as shown in Figure 1, thus facilitating the learning of basic game concepts. As the fundamental elements of Minecraft, the key entities, *e.g.*, stones, trees, and sheep, are shared between tasks in MineDojo and videos from YouTubers. Specifically, we identify entity keywords in the transcripts with a handmade keyword list and extract transcript clips of a fixed context length L to encompass as many keywords as possible. The extracted transcript clips serve as the text part of our dataset and determine the location of corresponding video clips.

Correlation Filtering. Secondly, we aim to find video clips that are highly correlated with the extracted transcript clips to ensure the alignment between the text and video modalities. As the reasonable start and end timestamps of the video clips do not necessarily coincide with the corresponding transcripts clips, we adopt a method of aligning the center points of the transcript and video clips for possible semantic overlap content and we extract video clips of a fixed duration D. However, even with this approach, we cannot guarantee consistency between the video content and the transcript content. Therefore, we utilize the pre-trained MineCLIP model to obtain the embeddings of selected video clips and transcript clips. Then, we calculate the cosine similarity distribution of their embeddings to represent their correlation. In more detail, we perform a second round of filtering, selecting from the top $k\%$ of video clips whose embedding is closest to the embedding of the corresponding transcript clips. We construct a dataset with a stronger content correlation between the video and transcript clips, as shown in Figure 3.

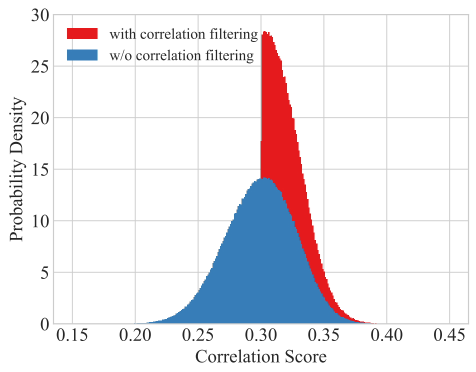


Figure 3: Distribution of correlation score of video-text pairs with and without correlation filtering in YouTube Dataset.

In terms of scale, we apply the aforementioned two-step approach to construct a training set of 640K video-text clip pairs and extract additional 4K pairs for validation of video-text retrieval. For the constants of the approach, we set L, D, and k to 25 words, 16 seconds and 50, respectively. This means that our dataset contains only 1 week's total duration of videos and 0.16B words, significantly smaller than the scale of the original database. *Importantly, we will release our YouTube dataset by specifying transcript clips and the corresponding timestamps of the videos in the original database.*

5. CLIP4MC

For the RL training procedure, we expect the learned vision-language model can provide a high-quality reward signal without any domain adaptation techniques. As it eliminates the need to manually engineer the reward function for each MineDojo task, the agent can continuously learn from open-ended tasks.

Given a video snippet V and a language prompt G, the vision-language model outputs a correlation score, C, that measures the similarity between the video snippet and the language prompt. Ideally, if the agent performs behaviors following the description of the language prompt, the vision-language model will generate a higher correlation score,

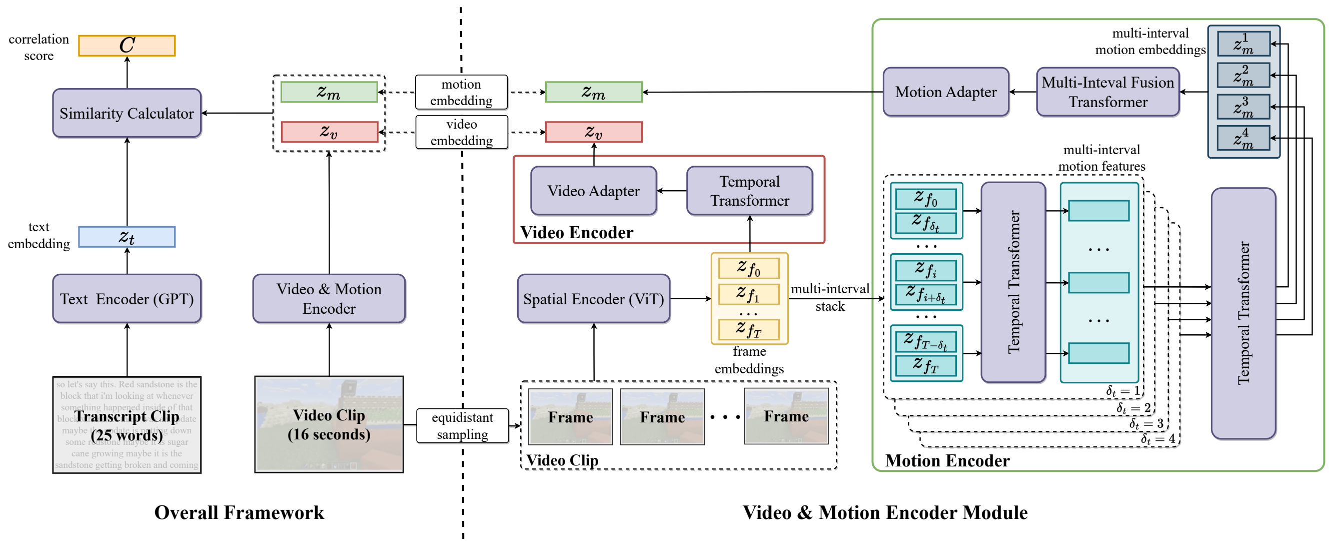


Figure 4: Overall architecture of CLIP4MC. The model consists of three encoders, *i.e.*, video, motion, and text encoders. Note that video and motion encoders aim to extract the embeddings from the video snippet at the entity and action levels, respectively. Correlation scores are used to construct the auxiliary reward signal for RL training.

leading to a higher reward. Otherwise, the agent will be given a lower reward.

5.1. Overall Architecture

Intuitively, only when the agent makes the right *behavior* toward the right *entity*, the agent shall be given a higher reward. CLIP4MC follows such a principle. Figure 4 depicts the overall architecture of CLIP4MC, which consists of a video encoder, a motion encoder, a text encoder, and a similarity calculator. We have two encoders to extract two different levels of video embeddings, *i.e.*, entity level and motion level.

Video Encoder. To generate entity-level video embedding, z_v, we follow the same design as MineCLIP [9]. All the video frames first go through the spatial transformer to obtain a sequence of frame features. The temporal transformer is then utilized to summarize the sequence of frame features into a single video embedding. CLIP Adapter further processes the video embedding for better features. Note that we initialize the weights of the spatial transformer from the checkpoint of CLIP [22] and only the last two layers are finetuned during training. Empirically, we found out the video encoder (essentially MineCLIP) can provide a bond between the entities and the language prompts and give a similar high reward as long as the target entities are in the video frames. However, such a reward is not instructive enough for RL tasks.

Motion Encoder. Further, we explicitly pilot the temporal transformer to encode motion embeddings. Normally, two consecutive or spaced frames contain entity displacement, which implies the actual actions with different amplitudes. Motivated by this, we stack two frame features with different intervals as action representations. Note that the frame features are reused from the video encoder. The temporal transformer then processes all representations to obtain the motion features separately.

For the motion features from the same interval, we again summarize them via the temporal transformer and obtain one motion embedding of this interval. The multi-interval motion embeddings, $z_m^{1:j}$, capture richer information about actual actions. The final step is to compress the multi-interval motion embeddings into one embedding. Empirically, we found out that max or average pooling operation cannot summarize the final motion embedding well. Therefore, another fusion transformer is adopted to generate the final motion embedding, z_m. Importantly, the temporal transformers are shared across different encoders, *i.e.*, video and motion encoders, for fast convergence.

Text Encoder. We also directly utilize the architecture from the CLIP to extract the text embedding, z_t. We initialized the weights from the checkpoint of CLIP and only the last two layers of the text encoder are finetuned during training.

Similarity Calculator. After extracting the embeddings of input from different modalities, the final stage comes to the similarity calculation. The similarity function $s(V, G)$ that measures the correlation score, C, between the video snippet and the language prompt is defined as the cosine similarity:

$$s(V,G) = \frac{z_v^\top z_t}{2\|z_v\|\|z_t\|} + \frac{z_m^\top z_t}{2\|z_m\|\|z_t\|}.$$

5.2. Training

Contrastive Learning. For CLIP4MC training, we use a contrastive loss [21] to learn the correspondence between video snippet and language prompt. In more detail, we aim to minimize the sum of the multi-modal contrastive losses, including video-to-text, text-to-video, motion-to-text, and text-to-motion:

$$\mathcal{L} = -\sum_{(z_v, z_m, z_t)\in\mathcal{B}} \Big(\log \mathrm{NCE}(z_v, z_t) + \log \mathrm{NCE}(z_t, z_v) + \lambda \log \mathrm{NCE}(z_m, z_t) + \lambda \log \mathrm{NCE}(z_t, z_m) \Big),$$

where $\mathcal{B}$ is the batch containing sampled video-text pairs, λ is the weight coefficient, and $\mathrm{NCE}(\cdot,\cdot)$ is the contrastive loss that calculates the similarity of two inputs. To illustrate, the video-to-text contrastive loss is given by

$$\mathrm{NCE}(z_v, z_t) = \frac{\exp(z_v \cdot z_t^+/\tau)}{\sum_{z\in\{z_t^+, z_t^-\}} \exp(z_v \cdot z/\tau)},$$

where τ is a temperature hyperparameter, z_t^+ is the positive text embedding matching with the video embedding z_v, and $\{z_t^-\}$ are negative text embeddings that are implicitly formed by other text clips in the training batch. Other contrastive losses, *i.e.*, text-to-video, motion-to-text, and text-to-motion, are defined in the same way.

RL Training. For RL training, the first step is reward generation. At timestep t, we concatenate the agent's latest 16 egocentric RGB frames in a temporal window to form a video snippet, V_t. CLIP4MC outputs the probability $P_{G,t}$ that calculates the similarity of V_t to the task prompt, G, against all other negative prompts. To compute the reward, we further process the raw probability as previous work [9] $r_t = \max(P_{G,t} - \frac{1}{N_t}, 0)$, where N_t is the number of prompts passed to CLIP4MC. Note that CLIP4MC can handle unseen language prompts without any further finetuning.

The ultimate goal is to train a policy network that takes as input raw pixels and other structural data and outputs discrete actions to accomplish the task that is described by the language prompt, G. We use Proximal Policy Optimization (PPO) [24] as our RL training backbone and the policy is trained on the CLIP4MC reward together with the sparse task-completion reward if any. The policy input contains several modality-specific components and more details can be found in Appendix. In addition, self-imitation learning (SI) [20] is used to further improve the sample efficiency as MineCLIP [9], because computing the reward using a vision-language model in the loop makes the training more expensive. The training phases alternate between the PPO phase and the SI phase.

In the next section, we will show that CLIP4MC can provide a more reliable reward signal that accelerates the learning of RL tasks compared with other methods. Importantly, a model that can achieve better performance on video-text retrieval metrics, *e.g.*, R@1, is not necessarily able to provide RL-friendly reward signals.

6. Experiments

In this section, we conduct a comprehensive evaluation and analysis of our proposed model, CLIP4MC, utilizing the open-ended platform, MineDojo. This platform features a benchmark suite comprised of thousands of diverse, open-ended Minecraft tasks designed for embodied agents.

We compare CLIP4MC against the following baselines:

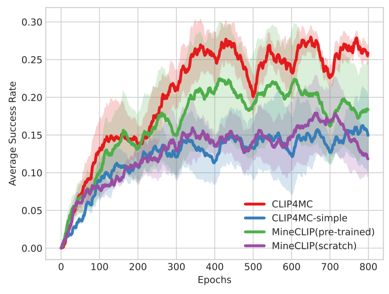
1. **MineCLIP(pre-trained)** refers to the officially released MineCLIP model by MineDojo [9]. It is a variant of the video-text retrieval method, CLIP4CLIP [18].
2. **MineCLIP(scratch)** uses the same architecture as MineCLIP(pre-trained) but is trained from scratch on our YouTube dataset. It also serves as the ablation of CLIP4MC without the motion encoder.
3. **CLIP4MC-single** is another ablation of CLIP4MC, where the motion encoder only processes the motion representations with one interval.

Note that CLIP4MC, MineCLIP(scratch), and CLIP4MC-single are trained on our YouTube dataset. Specifically, we trained these models for 20 epochs and select the models with the highest performance on RL tasks. Please refer to Appendix for more training details. All results are presented in terms of the mean and standard deviation of five runs with different random seeds.

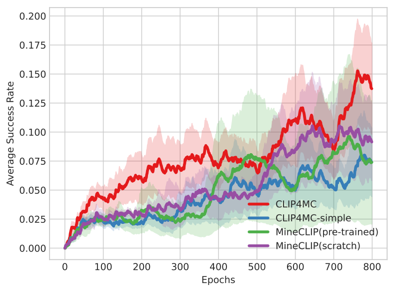
6.1. Environment Settings

As mentioned in Section 3, there are four categories of Programmatic tasks, *i.e.* Harvest, Combat, Survival, and Tech Tree. However, the goals of Survival and Tech Tree tasks are more complex and the corresponding language prompts are too abstract. Thus, it is hard for the agent to achieve the final goal without additional subgoal decomposition and planning. Surprisingly, all the methods, including MineCLIP(pre-trained), also fail in Combat tasks (nearly zero success rate). Similar results are also found in recent work [5]. This may be due to different experimental settings, such as episode length. A thorough investigation of Combat tasks is left as future work.

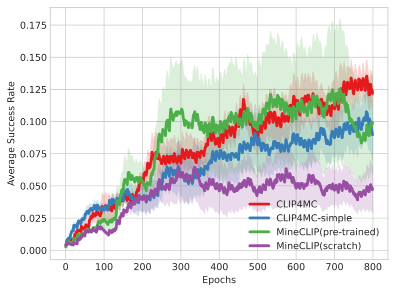
To comprehensively evaluate our proposed model, we conduct experiments on six different RL tasks featuring diverse domain-specific entities (*e.g.*, animals, resources, terrains, and tools). These tasks are either built-in tasks in the MineDojo benchmark suite, or designed by ourselves. Based on their semantic definitions, we group these six RL tasks into two categories: Harvest and Finding. To guarantee



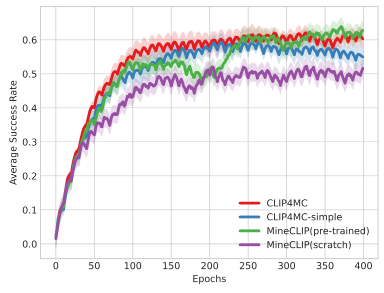
(a) Milk a Cow



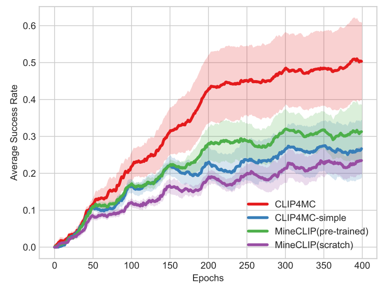
(b) Sheer Wool



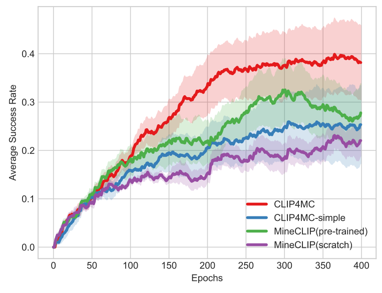
(c) Pick a Flower



(d) Pick a Leaf



(e) Find a Cow



(f) Find a Sheep

Figure 5: Learning curves of CLIP4MC and baselines in six Programmatic tasks of MineDojo. Note that CLIP4MC, CLIP4MC-single, and MineCLIP(scratch) are trained on our YouTube dataset. All results are presented in terms of the mean and standard deviation of five runs with different random seeds.

a fair comparison, we adopt the most hyperparameters of RL training from MineCLIP [9] for all tasks and all methods. We only adjust the batch and buffer size to fit our computation resource.

Harvest Task. There are four harvest tasks in total, each requiring the agent to obtain a specific item: one sunflower , one leaf , one bucket of milk , and one wool . The task to harvest one sunflower was designed by ourselves, and the other three are built-in tasks in MineDojo. To successfully complete these tasks, the agent is initialized in various biomes with specific tools that are necessary to obtain the target item in Minecraft. A harvest task is regarded as successful when the target item is obtained by the agent with a specified quantity. In addition, a language prompt for the task of harvesting milk is, "*obtain milk from a cow using an empty bucket*."

Finding Task. We also include two finding tasks, in which the agent is tasked with locating a specific target object, such as a cow or a sheep . These finding tasks are not from the MineDojo benchmark suite but are specially designed to assess the agent's ability to efficiently explore the environment. In Minecraft, each block is 1x1 square. To initialize our experiment, we randomly spawn a target object in a 60x60 square area, while the agent is positioned at the center of the area. A task is considered successful when two conditions are met: the target object is within the agent's field of view, and the Euclidean distance between the agent and the target object must be less than 3. We use the built-in lidar function to detect whether the target object is within the agent's field of view. A natural language prompt for the task of finding cow could be "*find a cow in the plain and the cow is nearby*." More details about the environment settings can be found in Appendix.

6.2. Results

RL Results. Figure 5 shows the learning curves of all the methods regarding the success rate. CLIP4MC achieves better performance than MineCLIP(pre-trained) in all the tasks except for picking a leaf where they perform comparably. It is also shown that the RL training based on MineCLIP(pre-trained) is highly unstable. Note that these two models are trained on two different datasets. The comparison may not be fair since the difference in the key entity distribution of the dataset can affect the models' performance on different tasks. In addition, the difference in training details could also impact the final results.

Moreover, CLIP4MC outperforms MineCLIP(scratch) and CLIP4MC-simple by converging to the highest success

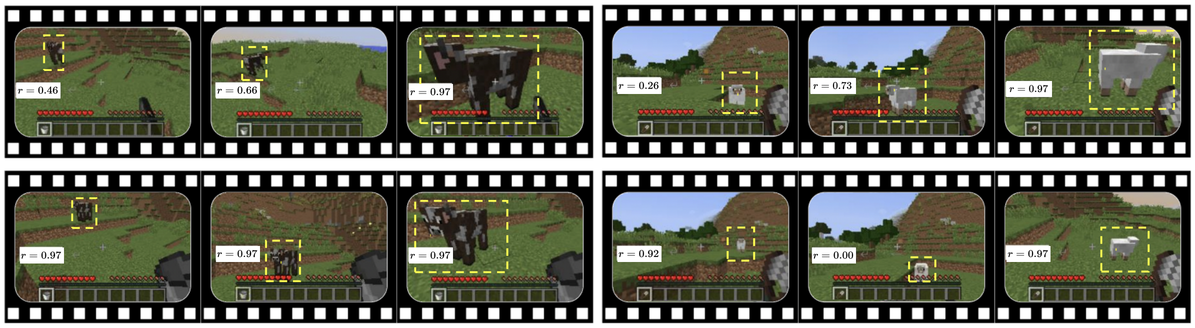


Figure 6: Illustration of learned reward function of CLIP4MC (*upper panel*) and MineCLIP(pre-trained) (*lower panel*). For each row, the first three screenshots are from Milk a Cow and the last three screenshots are from Sheer Wool. The key entities are boxed in yellow and the reward signal, r, is shown in the white box.

rate in all six RL tasks. It demonstrates the benefits brought by the motion encoder. In more detail, it turns out that the usage of MineCLIP does not fully exploit the temporal relationship of video snippets. It is hard to learn motion features with a temporal transformer by only relying on video embedding. The improvement of CLIP4MC over CLIP4MC-simple is attributed to CLIP4MC's ability to capture more extensive and detailed motion features with different amplitudes. Intuitively, multi-interval action representations contain temporal correlations of short-term and long-term segments. Therefore, the motion encoder is more likely to obtain useful features that match the behaviors reflected in the language prompts.

Video-Text Retrieval Results. Table 1 shows the results of text-to-video retrieval on the test set. Note that the $*$ symbol means the model is trained on the YouTube dataset without correlation filtering. From the results, the models trained on the dataset without correlation filtering obtain worse performance than those trained on our YouTube dataset. In addition, they also fail in most RL tasks. The results demonstrate the superiority and necessity of our proposed YouTube dataset.

Moreover, we found a model that can achieve better performance on video-text retrieval metrics is not necessarily able to provide RL-friendly reward signals. In more detail, CLIP4MC achieves the best performance on RL tasks, but CLIP4MC-simple obtained the best R@1 value. Through the model analysis in Section 6.3, we provide one of the possible answers.

6.3. Model Analysis

In this section, we conduct model analysis by extracting a series of screenshots in an episode. Figure 6 shows the learned reward function of CLIP4MC and MineCLIP(pre-trained) in two tasks. For MinCLIP(pre-trained), we can find that it provides a tight bond between the entities and the language prompts. However, if the agent can get a higher reward far away from the entity, it prefers to choose to stay, rather than approach the entity. Therefore, further exploration is relatively hard and this may explain its unstable learning procedure of RL tasks.

On the contrary, though not always, CLIP4MC can learn a reward signal that gradually increases as the agent approaches the entity. Under such a reward function, agents are more likely to explore the right actions moving toward the targets. Note that we can observe a similar phenomenon in both tasks.

Table 1: Results of video-to-text retrieval on the test set. We train these models for 20 epochs and select the models with the highest R@1 value on the test set, respectively. The best results are shown in bold.

Methods	R@1 $\uparrow$	R@5 $\uparrow$	R@10 $\uparrow$	MdR $\downarrow$	MnR $\downarrow$
CLIP4MC	11.9	25.1	33.1	40.0	**270.3**
CLIP4MC-simple	**12.3**	25.1	32.5	41.0	281.3
MineCLIP(scratch)	12.2	**26.0**	**33.4**	**38.0**	278.1
CLIP4MC*	10.7	23.1	30.5	46.0	281.8
CLIP4MC-simple*	10.5	22.8	29.8	49.0	291.6
MineCLIP(scratch)*	10.4	21.4	28.8	49.0	301.5

7. Conclusions

In this paper, we first bring up three key issues we need to solve for learning a vision-language model as the universal reward function for open-ended RL tasks. To address the issue of whether the agent has made the right action toward the right target, we propose an RL-friendly vision-language model. It can align actions implicitly contained in the video and transcript clips in addition to entities. Moreover, we construct and release a neat vision-language dataset for Minecraft based on YouTube videos from MineDojo. Empirically, our method can provide a more friendly reward signal for the RL training procedure.

References

[1] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[2] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (VPT): learning to act by watching unlabeled online videos. *arXiv preprint arXiv:2206.11795*, 2022.

[3] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[4] Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. *arXiv preprint arXiv:2301.10034*, 2023.

[5] Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. *arXiv preprint arXiv:2301.10034*, 2023.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[8] Maksim Dzabraev, Maksim Kalashnikov, Stepan Komkov, and Aleksandr Petiushko. MDMMT: multidomain multimodal transformer for video retrieval. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.

[9] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2022.

[10] William H Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, et al. Neurips 2019 competition: the minerl competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv:1904.10079*, 2019.

[11] William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[12] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[13] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[14] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[15] Anssi Kanervisto, Stephanie Milani, Karolis Ramanauskas, Nicholay Topin, Zichuan Lin, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, Wei Yang, Weijun Hong, Zhongyue Huang, Haicheng Chen, Guangjun Zeng, Yue Lin, Vincent Micheli, Eloi Alonso, François Fleuret, Alexander Nikulin, Yury Belousov, Oleg Svidchenko, and Aleksei Shpilman. Minerl diamond 2021 competition: Overview, results, and lessons learned. *arXiv preprint arXiv:2202.10583*, 2022.

[16] Zichuan Lin, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, and Wei Yang. Juewu-mc: Playing minecraft with sample-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:2112.04907*, 2021.

[17] Song Liu, Haoqi Fan, Shengsheng Qian, Yiru Chen, Wenkui Ding, and Zhongyuan Wang. Hit: Hierarchical transformer with momentum contrast for video-text retrieval. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[18] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. Clip4clip: An empirical study of CLIP for end to end video clip retrieval and captioning. *Neurocomputing*, 508:293–304, 2022.

[19] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Neural Information Processing Systems (NeurIPS)*, 2015.

[20] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning (ICML)*. PMLR, 2018.

[21] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning, ICML*, 2021.

[23] 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. *arXiv preprint arXiv:1902.04043*, 2019.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[25] Rohin Shah, Steven H. Wang, Cody Wild, Stephanie Milani, Anssi Kanervisto, Vinicius G. Goecks, Nicholas R. Waytowich, David Watkins-Valls, Bharat Prakash, Edmund Mills, Divyansh Garg, Alexander Fries, Alexandra Souly, Jun Shern Chan, Daniel del Castillo, and Tom Lieberum. Retrospective on the 2021 minerl BASALT competition on learning from human feedback. In *Neural Information Processing Systems (NeurIPS) Competitions and Demonstrations Track*, 2021.

[26] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[27] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[28] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI conference on artificial intelligence (AAAI)*, 2017.

[29] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023.

A. Environment Details

Environment Initialization. Table 1 outlines how we set up and initialize the environment for each harvest task. Tabel 2 outlines the environment configuration for each finding task.

Table 1: Environment Setup for Harvest Tasks

Harvest Item	Initialized Tool	Biome
milk	empty bucket	plains
wool	shears	plains
leaf	shears	jungle
sunflower	diamond shovel	sunflower plains

Table 2: Environment Setup for Finding Tasks

Target Item	Initialized Tool	Biome
cow	barehand	plains
sheep	barehand	plains

Biomes. As described in the task description, our model was tested in a simulation environment featuring three distinct terrains: jungle, plains, and sunflower plains. Each terrain presents unique challenges for our RL tasks. Specifically, while the plains offer a wider field of view, resources and targets are situated further away from the agent. Conversely, the jungle terrain features resources and targets that are closer to the agent, but the terrain is bumpier and presents additional obstacles. Figure 1 shows the biomes of plains, sunflower plains, and jungle in Minecraft, respectively.

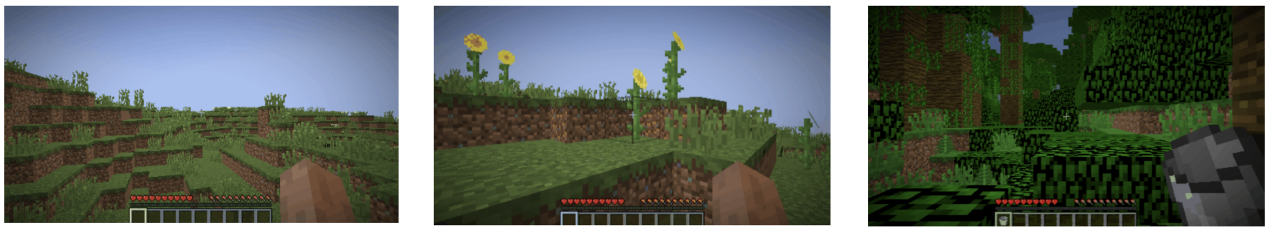


Figure 1: Biomes of plains, sunflower plains, and jungle (from left to right).

Observation Space. To enable the creation of multi-task and continually learning agents that can adapt to new scenarios and tasks, MineDojo provides unified observation and action spaces. The observation space mainly includes 9 parts. Table 3 provides detailed descriptions of the observation space. For more details, please refer to MineDojo's [9] observation space.

Action Space. The action space is an 8-dimensional multi-discrete space, including moving actions (forward, backward, camera actions, etc.) and functional actions (attack, use, craft, etc.). At each step, the agent chooses one movement action and one optional functional action. Table 4 summarizes the action space in the Minecraft environment. For more details, please refer to MineDojo's [9] action space.

Table 3: Observation Space of MineDojo Environment

Observation	Descriptions
Egocentric RGB fram	RGB frames provide an egocentric view of the running Minecraft client; The shape height (H) and width (W) are specified by argument image_size. In our experiment, it is (160, 256).
Equipment	Equipment observation includes names, quantities, and durability of agent's equipment. They are flattened with the order of "main hand, foot, leg, body, head, off hand".
Inventory	Inventory observation includes names, quantities, and durability of items in agent's inventory. There are 36 slots in the inventory, including 10 hotbar slots and 26 inventory slots.
Inventory Change	The features on inventory change can help the agent associate the inventory with its activities.
Voxels	Voxels observation refers to the 3x3x3 surrounding blocks around the agent. This type of observation is similar to how human players perceive their surrounding blocks. It includes names and properties of blocks.
Life Statistics	Life statistics include agent's health, armor, food saturation, etc. It can be regarded as a vector representation of the heads-up display
Location Statistics	Location statistics include information about the terrain the agent currently occupies. It also includes agent's location and compass.
Nearby Tools	This observation indicates if a crafting table or a furnace are nearby. Both are important tools for crafting/smelting new items.
Damage Source	Damage source tells information about damage taken by the agent. It includes the damage amount and properties of damage.

Table 4: Action Space of MineDojo Environment

Index	Descriptions	Num of Actions
0	Forward and backward	3
1	Move left and right	3
2	Jump, sneak, and sprint	4
3	Camera delta pitch	25
4	Camera delta yaw	25
5	Functional actions	8
6	Argument for "craft"	244
7	Argument for "equip", "place", and "destroy"	36

B. RL Training Details

During the RL training stage, we adopt an algorithm pipeline similar to MineDojo [9]. We apply both Proximal Policy Optimization (PPO) algorithm and Self-Imitation Learning (SIL) [20] by storing the trajectories with high clip reward values in a buffer. We then alternated between PPO and SIL gradient steps during the training process. The hybrid method enables us to leverage their respective strengths and achieve better results than using either method alone. The training hyper-parameters for the RL task are listed in Table 5.

Table 5: Training hyper-parameters for RL&IL.

Hyperparameter	**Value**
Optimizer	Adam
LR	1e-4
RL discount factor	0.99
Lambda for GAE	0.95
Steps per episode	200
Buffer size for IL	500
Imitate frequency	per 100 epochs
Number of workers (CPU)	1
Parallel GPUs	1
MineDojo image size	160×256
Clip ratio	0.2
Early stopping KL	0.01

C. Details on Dataset

In this section, we provide more details about the dataset, including the construction process and a sample of the dataset format. The construction steps of the dataset are as follows:

1. Obtain YouTube videos and corresponding transcripts from the MineDojo database.

2. Manually construct a list of keywords related to Minecraft gameplay.

3. For each video with a transcript, annotate all keywords (including different forms of keywords such as combined words, plural forms, etc.) appearing in the transcript.

4. Slide a window of length L words on the transcript until the first keyword in the window is about to leave the window. Use the midpoint between the first and last keyword in the window as the center to extract a transcript clip of length L words.

5. Extract all non-overlapping transcript clips from each video with a transcript following step 4.

6. For each transcript clip, calculate the central timestamp corresponding to the clip based on the transcript timestamps. Use this central timestamp to extract a video clip of duration D seconds from the video.

7. From all the video-clip pairs extracted in the previous steps, extract M pairs and encode these pairs using a pre-trained MineCLIP attention variant to calculate the cosine similarity.

8. Select the top $k\%$ pairs with the highest cosine similarity as the training set from the M pairs.

9. Randomly select M' pairs in addition to the M pairs as the validation set.

The values of the parameters used in the above process are listed in Table 6. Specifically, steps 2-6 of the process constitute content filtering, while steps 7-8 are correlation filtering. Following this process, we construct a training set of size 640K and a test set of size 4096. Both of the filtering methods have been applied to construct the training set, while only content filtering is used for the test set to reflect the distribution of data in the database. we release our YouTube dataset by specifying the transcript clips and the corresponding timestamps of the videos in the original database. The link is https://drive.google.com/drive/folders/19vDy2jaooF74MDt3dLAsyLRpRcUFKVCY?usp=sharing.

D. CLIP4MC Training

This section describes the training process of CLIP4MC. The training process for CLIP4MC was adapted from the training processes for CLIP4CLIP and MineCLIP. Practically, we trained all models on the 640K training set. For each video-text clip pair, we obtain 16 frames of RGB image through equidistant sampling and normalize each channel separately. During

Table 6: Values of parameters used in the dataset construction process.

Parameter	Value
L	25
D	16
M	1,280,000
k	50
M'	4,096

training, we use random resize crops for data augmentation. We use cosine learning rate annealing with 320 gradient steps of warming up. We only fine-tune the last two layers of pre-trained CLIP encoders, and we apply a module-wise learning rate decay (learning rate decays along with the modules) for better fine-tuning. Training is performed on 1 node of 4 × V100 GPUs with FP16 mixed precision via the PyTorch native `amp` module. All hyperparameters are listed in Table 7.

Table 7: Training hyperparameters for CLIP4MC.

Hyperparameter	Value
LR schedule	Cosine with warmup
Warmup steps	320
LR	1.5e-4
Weight decay	0.2
Layerwise LR decay	0.65
Batch size per GPU	100
Parallel GPUs	4
Video resolution	160 × 256
Number of frames	16
Image encoder	ViT-B/16

E. CLIP4MC Architecture

Input. The length of each transcript clip is 25 words, while the length of the video is 16 seconds. The resolution of the video stream is 160 × 256, with 5 fps. In other words, the video stream is 80 frames. As for the video snippet, we further equidistantly sample it to 16 frames for fewer computing resources.

Text Encoder. Referring to the design of MineCLIP [9], the text encoder is a 12-layer 512-width GPT model, which has 8 attention heads. The textual input is tokenized via the tokenizer used in CLIP and is padded/truncated to 77 tokens. The initial weights of the model use the public checkpoint of CLIP and only finetune the last two layers during training.

Spatial Encoder. The Spatial encoder is a frame-wise image encoder referred to the design of MineCLIP [9], which uses the ViT-B/16 architecture to compute a 512-D embedding for each frame. The initial weights of the model use the public checkpoint of OpenAI CLIP, and only the last two layers are finetuned during training.

Temporal Transformer. The temporal Transformer is a shared module across video and motion encoders for fast convergence, which is a 2-depth 8-head Transformer module whose input dimension is 512 and maximum sequence length is 32.

Multi-interval Fusion Transformer. Multi-interval fusion transformer is a Transformer model similar to Temporal Transformer, which fuses motion information of different intervals. The architecture is exactly the same as the Temporal Transformer, except that the parameters are not shared.

Adapter. In order to get better embedding, we use an adapter to map video embedding, text embedding, and motion embedding. The adapter models are 2-layer MLP, except the text adapter which is an identity.

F. Agent Architecture

Like MineDojo [9], our policy framework is also composed of three components: an encoder for input features, a policy head, and a value function head. In order to deal with cross-modal observations, the feature extractor includes a variety of modality-specific components as described in Table 8.

- RGB frame: To optimize for computational efficiency and equip the agent with strong visual representations from scratch, we use the fixed frame-wise image encoder from CLIP4MC to process RGB frames.
- Yaw and Pitch: We first compute sin(·) and cos(·) features respectively, then pass them through CompassMLP.
- GPS: normalize and pass through GPSMLP.
- Voxel: To process the $3 \times 3 \times 3$ voxels surrounding the agent, we embed discrete block names as dense vectors, flatten them, and pass them through VoxelEncoder.
- Previous action: Our agent relies on its immediate previous action, which is embedded and processed through PrevActionEmb, which is a conditioning factor.
- BiomeID: To perceive the discrepancy in different environments, we embed BiomeID as a vector through an MLP named BiomeIDEmb.

The features from each modality are combined by concatenation, passed through an additional feature fusion network (FeatureFusion), and then directed into both the policy head and value function head. The policy head is modeled using an MLP (PolicyMLP), which transforms the input feature vectors into an action probability distribution. Similarly, ValueMLP is used to estimate the value function, conditioned on the same input features. Note that we do not use all the information of the observation space. Only partial observation information will be sent into the policy network.

Table 8: Agent Networks

Network	Details
Policy&ValueMLP	hidden_dim: 256 hidden_depth: 3
CompassMLP	hidden_dim: 128 output_dim: 128 hidden_depth: 2
GPSMLP	hidden_dim: 128 output_dim: 128 hidden_depth: 2
VoxelEncoder	embed_dim: 8 hidden_dim: 128 output_dim: 128 hidden_depth: 2
BiomeIDEmb	embed_dim: 8
PrevActionEmb	embed_dim: 8
FeatureFusion	output_dim: 512 hidden_depth: 0