# MINEDOJO: Building Open-Ended Embodied Agents with Internet-Scale Knowledge

**Linxi Fan**[1], **Guanzhi Wang**[2*], **Yunfan Jiang**[3*], **Ajay Mandlekar**[1], **Yuncong Yang**[4], **Haoyi Zhu**[5], **Andrew Tang**[4], **De-An Huang**[1], **Yuke Zhu**[1 6†], **Anima Anandkumar**[1 2†]
[1]NVIDIA, [2]Caltech, [3]Stanford, [4]Columbia, [5]SJTU, [6]UT Austin
[*]Equal contribution [†]Equal advising
https://minedojo.org

## Abstract

Autonomous agents have made great strides in specialist domains like Atari games and Go. However, they typically learn *tabula rasa* in isolated environments with limited and manually conceived objectives, thus failing to generalize across a wide spectrum of tasks and capabilities. Inspired by how humans continually learn and adapt in the open world, we advocate a trinity of ingredients for building generalist agents: 1) an environment that supports a multitude of tasks and goals, 2) a large-scale database of multimodal knowledge, and 3) a flexible and scalable agent architecture. We introduce MINEDOJO, a new framework built on the popular *Minecraft* game that features a simulation suite with thousands of diverse open-ended tasks and an internet-scale knowledge base with Minecraft videos, tutorials, wiki pages, and forum discussions. Using MINEDOJO's data, we propose a novel agent learning algorithm that leverages large pre-trained video-language models as a learned reward function. Our agent is able to solve a variety of open-ended tasks specified in free-form language without any manually designed dense shaping reward. We open-source the simulation suite, knowledge bases, algorithm implementation, and pretrained models (https://minedojo.org) to promote research towards the goal of generally capable embodied agents.

## 1 Introduction

Developing autonomous embodied agents that can attain human-level performance across a wide spectrum of tasks has been a long-standing goal for AI research. There has been impressive progress towards this goal, most notably in games [80, 85, 126] and robotics [68, 99, 146, 134, 107]. These embodied agents are typically trained *tabula rasa* in isolated worlds with limited complexity and diversity. Although highly performant, they are specialist models that do not generalize beyond a narrow set of tasks. In contrast, humans inhabit an infinitely rich reality, continuously learn from and adapt to a wide variety of open-ended tasks, and are able to leverage large amount of prior knowledge from their own experiences as well as others.

We argue that **three main pillars** are necessary for generalist embodied agents to emerge. First, the environment in which the agent acts needs to **enable an unlimited variety of open-ended goals** [116, 71, 120, 117]. Natural evolution is able to nurture an ever-expanding tree of diverse life forms thanks to the infinitely varied ecological settings that the Earth supports [117, 129]. This process has not stagnated for billions of years. In contrast, today's agent training algorithms cease to make new progress after convergence in narrow environments [80, 146]. Second, a **large-scale database of prior knowledge** is necessary to facilitate learning in open-ended settings. Just as humans frequently learn from the internet, agents should also be able to harvest practical knowledge encoded in large amounts of video demos [42, 77], multimedia tutorials [79], and forum discussions [127, 65, 54]. In a

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

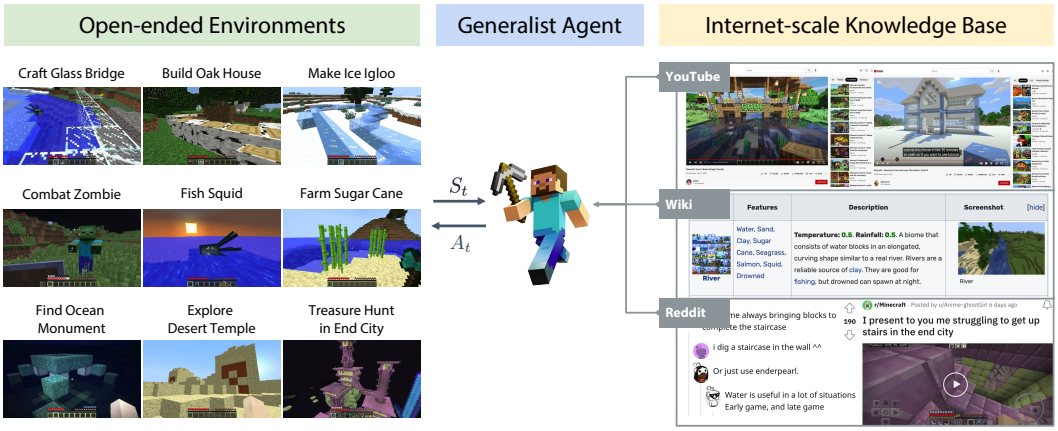



Figure 1: MINEDOJO is a novel framework for developing open-ended, generally capable agents that can learn and adapt continually to new goals. MINEDOJO features a benchmarking suite with ***thousands* of diverse open-ended tasks** specified in natural language prompts, and also provides an **internet-scale, multimodal knowledge base** of YouTube videos, Wiki pages, and Reddit posts. The database captures the collective experience and wisdom of millions of Minecraft gamers for an AI agent to learn from. Best viewed zoomed in.

complex world, it would be extremely inefficient for an agent to learn everything from scratch through trial and error. Third, the **agent's architecture** needs to be flexible enough to pursue any task in open-ended environments, and scalable enough to convert large-scale knowledge sources into actionable insights [19, 96]. This motivates the design of an agent that has a unified observation/action space, conditions on natural language task prompts, and adopts the Transformer pre-training paradigm [27, 91, 15] to internalize knowledge effectively.

In light of these three pillars, we introduce MINEDOJO, a new framework to help the community develop open-ended, generally-capable agents. It is built on the popular Minecraft game, where a player explores a procedurally generated 3D world with diverse types of terrains to roam, materials to mine, tools to craft, structures to build, and wonders to discover. Unlike most other games [80, 85, 126], Minecraft defines no specific reward to maximize and no fixed storyline to follow, making it well suited for developing open-ended environments for embodied AI research. We make the following three major contributions:

**1. Simulation platform with thousands of diverse open-ended tasks.** MINEDOJO provides convenient APIs on top of Minecraft that standardize task specification, world settings, and agent's observation/action spaces. We introduce a benchmark suite that consists of thousands of natural language-prompted tasks, making it *two orders of magnitude* larger than prior Minecraft benchmarks like the MineRL Challenge [48, 62]. The suite includes long-horizon, open-ended tasks that cannot be easily evaluated through automated procedures, such as "*build an epic modern house with two floors and a swimming pool*". Inspired by the Inception score [98] and FID score [55] that are commonly used to assess AI-generated image quality, we introduce a novel agent evaluation protocol using a large video-language model pre-trained on Minecraft YouTube videos. This complements human scoring [104] that is precise but more expensive. Our learned evaluation metric has good agreement with human judgment in a subset of the full task suite considered in the experiments.

**2. Internet-scale multimodal Minecraft knowledge base.** Minecraft has more than 100 million active players [131], who have collectively generated an enormous wealth of data. They record tutorial videos, stream live play sessions, compile recipes, and discuss tips and tricks on forums. MINEDOJO features a massive collection of 730K+ YouTube videos with time-aligned transcripts, 6K+ free-form Wiki pages, and 340K+ Reddit posts with multimedia contents (Fig. 3). We hope that this enormous knowledge base can help the agent acquire diverse skills, develop complex strategies, discover interesting objectives, and learn actionable representations automatically.

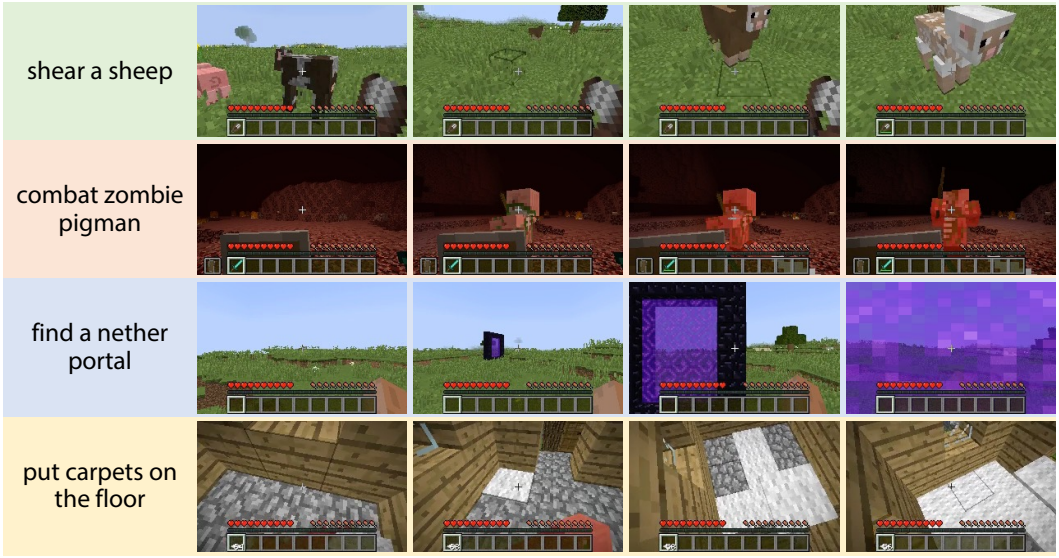



Figure 2: Visualization of our agent's learned behaviors on four selected tasks. Leftmost texts are the task prompts used in training. Best viewed on a color display.

**3. Novel algorithm for embodied agents with large-scale pre-training.** We develop a new learning algorithm for embodied agents that makes use of the internet-scale domain knowledge we have collected from the web. Using the massive volume of YouTube videos from MINEDOJO, we train a video-text contrastive model in the spirit of CLIP [92], which associates natural language subtitles with their time-aligned video segments. We demonstrate that this learned correlation score can be used effectively as an *open-vocabulary, massively multi-task reward function* for RL training. Our agent solves the majority of 12 tasks in our experiment using the learned reward model (Fig. 2). It achieves competitive performance to agents trained with meticulously engineered dense-shaping rewards, and in some cases outperforms them, with up to 73% improvement in success rates. For open-ended tasks that do not have a simple success criterion, our agents also perform well without any special modifications.

In summary, this paper proposes an open-ended task suite, internet-scale domain knowledge, and agent learning with recent advances on large pre-trained models [13]. We have open-sourced MINEDOJO's simulator, knowledge bases, algorithm implementations, pretrained model checkpoints, and task curation tools at `https://minedojo.org/`. We hope that MINEDOJO will serve as an effective starter framework for the community to develop new algorithms and advance towards generally capable embodied agents.

## 2 MINEDOJO Simulator & Benchmark Suite

MINEDOJO offers a set of simulator APIs help researchers develop generally capable, open-ended agents in Minecraft. It builds upon the open-source MineRL codebase [48] and makes the following upgrades: 1) We provide **unified observation and action spaces** across all tasks, facilitating the development of multi-task and continually learning agents that can constantly adapt to new scenarios and novel tasks. This deviates from the MineRL Challenge design that tailors observation and action spaces to individual tasks; 2) Our simulation unlocks all three types of worlds in Minecraft, including the *Overworld*, the *Nether*, and the *End*, which **substantially expands the possible task space**, while MineRL only supports the Overworld natively; and 3) We provide convenient APIs to configure initial conditions and world settings to standardize our tasks.

With this MINEDOJO simulator, we define thousands of benchmarking tasks, which are divided into two categories: 1) *Programmatic tasks* that can be automatically assessed based on the ground-truth simulator states; and 2) *Creative tasks* that do not have well-defined or easily-automated success criteria, which motivates our novel evaluation protocol using a learned model (Sec. 4). To scale up the number of Creative tasks, we mine ideas from YouTube tutorials and use OpenAI's GPT-3 [15]

service to generate substantially more task definitions. Compared to Creative tasks, Programmatic tasks are simpler to get started, but tend to have restricted scope, limited language variations, and less open-endedness in general.

### 2.1 Task Suite I: Programmatic Tasks

We formalize each programmatic task as a 5-tuple: $T = (G, \mathcal{G}, \mathcal{I}, f_{\mathcal{S}}, f_{\mathcal{R}})$. $G$ is an English description of the task goal, such as "*find material and craft a gold pickaxe*". $\mathcal{G}$ is a natural language guidance that provides helpful hints, recipes, or advice to the agent. We leverage OpenAI's `GPT-3-davinci` API to automatically generate detailed guidance for a subset of the tasks. For the example goal "*bring a pig into Nether*", GPT-3 returns: `1) Find a pig in the overworld; 2) Right-click on the pig with a lead; 3) Right-click on the Nether Portal with the lead and pig selected; 4) The pig will be pulled through the portal!` $\mathcal{I}$ is the initial conditions of the agent and the world, such as the initial inventory, spawn terrain, and weather. $f_{\mathcal{S}}$: $s_t \rightarrow \{0, 1\}$ is the success criterion, a deterministic function that maps the current world state $s_t$ to a Boolean success label. $f_{\mathcal{R}}$: $s_t \rightarrow \mathbb{R}$ is an optional dense reward function. We only provide $f_{\mathcal{R}}$ for a small subset of the tasks in MINEDOJO due to the high costs of meticulously crafting dense rewards. For our current agent implementation (Sec. 4.1), we do not use detailed guidance. Inspired by concurrent works SayCan [3] and Socratic Models [143], one potential idea is to feed each step in the guidance to our learned reward model sequentially so that it becomes a stagewise reward function for a complex multi-stage task.

MINEDOJO provides 4 categories of programmatic tasks with 1,581 template-generated natural language goals to evaluate the agent's different capabilities systematically and comprehensively:

1. **Survival**: surviving for a designated number of days.
2. **Harvest**: finding, obtaining, cultivating, or manufacturing hundreds of materials and objects.
3. **Tech Tree**: crafting and using a hierarchy of tools.
4. **Combat**: fighting various monsters and creatures that require fast reflex and martial skills.

Each task template has a number of variations based on the terrain, initial inventory, quantity, etc., which form a flexible spectrum of difficulty. In comparison, the NeurIPS MineRL Diamond challenge [48] is a subset of our programmatic task suite, defined by the task goal "*obtain 1 diamond*" in MINEDOJO.

### 2.2 Task Suite II: Creative Tasks

We define each creative task as a 3-tuple, $T = (G, \mathcal{G}, \mathcal{I})$, which differs from programmatic tasks due to the lack of straightforward success criteria. Inspired by model-based metrics like the Inception score [98] and FID score [55] for image generation, we design a novel task evaluation metric based on a pre-trained contrastive video-language model (Sec. 4.1). In the experiments, we find that the learned metric exhibits a high level of agreement with human evaluations (see Table 2).

We brainstorm and author 216 Creative tasks, such as "*build a haunted house with zombie inside*" and "*race by riding a pig*". Nonetheless, such a manual approach is not scalable. Therefore, we develop two systematic approaches to extend the total number of task definitions to 1,560. This makes our Creative tasks *3 orders of magnitude* larger than Minecraft BASALT challenge [104], which has 4 Creative tasks.

**Approach 1. Task Mining from YouTube Tutorial Videos.** We identify our YouTube dataset as a rich source of tasks, as many human players demonstrate and narrate creative missions in the tutorial playlists. To collect high-quality tasks and accompanying videos, we design a 3-stage pipeline that makes it easy to find and annotate interesting tasks (see Sec. C.2 for details). Through this pipeline, we extract 1,042 task ideas from the common wisdom of a huge number of veteran Minecraft gamers, such as "*make an automated mining machine*" and "*grow cactus up to the sky*".

**Approach 2. Task Creation by GPT-3.** We leverage GPT-3's few-shot capability to generate new task ideas by seeding it with the tasks we manually author or mine from YouTube. The prompt template is: `Here are some example creative tasks in Minecraft: {a few examples}.`

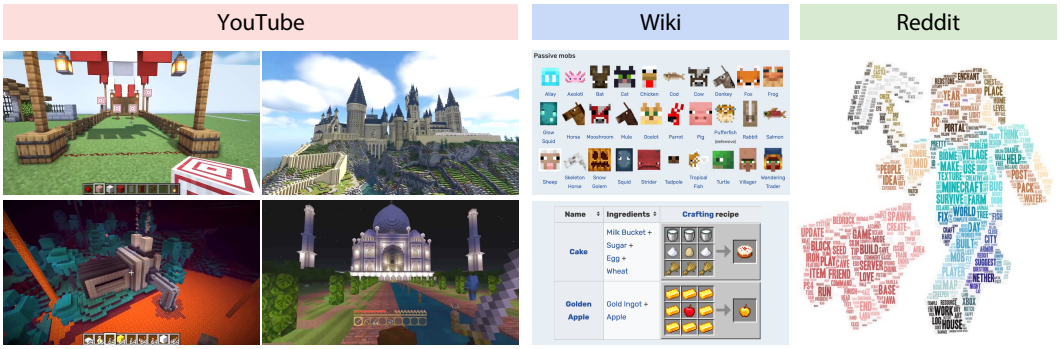



Figure 3: MINEDOJO's internet-scale, multimodal knowledge base. **Left, YouTube videos:** Minecraft gamers showcase the impressive feats they are able to achieve. Clockwise order: an archery range, Hogwarts castle, Taj Mahal, a Nether homebase. **Middle, Wiki:** Wiki pages contain multimodal knowledge in structured layouts, such as comprehensive catalogs of creatures and recipes for crafting. More examples in Fig. A.4 and A.5. **Right, Reddit:** We create a word cloud from Reddit posts and comment threads. Gamers ask questions, share achievements, and discuss strategies extensively. Sample posts in Fig. A.7. Best viewed zoomed in.

`Let's brainstorm more detailed while reasonable creative tasks in Minecraft.` GPT-3 contributes 302 creative tasks after de-duplication, and demonstrates a surprisingly proficient understanding of Minecraft terminology.

### 2.3 Collection of Starter Tasks

We curate a set of 64 core tasks for future researchers to get started more easily. If their agent works well on these tasks, they can more confidently scale to the full benchmark.

- **32 programmatic tasks**: 16 "standard" and 16 "difficult", spanning all 4 categories (survival, harvesting, combat, and tech tree). We rely on our Minecraft knowledge to decide the difficulty level. "Standard" tasks require fewer steps and lower resource dependencies to complete.
- **32 creative tasks**: 16 "standard" and 16 "difficult". Similarly, tasks labeled with "standard" are typically short-horizon tasks.

We recommend that researchers run 100 evaluation episodes for each task and report the percentage success rate. The programmatic tasks have ground-truth success, while the creative tasks need our novel evaluation protocol (Sec. 5).

## 3 Internet-scale Knowledge Base

Two commonly used approaches [112, 126, 85, 36] to train embodied agents include training agents from scratch using RL with well-tuned reward functions for each task, or using a large amount of human-demonstrations to bootstrap agent learning. However, crafting well-tuned reward functions is challenging or infeasible for our task suite (Sec. 2.2), and employing expert gamers to provide large amounts of demonstration data would also be costly and infeasible [126].

Instead, we turn to the open web as an ever-growing, virtually unlimited source of learning material for embodied agents. The internet provides a vast amount of domain knowledge about Minecraft, which we harvest by extensive web scraping and filtering. We collect 33 years worth of YouTube videos, 6K+ Wiki pages, and millions of Reddit comment threads. Instead of hiring a handful of human demonstrators, we capture the collective wisdom of millions of Minecraft gamers around the world. Furthermore, language is a key and pervasive component of our database that takes the form of YouTube transcripts, textual descriptions in Wiki, and Reddit discussions. Language facilitates open-vocabulary understanding, provides grounding for image and video modalities, and unlocks the power of large language models [27, 109, 15] for embodied agents. To ensure socially responsible model development, we take special measures to filter out low-quality and toxic contents [13, 51] from our databases, detailed in the Appendix (Sec. D).

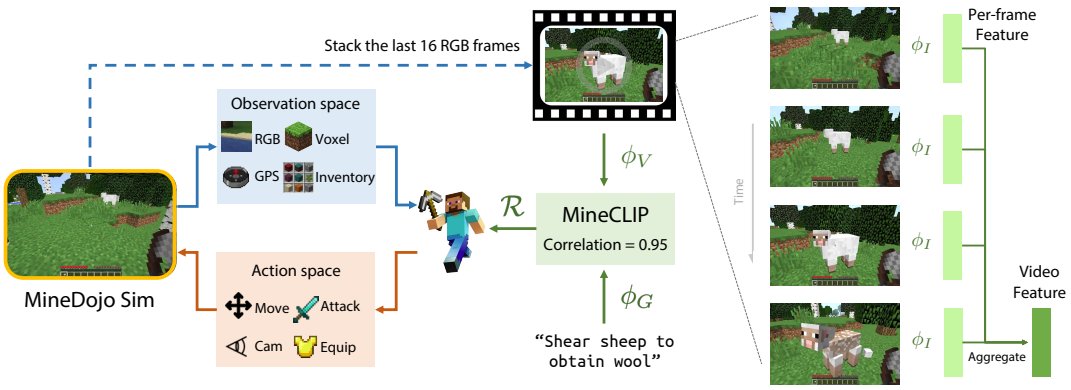



Figure 4: Algorithm design. MINECLIP is a contrastive video-language model pre-trained on MINEDOJO's massive Youtube database. It computes the correlation between an open-vocabulary language goal string and a 16-frame video snippet. The correlation score can be used as a learned dense reward function to train a strong multi-task RL agent.

**YouTube Videos and Transcripts.** Minecraft is among the most streamed games on YouTube [41]. Human players have demonstrated a stunning range of creative activities and sophisticated missions that take hours to complete (examples in Fig. 3). We collect 730K+ narrated Minecraft videos, which add up to 33 years of duration and 2.2B words in English transcripts. In comparison, HowTo100M [77] is a large-scale human instructional video dataset that includes 15 years of experience in total – about half of our volume. The time-aligned transcripts enable the agent to ground free-form natural language in video pixels and learn the semantics of diverse activities without laborious human labeling. We operationalize this insight in our pre-trained video-language model (Sec. 4.1).

**Minecraft Wiki.** The Wiki pages cover almost every aspect of the game mechanics, and supply a rich source of unstructured knowledge in multimodal tables, recipes, illustrations, and step-by-step tutorials. We use Selenium [103] to scrape 6,735 pages that interleave text, images, tables, and diagrams. The pages are highly unstructured and do not share any common schema, as the Wiki is meant for human consumption rather than AI training. To preserve the layout information, we additionally save the screenshots of entire pages and extract 2.2M bounding boxes of the visual elements (visualization in Fig. A.4 and A.5). We do not use Wiki data in our current experiments. Since the Wiki contains detailed recipes for all crafted objects, they could be provided as input or training data for hierarchical planning methods and policy sketches [8]. Another promising future direction is to apply document understanding models such as LayoutLM [138, 137] and DocFormer [9] to learn actionable knowledge from these unstructured Wiki data.

**Reddit.** We scrape 340K+ posts along with 6.6M comments under the "r/Minecraft" subreddit. These posts ask questions on how to solve certain tasks, showcase cool architectures and achievements in image/video snippets, and discuss general tips and tricks for players of all expertise levels. We do not use Reddit data for training in Sec. 5, but a potential idea is to finetune large language models [27, 91] on our Reddit corpus to generate instructions and execution plans that are better grounded in the Minecraft domain. Concurrent works [3, 56, 143] have explored similar ideas and showed excellent results on robot learning, which is encouraging for more future research in MINEDOJO.

## 4 Agent Learning with Large-scale Pre-training

One of the grand challenges of embodied AI is to build a single agent that can complete a wide range of open-world tasks. The MINEDOJO framework aims to facilitate new techniques towards this goal by providing an open-ended task suite (Sec. 2) and large-scale internet knowledge base (Sec. 3). Here we take an initial step towards this goal by developing a proof of concept that demonstrates how a single language-prompted agent can be trained in MINEDOJO to complete several complex Minecraft tasks. To this end, we propose a novel agent learning algorithm that takes advantage of the massive YouTube data offered by MINEDOJO. We note that this is only one of the numerous possible

Table 1: Our novel MINECLIP reward model is able to achieve competitive performance with manually written dense reward function for Programmatic tasks, and significantly outperforms the $\text{CLIP}_{\text{OpenAI}}$ method across all Creative tasks. Entries represent percentage success rates averaged over 3 seeds, each tested for 200 episodes. Success conditions are precise in Programmatic tasks, but estimated by MineCLIP for Creative tasks.

| Group | Tasks | **Ours** (Attn) | **Ours** (Avg) | Manual Reward | Sparse-only | $\text{CLIP}_{\text{OpenAI}}$ |
|---|---|---|---|---|---|---|
|  | Milk Cow | $\mathbf{64.5 \pm 37.1}$ | $6.5 \pm 3.5$ | $62.8 \pm 40.1$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ |
| | Hunt Cow | $\mathbf{83.5 \pm 7.1}$ | $0.0 \pm 0.0$ | $48.3 \pm 35.9$ | $0.3 \pm 0.4$ | $0.0 \pm 0.0$ |
| | Shear Sheep | $12.1 \pm 9.1$ | $0.6 \pm 0.2$ | $\mathbf{52.3 \pm 33.2}$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ |
| | Hunt Sheep | $8.1 \pm 4.1$ | $0.0 \pm 0.0$ | $\mathbf{41.9 \pm 33.0}$ | $0.3 \pm 0.4$ | $0.0 \pm 0.0$ |
|  | Combat Spider | $80.5 \pm 13.0$ | $60.1 \pm 42.5$ | $\mathbf{87.5 \pm 4.6}$ | $47.8 \pm 33.8$ | $0.0 \pm 0.0$ |
| | Combat Zombie | $47.3 \pm 10.6$ | $\mathbf{72.3 \pm 6.4}$ | $49.8 \pm 26.9$ | $8.8 \pm 12.4$ | $0.0 \pm 0.0$ |
| | Combat Pigman | $1.6 \pm 2.3$ | $0.0 \pm 0.0$ | $\mathbf{13.6 \pm 9.8}$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ |
| | Combat Enderman | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $0.3 \pm 0.2$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ |
|  | Find Nether Portal | $37.4 \pm 40.8$ | $\mathbf{89.8 \pm 5.7}$ | N/A | N/A | $26.3 \pm 32.6$ |
| | Find Ocean | $33.4 \pm 45.6$ | $\mathbf{54.3 \pm 40.7}$ | N/A | N/A | $9.9 \pm 14.1$ |
| | Dig Hole | $\mathbf{91.6 \pm 5.9}$ | $88.1 \pm 13.3$ | N/A | N/A | $0.0 \pm 0.0$ |
| | Lay Carpet | $97.6 \pm 1.9$ | $\mathbf{98.8 \pm 1.0}$ | N/A | N/A | $0.0 \pm 0.0$ |

ways to use MINEDOJO's internet database — the Wiki and Reddit corpus also hold great potential to drive new algorithm discoveries for the community in future works.

In this paper, we consider a multi-task reinforcement learning (RL) setting, where an agent is tasked with completing a collection of MINEDOJO tasks specified by language instructions (Sec. 2). Solving these tasks often requires the agent to interact with the Minecraft world in a prolonged fashion. Agents developed in popular RL benchmarks [119, 146] often rely on meticulously crafted dense and task-specific reward functions to guide random explorations. However, these rewards are hard or even infeasible to define for our diverse and open-ended tasks in MINEDOJO. To address this challenge, our key insight is to learn **a dense, language-conditioned reward function from in-the-wild YouTube videos and their transcripts**. Therefore, we introduce **MINECLIP**, a contrastive video-language model that learns to correlate video snippets and natural language descriptions (Fig. 4). MINECLIP is multi-task by design, as it is trained on open-vocabulary and diverse English transcripts.

During RL training, MINECLIP provides a high-quality reward signal *without* any domain adaptation techniques, despite the domain gap between noisy YouTube videos and clean simulator-rendered frames. MINECLIP eliminates the need to manually engineer reward functions for each and every MINEDOJO task. For Creative tasks that lack a simple success criterion (Sec. 2.2), MINECLIP also serves the dual purpose of an **automatic evaluation metric** that agrees well with human judgement on a subset of tasks we investigate (Sec. 4.2, Table 2). Because the learned reward model incurs a non-trivial computational overhead, we introduce several techniques to significantly improve RL training efficiency, making MINECLIP a practical module for open-ended agent learning in Minecraft (Sec. 4.2).

### 4.1 Pre-Training MINECLIP on Large-scale Videos

Formally, the learned reward function can be defined as $\Phi_{\mathcal{R}} : (G, V) \to \mathbb{R}$ that maps a language goal $G$ and a video snippet $V$ to a scalar reward. An ideal $\Phi_{\mathcal{R}}$ should return a high reward if the behavior depicted in the video faithfully follows the language description, and a low reward otherwise. This can be achieved by optimizing the InfoNCE objective [125, 52, 20], which learns to correlate positive video and text pairs [118, 6, 78, 4, 136].

Similar to the image-text CLIP model [92], MINECLIP is composed of a separate text encoder $\phi_G$ that embeds a language goal and a video encoder $\phi_V$ that embeds a moving window of 16 consecutive frames with $160 \times 256$ resolution (Fig. 4). Our neural architecture has a similar design as CLIP4Clip [75], where $\phi_G$ reuses OpenAI CLIP's pretrained text encoder, and $\phi_V$ is factorized into a frame-wise image encoder $\phi_I$ and a temporal aggregator $\phi_a$ that summarizes the sequence of 16 image features into a single video embedding. Unlike CLIP4Clip, we insert two extra layers of residual CLIP Adapter [38] after the aggregator $\phi_a$ to produce a better video feature, and finetune *only* the last two layers of the pretrained $\phi_I$ and $\phi_G$.

Table 2: MINECLIP agrees well with the ground-truth human judgment on the Creative tasks we consider. Numbers are F1 scores between MINECLIP's binary classification of tasks success and human labels (scaled to the percentage for better readability).

| Tasks | Find Nether Portal | Find Ocean | Dig Hole | Lay Carpet |
|---|---|---|---|---|
| Ours (Attn) | 98.7 | **100.0** | 99.4 | 97.4 |
| Ours (Avg) | **100.0** | **100.0** | **100.0** | **98.4** |
| $\text{CLIP}_{\text{OpenAI}}$ | 48.7 | 98.4 | 80.6 | 54.1 |

From the MINEDOJO YouTube database, we follow the procedure in VideoCLIP [136] to sample 640K pairs of 16-second video snippets and time-aligned English transcripts, after applying a keyword filter. We train two MINECLIP variants with different types of aggregator $\phi_a$: (1) MINECLIP[avg] does simple average pooling, which is fast but agnostic to the temporal ordering; (2) MINECLIP[attn] encodes the sequence by two transformer layers, which is relatively slower but captures more temporal information, and thus produces a better reward signal in general. Details of data preprocessing, architecture, and hyperparameters are listed in the Appendix (Sec. E).

### 4.2 RL with MINECLIP Reward

We train a language-conditioned policy network that takes as input raw pixels and predicts discrete control. The policy is trained with PPO [102] on the MINECLIP rewards. In each episode, the agent is prompted with a language goal and takes a sequence of actions to fulfill this goal. When calculating the MINECLIP rewards, we concatenate the agent's latest 16 egocentric RGB frames in a temporal window to form a video snippet. MINECLIP handles all task prompts *zero-shot* without any further finetuning. In our experiments (Sec. 5), we show that MINECLIP provides effective dense rewards out of the box, despite the domain shift between in-the-wild YouTube frames and simulator frames. Besides regular video data augmentation, we do not employ any special domain adaptation methods during pre-training. Our finding is consistent with CLIP's strong zero-shot performances on robustness benchmarks in object recognition [92].

Compared to hard-coded reward functions in popular benchmarks [146, 119, 34], the MINECLIP model has 150M parameters and is thus much more expensive to query. We make several design choices to greatly accelerate RL training with MINECLIP in the loop:

1. The language goal $G$ is fixed for a specific task, so the **text features $\phi_G$ can be precomputed** to avoid invoking the text encoder repeatedly.
2. Our agent's **RGB encoder reuses the pre-trained weights of $\phi_I$** from MINECLIP. We do not finetune $\phi_I$ during RL training, which saves computation and endows the agent with good visual representations from the beginning.
3. MINECLIP's video encoder $\phi_V$ is factorized into an image encoder $\phi_I$ and a light-weight aggregator $\phi_a$. This design choice enables **efficient image feature caching**. Consider two overlapping video sequences of 8 frames, `V[0:8]` and `V[1:9]`. We can cache the image features of the 7 overlapping frames `V[1]` to `V[7]` to maximize compute savings. If $\phi_V$ is a monolithic model like S3D [135] in VideoCLIP [136], then the video features from every sliding window must be recomputed, which would incur a much higher cost per time step.
4. We leverage **Self-Imitation Learning** [84] to store the trajectories with high MINECLIP reward values in a buffer, and alternate between PPO and self-imitation gradient steps. It further improves sample efficiency as shown in the Appendix (Fig. A.8).

## 5 Experiments

We evaluate our agent-learning approach (Section 4) on 8 Programmatic tasks and 4 Creative tasks from the MINEDOJO benchmarking suite. We select these 12 tasks due to the diversity of skills required to solve them (e.g., harvesting, combat, building, navigation) and domain-specific entities (e.g., animals, resources, monsters, terrains, and structures). We split the tasks into 3 groups and train one multi-task agent for each group: `Animal-Zoo` (4 Programmatic tasks on hunting or

Table 3: MINECLIP agents have stronger zero-shot visual generalization ability to unseen terrains, weathers, and lighting. Numbers outside parentheses are percentage success rates averaged over 3 seeds (each tested for 200 episodes), while those inside parentheses are relative performance changes.

| | Tasks | **Ours** (Attn), train | **Ours** (Attn), unseen test | $\text{CLIP}_{\text{OpenAI}}$, train | $\text{CLIP}_{\text{OpenAI}}$, unseen test |
|---|---|---|---|---|---|
|  | Milk Cow | $64.5 \pm 37.1$ | $\mathbf{64.8 \pm 31.3}(+\ \mathbf{0.8}\%)$ | $90.0 \pm 0.4$ | $29.2 \pm 3.7\ (-67.6\%)$ |
| | Hunt Cow | $83.5 \pm 7.1$ | $\mathbf{55.9 \pm 7.2}\ (\mathbf{-32.9}\%)$ | $72.7 \pm 3.5$ | $16.7 \pm 1.6\ (-77.0\%)$ |
| | Combat Spider | $80.5 \pm 13.0$ | $\mathbf{62.1 \pm 29.7}(\mathbf{-22.9}\%)$ | $79.5 \pm 2.5$ | $54.2 \pm 9.6\ (-31.8\%)$ |
| | Combat Zombie | $47.3 \pm 10.6$ | $\mathbf{39.9 \pm 25.3}(\mathbf{-15.4}\%)$ | $50.2 \pm 7.5$ | $30.8 \pm 14.4(-38.6\%)$ |

harvesting resource from animals), `Mob-Combat` (Programmatic, fight 4 types of hostile monsters), and `Creative` (4 tasks).

In the experiments, we empirically check the quality of MINECLIP against manually written reward functions, and quantify how different variants of our learned model affect the RL performance. Table 1 presents our main results, and Fig. 2 visualizes our learned agent behavior in 4 of the considered tasks. Policy networks of all methods share the same architecture and are trained by PPO + Self-Imitation (Sec. 4.2, training details in the Appendix, Sec. F). We compare the following methods:

- **Ours (Attn)**: our agent trained with the MINECLIP[attn] reward model. For Programmatic tasks, we also add the final success condition as a binary reward. For Creative tasks, MINECLIP is the only source of reward.
- **Ours (Avg)**: the average-pooling variant of our method.
- **Manual Reward**: hand-engineered dense reward using ground-truth simulator states.
- **Sparse-only**: the final binary success as a single sparse reward. Note that neither sparse-only nor manual reward is available for Creative tasks.
- **$\text{CLIP}_{\text{OpenAI}}$**: pre-trained OpenAI CLIP model that has **not** been finetuned on any MINEDOJO videos.

**MINECLIP is competitive with manual reward.** For Programmatic tasks (first 8 rows), RL agents guided by MINECLIP achieve competitive performance as those trained by manual reward. In three of the tasks, they even *outperform* the hand-engineered reward functions, which rely on privileged simulator states unavailable to MINECLIP. For a more statistically sound analysis, we conduct the Paired Student's *t*-test to compare the mean success rate of each task (pairing column 3 "Ours (Attn)" and column 5 "Manual Reward" in Table 1). The test yields p-value $0.3991 \gg 0.05$, which indicates that the difference between our method and manual reward is not considered statistically significant, and therefore they are comparable with each other. Because all tasks require nontrivial exploration, our approach also dominates the Sparse-only baseline. Note that the original OpenAI CLIP model fails to achieve any success. We hypothesize that the creatures in Minecraft look dramatically different from their real-world counterparts, which causes CLIP to produce *misleading* signals worse than no shaping reward at all. It implies the importance of finetuning on MINEDOJO's YouTube data.

**MINECLIP provides automated evaluation.** For Creative tasks (last 4 rows), there are no programmatic success criteria available. We convert MINECLIP into a binary success classifier by thresholding the reward value it outputs for an episode. To test the quality of MINECLIP as an automatic evaluation metric, we ask human judges to curate a dataset of 100 successful and 100 failed trajectories for each task. We then run both MINECLIP variants and $\text{CLIP}_{\text{OpenAI}}$ on the dataset and report the binary F1 score of their judgement against human ground-truth in Table 2. The results demonstrate that both MINECLIP[attn] and MINECLIP[avg] attain a very high degree of agreement with human evaluation results on this subset of the Creative task suite that we investigate. $\text{CLIP}_{\text{OpenAI}}$ baseline also achieves nontrivial agreement on Find Ocean and Dig Hole tasks, likely because real-world oceans and holes have similar texture. We use the `attn` variant as an automated success criterion to score the 4 Creative task results in Table 1. Our proposed method consistently learns better than $\text{CLIP}_{\text{OpenAI}}$-guided agents. It shows that MINECLIP is an effective approach to solving open-ended tasks when no straightforward reward signal is available. We provide further analysis beyond these 4 tasks in the Appendix (Sec. G.4).

Table 4: We train a single multi-task agent for all 12 tasks. All numbers represent percentage success rates averaged over 3 seeds, each tested for 200 episodes.

| Group | Tasks | Single Agent on All Tasks | Original | Performance Change |
|---|---|---|---|---|
|  | Milk Cow | $\mathbf{91.5 \pm 0.7}$ | $64.5 \pm 37.1$ | $\uparrow$ |
| | Hunt Cow | $46.8 \pm 3.7$ | $\mathbf{83.5 \pm 7.1}$ | $\downarrow$ |
| | Shear Sheep | $\mathbf{73.5 \pm 0.8}$ | $12.1 \pm 9.1$ | $\uparrow$ |
| | Hunt Sheep | $\mathbf{27.0 \pm 1.0}$ | $8.1 \pm 4.1$ | $\uparrow$ |
|  | Combat Spider | $72.1 \pm 1.3$ | $\mathbf{80.5 \pm 13.0}$ | $\downarrow$ |
| | Combat Zombie | $27.1 \pm 2.7$ | $\mathbf{47.3 \pm 10.6}$ | $\downarrow$ |
| | Combat Pigman | $\mathbf{6.5 \pm 1.2}$ | $1.6 \pm 2.3$ | $\uparrow$ |
| | Combat Enderman | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $=$ |
|  | Find Nether Portal | $\mathbf{99.1 \pm 0.4}$ | $37.4 \pm 40.8$ | $\uparrow$ |
| | Find Ocean | $\mathbf{95.1 \pm 1.5}$ | $33.4 \pm 45.6$ | $\uparrow$ |
| | Dig Hole | $85.8 \pm 1.2$ | $\mathbf{91.6 \pm 5.9}$ | $\downarrow$ |
| | Lay Carpet | $96.5 \pm 0.8$ | $\mathbf{97.6 \pm 1.9}$ | $=$ |

Table 5: We test the open-vocabulary generalization ability to two unseen tasks. All numbers represent percentage success rates averaged over 3 seeds, each tested for 200 episodes.

| | Tasks | Ours (zero-shot) | Ours (after RL finetune) | Baseline (RL from scratch) |
|---|---|---|---|---|
|  | Hunt Pig | $1.3 \pm 0.6$ | $\mathbf{46.0 \pm 15.3}$ | $0.0 \pm 0.0$ |
| | Harvest Spider String | $1.6 \pm 1.3$ | $\mathbf{36.5 \pm 16.9}$ | $12.5 \pm 12.7$ |

**MINECLIP shows good zero-shot generalization to significant visual distribution shift.** We evaluate the learned policy without finetuning on a combination of unseen weather, lighting conditions, and terrains — 27 scenarios in total. For the baseline, we train agents with the original $\text{CLIP}_{\text{OpenAI}}$ image encoder (not trained on our YouTube videos) by imitation learning. The robustness against visual shift can be quantitatively measured by the relative performance degradation on novel test scenarios for each task. Table 3 shows that while all methods incur performance drops, agents with MINECLIP encoder is more robust to visual changes than the baseline across all tasks. Pre-training on diverse in-the-wild YouTube videos is important to boosting zero-shot visual generalization capability, a finding consistent with literature [92, 82].

**Learning a Single Agent for All 12 Tasks** We have also trained a single agent for all 12 tasks. To reduce the computational burden without loss of generality, the agent is trained by self-imitating from successful trajectories generated from the self-imitation learning pipeline (Section F.3). We summarize the result in Table 4. Similar to our main experiments, all numbers represent percentage success rates averaged over 3 training seeds, each tested for 200 episodes. Compared to the original agents, the new 12-multitask agent sees a performance boost in 6 tasks, degradation in 4 tasks, and roughly the same success rates in 2 tasks. This result suggests that there are both positive and negative task transfers happening. To improve the multi-task performance, more advanced algorithms [141, 133] can be employed to mitigate the negative transfer effects.

We also perform Paired Student's *t*-test to statistically compare the performance of the 12-multitask agent and those separately trained on each task group. We obtain a p-value of $0.3720 \gg 0.05$, which suggests that the difference between the two training settings is not statistically significant.

**Generalize to Novel Tasks** To test the ability to generalize to new open-vocabulary commands, we evaluate the agent on two novel tasks: "harvest spider string" and "hunt pig". Table 5 shows that the agent struggles in the zero-shot setting because it has not interacted with pigs or spider strings during training, and thus does not know how to interact with them effectively. However, the performance improves substantially by finetuning with the MINECLIP reward. Here the baseline methods are trained from scratch using RL with the MINECLIP encoders and reward. Therefore, the only difference is whether the policy has been pre-trained on the 12 tasks or not. Given the

same environment sampling budget (only around 5% of total samples), it significantly outperforms baselines. It suggests that the multitask agent has learned transferable knowledge on hunting and resource collection, which enables it to quickly adapt to novel tasks.

## 6 Related work

**Open-ended Environments for Decision-making Agents.** There are many environments developed with the goal of open-ended agent learning. Prior works include maze-style worlds [121, 129, 61], purely text-based game [69], grid worlds [21, 16], browser/GUI-based environments [108, 124], and indoor simulators for robotics [1, 107, 114, 34, 110, 99, 89]. Minecraft offers an exciting alternative for open-ended agent learning. It is a 3D visual world with procedurally generated landscapes and extremely flexible game mechanics that support an enormous variety of activities. Prior methods in open-ended agent learning [30, 57, 130, 63, 26] do not make use of external knowledge, but our approach leverages internet-scale database to learn open-vocabulary reward models, thanks to Minecraft's abundance of gameplay data online.

**Minecraft for AI Research.** The Malmo platform [60] is the first comprehensive release of a Gym-style agent API [14] for Minecraft. Based on Malmo, MineRL [48] provides a codebase and human play trajectories for the annual Diamond Challenge at NeurIPS [47, 49, 62]. MINEDOJO's simulator builds upon the pioneering work of MineRL, but greatly expands the API and benchmarking task suite. Other Minecraft benchmarks exist with different focuses. For example, CraftAssist [44] and IGLU [66] study agents with interactive dialogues. BASALT [104] applies human evaluation to 4 open-ended tasks. EvoCraft [45] is designed for structure building, and Crafter [50] optimizes for fast experimentation. Unlike prior works, MINEDOJO's core mission is to facilitate the development of generally capable embodied agents using internet-scale knowledge. We include a feature comparison table of different Minecraft platforms for AI research in Table A.1.

**Internet-scale Multimodal Knowledge Bases.** Big dataset such as Common Crawl [24], the Pile [37], LAION [100], YouTube-8M [2] and HowTo100M [77] have been fueling the success of large pre-trained language models [27, 91, 15] and multimodal models [118, 6, 78, 145, 7, 4, 136]. While generally useful for learning representations, these datasets are not specifically targeted at embodied agents. To provide agent-centric training data, RoboNet [25] collects video frames from 7 robot platforms, and Ego4D [43] recruits volunteers to record egocentric videos of household activities. In comparison, MINEDOJO's knowledge base is constructed without human curation efforts, much larger in volume, more diverse in data modalities, and comprehensively covers all aspects of the Minecraft environment.

**Embodied Agents with Large-scale Pre-training.** Inspired by the success in NLP, embodied agent research [29, 11, 94, 23] has seen a surge in adoption of the large-scale pre-training paradigm. The recent advances can be roughly divided into 4 categories. 1) **Novel agent architecture**: Decision Transformer [19, 58, 144] applies the powerful self-attention models to sequential decision making. GATO [95] and Unified-IO [74] learn a single model to solve various decision-making tasks with different control interfaces. VIMA [59] unifies a wide range of robot manipulation tasks with multimodal prompting. 2) **Pre-training for better representations**: R3M [82] trains a general-purpose visual encoder for robot perception on Ego4D videos [43]. CLIPort [111] leverages the pre-trained CLIP model [92] to enable free-form language instructions for robot manipulation. 3) **Pre-training for better policies**: AlphaStar [126] achieves champion-level performance on StarCraft by imitating from numerous human demos. SayCan [3] leverages large language models (LMs) to ground value functions in the physical world. [72] and [96] directly reuse pre-trained LMs as policy backbone. VPT [10] is a concurrent work that learns an inverse dynamics model from human contractors to pseudo-label YouTube videos for behavior cloning. VPT is complementary to our approach, and can be finetuned to solve language-conditioned open-ended tasks with our learned reward model. 4) **Data-driven reward functions**: Concept2Robot [105] and DVD [18] learn a binary classifier to score behaviors from in-the-wild videos [42]. LOReL [81] crowd-sources humans labels to train language-conditioned reward function for offline RL. AVID [113] and XIRL [142] extract reward signals via cycle consistency. MINEDOJO's task benchmark and internet knowledge base are generally useful for developing new algorithms in all the above categories. In Sec. 4, we also propose an open-vocabulary, multi-task reward model using MINEDOJO YouTube videos.

## 7 Conclusion

In this work, we introduce the MINEDOJO framework for developing generally capable embodied agents. MINEDOJO features a benchmarking suite of thousands of Programmatic and Creative tasks, and an internet-scale multimodal knowledge base of videos, wiki, and forum discussions. As an example of the novel research possibilities enabled by MINEDOJO, we propose MINECLIP as an effective language-conditioned reward function trained with in-the-wild YouTube videos. MINECLIP achieves strong performance empirically and agrees well with human evaluation results, making it a good automatic metric for Creative tasks. We look forward to seeing how MINEDOJO empowers the community to make progress on the important challenge of open-ended agent learning.

## 8 Acknowledgement

We are extremely grateful to Anssi Kanervisto, Shikun Liu, Zhiding Yu, Chaowei Xiao, Weili Nie, Jean Kossaifi, Jonathan Raiman, Neel Kant, Saad Godil, Jaakko Haapasalo, Bryan Catanzaro, John Spitzer, Zhiyuan "Jerry" Lin, Yingqi Zheng, Chen Tessler, Dieter Fox, Oli Wright, Jeff Clune, Jack Parker-Holder, and many other colleagues and friends for their helpful feedback and insightful discussions. We also thank the anonymous reviewers for offering us highly constructive advice and kind encouragement during the review and rebuttal period. NVIDIA provides the necessary computing resource and infrastructure for this project. Guanzhi Wang is supported by the Kortschak fellowship in Computing and Mathematical Sciences at Caltech.

## References

[1] Josh Abramson, Arun Ahuja, Iain Barr, Arthur Brussee, Federico Carnevale, Mary Cassin, Rachita Chhaparia, Stephen Clark, Bogdan Damoc, Andrew Dudzik, Petko Georgiev, Aurelia Guy, Tim Harley, Felix Hill, Alden Hung, Zachary Kenton, Jessica Landon, Timothy Lillicrap, Kory Mathewson, Soňa Mokrá, Alistair Muldal, Adam Santoro, Nikolay Savinov, Vikrant Varma, Greg Wayne, Duncan Williams, Nathaniel Wong, Chen Yan, and Rui Zhu. Imitating interactive intelligence. *arXiv preprint arXiv: Arxiv-2012.05672*, 2020.

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv: Arxiv-1609.08675*, 2016.

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv: Arxiv-2204.01691*, 2022.

[4] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. *arXiv preprint arXiv: Arxiv-2104.11178*, 2021.

[5] Rami Al-Rfou, Marc Pickett, Javier Snaider, Yun hsuan Sung, Brian Strope, and Ray Kurzweil. Conversational contextual cues: The case of personalization and history for response ranking. *arXiv preprint arXiv: Arxiv-1606.00372*, 2016.

[6] Jean-Baptiste Alayrac, Adrià Recasens, Rosalia Schneider, Relja Arandjelovic, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. Self-supervised multimodal versatile networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/0060ef47b12160b9198302ebdb144dcf-Abstract.html`.

[7] Elad Amrani, Rami Ben-Ari, Daniel Rotman, and Alex M. Bronstein. Noise estimation using density estimation for self-supervised multimodal learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6644–6652. AAAI Press, 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16822`.

[8] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 166–175. PMLR, 2017. URL `http://proceedings.mlr.press/v70/andreas17a.html`.

[9] Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. Docformer: End-to-end transformer for document understanding. *arXiv preprint arXiv: Arxiv-2106.11539*, 2021.

[10] 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: Arxiv-2206.11795*, 2022.

[11] Dhruv Batra, Angel X. Chang, Sonia Chernova, Andrew J. Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv: Arxiv-2011.01975*, 2020.

[12] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv: Arxiv-2006.13171*, 2020.

[13] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *arXiv preprint arXiv: Arxiv-2108.07258*, 2021.

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv: Arxiv-1606.01540*, 2016.

[15] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[16] Tianshi Cao, Jingkang Wang, Yining Zhang, and Sivabalan Manivasagam. Babyai++: Towards grounded-language learning beyond memorization. *arXiv preprint arXiv: Arxiv-2004.07200*, 2020.

[17] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4724–4733. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.502. URL https://doi.org/10.1109/CVPR.2017.502.

[18] Annie S. Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from in-the-wild human videos. In Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh, editors, *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021. doi: 10.15607/RSS.2021.XVII.012. URL https://doi.org/10.15607/RSS.2021.XVII.012.

[19] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15084–15097, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

[20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20j.html.

[21] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJeXCo0cYX.

[22] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv: Arxiv-2204.02311*, 2022.

[23] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. A review of physics simulators for robotic applications. *IEEE Access*, 9:51416–51431, 2021.

[24] Common Crawl. Common crawl. https://commoncrawl.org/, 2012. Accessed: 2022-06-06.

[25] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 885–897. PMLR, 2019. URL http://proceedings.mlr.press/v100/dasari20a.html.

[26] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre M. Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33:*

*Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

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

[28] 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. *arXiv preprint arXiv: Arxiv-2010.11929*, 2020.

[29] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied AI: from simulators to research tasks. *IEEE Trans. Emerg. Top. Comput. Intell.*, 6(2):230–244, 2022. doi: 10.1109/TETCI.2022.3141105. URL https://doi.org/10.1109/TETCI.2022.3141105.

[30] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv: Arxiv-1901.10995*, 2019.

[31] Ashley D. Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L. Isbell. Imitating latent policies from observation. *arXiv preprint arXiv: Arxiv-1805.07914*, 2018.

[32] William Falcon and The PyTorch Lightning team. PyTorch Lightning. *Github*, 3 2019. doi: 10.5281/zenodo.3828935. URL https://github.com/PyTorchLightning/pytorch-lightning.

[33] Linxi Fan*, Shyamal Buch*, Guanzhi Wang, Ryan Cao, Yuke Zhu, Juan Carlos Niebles, and Li Fei-Fei. Rubiksnet: Learnable 3d-shift for efficient video action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[34] Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Anima Anandkumar. Secant: Self-expert cloning for zero-shot generalization of visual policies. *arXiv preprint arXiv: Arxiv-2106.09678*, 2021.

[35] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv: Arxiv-2004.07219*, 2020.

[36] Florian Fuchs, Yunlong Song, Elia Kaufmann, Davide Scaramuzza, and Peter Dürr. Super-human performance in gran turismo sport using deep reinforcement learning. *IEEE Robotics Autom. Lett.*, 6(3):4257–4264, 2021. doi: 10.1109/LRA.2021.3064284. URL https://doi.org/10.1109/LRA.2021.3064284.

[37] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[38] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv preprint arXiv: Arxiv-2110.04544*, 2021.

[39] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna M. Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Commun. ACM*, 64(12): 86–92, 2021. doi: 10.1145/3458723. URL https://doi.org/10.1145/3458723.

[40] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv: Arxiv-2009.11462*, 2020.

[41] Jordan Gerblick. Minecraft, the most-watched game on youtube, passes 1 trillion views, Dec 2021. URL https://www.gamesradar.com/minecraft-the-most-watched-game-on-youtube-passes-1-trillion-views/.

[42] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

[43] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Abrham Gebreselasie, Cristina Gonzalez, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jachym Kolar, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Ziwei Zhao, Yunyi Zhu, Pablo Arbelaez, David Crandall, Dima Damen, Giovanni Maria Farinella, Christian Fuegen, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the world in 3,000 hours of egocentric video. *arXiv preprint arXiv: Arxiv-2110.07058*, 2021.

[44] Jonathan Gray, Kavya Srinet, Yacine Jernite, Haonan Yu, Zhuoyuan Chen, Demi Guo, Siddharth Goyal, C. Lawrence Zitnick, and Arthur Szlam. Craftassist: A framework for dialogue-enabled interactive agents. *arXiv preprint arXiv: Arxiv-1907.08584*, 2019.

[45] Djordje Grbic, Rasmus Berg Palm, Elias Najarro, Claire Glanois, and Sebastian Risi. *EvoCraft: A New Challenge for Open-Endedness*, pages 325–340. Springer International Publishing, 2021. doi: 10.1007/978-3-030-72699-7_21. URL `http://link.springer.com/content/pdf/10.1007/978-3-030-72699-7_21`.

[46] Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. Metamorph: Learning universal controllers with transformers. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=Opmqtk_GvYL`.

[47] William H. Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, Manuela Veloso, and Phillip Wang. The minerl 2019 competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv: Arxiv-1904.10079*, 2019.

[48] William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv: Arxiv-1907.13440*, 2019.

[49] William H. Guss, Mario Ynocente Castro, Sam Devlin, Brandon Houghton, Noboru Sean Kuno, Crissman Loomis, Stephanie Milani, Sharada Mohanty, Keisuke Nakata, Ruslan Salakhutdinov, John Schulman, Shinya Shiroshita, Nicholay Topin, Avinash Ummadisingu, and Oriol Vinyals. The minerl 2020 competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv: Arxiv-2101.11071*, 2021.

[50] Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv: Arxiv-2109.06780*, 2021.

[51] Laura Hanu and Unitary team. Detoxify. Github. `https://github.com/unitaryai/detoxify`, 2020.

[52] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00975. URL `https://doi.org/10.1109/CVPR42600.2020.00975`.

[53] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv: Arxiv-2111.06377*, 2021.

[54] Matthew Henderson, Paweł Budzianowski, Iñigo Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrkšić, Georgios Spithourakis, Pei-Hao Su, Ivan Vulić, and Tsung-Hsien Wen. A repository of conversational datasets. *arXiv preprint arXiv: Arxiv-1904.06472*, 2019.

[55] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

[56] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv: Arxiv-2207.05608*, 2022.

[57] Joost Huizinga and Jeff Clune. Evolving multimodal robot behavior via many stepping stones with the combinatorial multiobjective evolutionary algorithm. *Evolutionary computation*, 30 (2):131–164, 2022.

[58] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1273–1286, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/099fe6b0b444c23836c4a5d07346082b-Abstract.html.

[59] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv: Arxiv-2210.03094*, 2022.

[60] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. *IJCAI*, 2016. URL https://dl.acm.org/doi/10.5555/3061053.3061259.

[61] Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2684–2691. ijcai.org, 2019. doi: 10.24963/ijcai.2019/373. URL https://doi.org/10.24963/ijcai.2019/373.

[62] 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: Arxiv-2202.10583*, 2022.

[63] Ingmar Kanitscheider, Joost Huizinga, David Farhi, William Hebgen Guss, Brandon Houghton, Raul Sampedro, Peter Zhokhov, Bowen Baker, Adrien Ecoffet, Jie Tang, Oleg Klimov, and Jeff Clune. Multi-task curriculum learning in a complex, visual, hard-exploration domain: Minecraft. *arXiv preprint arXiv: Arxiv-2106.14876*, 2021.

[64] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv: Arxiv-1705.06950*, 2017.

[65] Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. Abstractive summarization of reddit posts with multi-level memory networks. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2519–2531. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1260. URL https://doi.org/10.18653/v1/n19-1260.

[66] Julia Kiseleva, Ziming Li, Mohammad Aliannejadi, Shrestha Mohanty, Maartje ter Hoeve, Mikhail Burtsev, Alexey Skrynnik, Artem Zholus, Aleksandr Panov, Kavya Srinet, Arthur Szlam, Yuxuan Sun, Katja Hofmann, Michel Galley, and Ahmed Awadallah. Neurips 2021 competition iglu: Interactive grounded language understanding in a collaborative environment. *arXiv preprint arXiv: Arxiv-2110.06536*, 2021.

[67] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv: Arxiv-2205.11916*, 2022.

[68] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv: Arxiv-1712.05474*, 2017.

[69] Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7671–7684. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/569ff987c643b4bedf504efda8f786c2-Paper.pdf.

[70] Label Studio. Label studio. https://labelstud.io/, 2020. Accessed: 2022-06-06.

[71] WB Langdon. Pfeiffer–a distributed open-ended evolutionary system. In *AISB*, volume 5, pages 7–13. Citeseer, 2005.

[72] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, Jacob Andreas, Igor Mordatch, Antonio Torralba, and Yuke Zhu. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv: Arxiv-2202.01771*, 2022.

[73] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

[74] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks. *arXiv preprint arXiv: Arxiv-2206.08916*, 2022.

[75] 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. *arXiv preprint arXiv: Arxiv-2104.08860*, 2021.

[76] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

[77] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. *arXiv preprint arXiv: Arxiv-1906.03327*, 2019.

[78] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9876–9886. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00990. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Miech_End-to-End_Learning_of_Visual_Representations_From_Uncurated_Instructional_Videos_CVPR_2020_paper.html.

[79] Minecraft Wiki. Minecraft wiki. hhttps://minecraft.fandom.com/wiki/Minecraft_Wiki, 2016. Accessed: 2022-06-06.

[80] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv: Arxiv-1312.5602*, 2013.

[81] Suraj Nair, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 1303–1315. PMLR, 2021. URL https://proceedings.mlr.press/v164/nair22a.html.

[82] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv: Arxiv-2203.12601*, 2022.

[83] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. *arXiv preprint arXiv: Arxiv-2205.07802*, 2022.

[84] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887. PMLR, 2018.

[85] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv: Arxiv-1912.06680*, 2019.

[86] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv: Arxiv-1912.01703*, 2019.

[87] Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, Noburu Kuno, Andre Kramer, Sam Devlin, Raluca D. Gaina, and Daniel Ionita. The multi-agent reinforcement learning in malmÖ (marlÖ) competition. *arXiv preprint arXiv: Arxiv-1901.08129*, 2019.

[88] PRAW: The Python Reddit API Wrapper. Praw: The python reddit api wrapper. https://github.com/praw-dev/praw, 2010. Accessed: 2022-06-06.

[89] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8494–8502. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00886. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Puig_VirtualHome_Simulating_Household_CVPR_2018_paper.html.

[90] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI*, 2018.

[91] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[92] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[93] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv: Arxiv-2204.06125*, 2022.

[94] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020.

[95] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *arXiv preprint arXiv: Arxiv-2205.06175*, 2022.

[96] Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv: Arxiv-2201.12122*, 2022.

[97] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv: Arxiv-2205.11487*, 2022.

[98] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234, 2016. URL `https://proceedings.neurips.cc/paper/2016/hash/8a3363abe792db2d8761d6403605aeb7-Abstract.html`.

[99] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[100] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[101] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1506.02438`.

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

[103] Selenium WebDriver. Selenium webdriver. `https://www.selenium.dev/`, 2011. Accessed: 2022-06-06.

[104] Rohin Shah, Cody Wild, Steven H. Wang, Neel Alex, Brandon Houghton, William Guss, Sharada Mohanty, Anssi Kanervisto, Stephanie Milani, Nicholay Topin, Pieter Abbeel, Stuart Russell, and Anca Dragan. The minerl basalt competition on learning from human feedback. *arXiv preprint arXiv: Arxiv-2107.01969*, 2021.

[105] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14):1419–1434, 2021.

[106] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv: Arxiv-2002.05202*, 2020.

[107] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne P. Tchapmi, Micael E. Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. *arXiv preprint arXiv: Arxiv-2012.02924*, 2020.

[108] Tianlin Tim Shi, Andrej Karpathy, Linxi Jim Fan, Jonathan Hernandez, and Percy Liang. World of bits: an open-domain platform for web-based agents. *ICML*, 2017. URL `https://dl.acm.org/doi/10.5555/3305890.3306005`.

[109] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[110] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10737–10746. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.01075. URL `https://openaccess.thecvf.com/content_CVPR_2020/html/Shridhar_ALFRED_A_Benchmark_for_Interpreting_Grounded_Instructions_for_Everyday_Tasks_CVPR_2020_paper.html`.

[111] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. *arXiv preprint arXiv: Arxiv-2109.12098*, 2021.

[112] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv: Arxiv-1712.01815*, 2017.

[113] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv preprint arXiv: Arxiv-1912.04443*, 2019.

[114] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, C. Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. BEHAVIOR: benchmark for everyday household activities in virtual, interactive, and ecological environments. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 477–490. PMLR, 2021. URL `https://proceedings.mlr.press/v164/srivastava22a.html`.

[115] Bradly C. Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *arXiv preprint arXiv: Arxiv-1703.01703*, 2017.

[116] Russell K Standish. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(02):167–175, 2003.

[117] Kenneth O Stanley, Joel Lehman, and Lisa Soros. Open-endedness: The last grand challenge you've never heard of. *O'Reilly Online,*, 2017.

[118] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Learning video representations using contrastive bidirectional transformer. *arXiv preprint arXiv: Arxiv-1906.05743*, 2019.

[119] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite. *arXiv preprint arXiv: Arxiv-1801.00690*, 2018.

[120] Tim Taylor, Mark Bedau, Alastair Channon, David Ackley, Wolfgang Banzhaf, Guillaume Beslon, Emily Dolson, Tom Froese, Simon Hickinbotham, Takashi Ikegami, et al. Open-ended evolution: Perspectives from the oee workshop in york. *Artificial life*, 22(3):408–423, 2016.

[121] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv: Arxiv-2107.12808*, 2021.

[122] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv: Arxiv-1805.01954*, 2018.

[123] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv: Arxiv-1905.13566*, 2019.

[124] Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. Androidenv: A reinforcement learning platform for android. *arXiv preprint arXiv: Arxiv-2105.13231*, 2021.

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

[126] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, et al. Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind blog*, 2, 2019.

[127] Michael Vølske, Martin Potthast, Shahbaz Syed, and Benno Stein. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark, sep 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4508. URL `https://aclanthology.org/W17-4508`.

[128] Phil Wang. x-transformers. *Github*, 2022. URL `https://github.com/lucidrains/x-transformers`.

[129] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv: Arxiv-1901.01753*, 2019.

[130] Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeff Clune, and Kenneth O. Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. *arXiv preprint arXiv: Arxiv-2003.08536*, 2020.

[131] Wikipedia contributors. Minecraft — Wikipedia, the free encyclopedia, 2022. URL `https://en.wikipedia.org/w/index.php?title=Minecraft&oldid=1092238294`. [Online; accessed 9-June-2022].

[132] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv: Arxiv-1910.03771*, 2019.

[133] Sen Wu, Hongyang R. Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=SylzhkBtDB`.

[134] Fei Xia, William B. Shen, Chengshu Li, Priya Kasimbeg, Micael Tchapmi, Alexander Toshev, Li Fei-Fei, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark (igibson 0.5): A benchmark for interactive navigation in cluttered environments. *arXiv preprint arXiv: Arxiv-1910.14442*, 2019.

[135] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 305–321, 2018.

[136] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding. *arXiv preprint arXiv: Arxiv-2109.14084*, 2021.

[137] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. Layoutlmv2: Multimodal pre-training for visually-rich document understanding. *arXiv preprint arXiv: Arxiv-2012.14740*, 2020.

[138] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. *arXiv preprint arXiv: Arxiv-1912.13318*, 2019.

[139] Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Learning semantic textual similarity from conversations. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 164–174, Melbourne, Australia, jul 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-3022. URL `https://aclanthology.org/W18-3022`.

[140] YouTube Data API. Youtube data api. `https://developers.google.com/youtube/v3/`, 2012. Accessed: 2022-06-06.

[141] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv: Arxiv-2001.06782*, 2020.

[142] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. *arXiv preprint arXiv: Arxiv-2106.03911*, 2021.

[143] Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv: Arxiv-2204.00598*, 2022.

[144] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. *arXiv preprint arXiv: Arxiv-2202.05607*, 2022.

[145] Linchao Zhu and Yi Yang. Actbert: Learning global-local video-text representations. *arXiv preprint arXiv: Arxiv-2011.07231*, 2020.

[146] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv: Arxiv-2009.12293*, 2020.

# A Minecraft Framework Comparison

Table A.1: Comparison table of different Minecraft platforms for AI research.

| Environment | Simulator | | Task Suite | | Knowledge Base | |
|---|---|---|---|---|---|---|
| | Features | Real Minecraft | Number of Tasks | Language-grounded | Features | Data Scale |
| MINEDOJO | Unified observation and action space; unlocks all three types of world (the Overworld, the Nether, and the End) | ✓ | 3,000+ | ✓ | Automatically scraped from the Internet; multimodal data (videos, images, texts, tables and diagrams) | 740K YouTube videos; 7K Wiki pages; 350K Reddit posts |
| MineRL v0.4 [48] | Built on top of Malmo; actively maintained | ✓ | 11 | | Annotated state-action pairs of human demonstrations | 60M frames of recorded human player data |
| MineRL v1.0 (VPT) [10] | Mouse and keyboard control | ✓ | 5 | | Labeled contractor data; unlabeled videos scraped from the Internet | 2K hours of contractor data; 270K hours of unlabeled videos |
| MarLÖ [87] | Cooperative and competitive multiagent tasks; parameterizable environments | ✓ | 14 | | | |
| Malmo [60] | First comprehensive release of a Gym-style agent API for Minecraft | ✓ | N/A | | | |
| CraftAssist [44] | Bot assistant; dialogue interactions | ✓ | N/A | ✓ | Interactive dialogues; crowd-sourced house building dataset | 800K dialogue-action dictionary pairs; 2.6K houses with atomic building actions |
| IGLU [66] | Interactive dialogues with humans; aimed at building structures described by natural language | | 157 | ✓ | | |
| EvoCraft [45] | Aimed at generating creative artifacts; allows for direction manipulation of blocks | | N/A | | | |
| Crafter [50] | 2D clone of Minecraft; fast experimentation | | 22 | | Human experts dataset | 100 episodes |

# B MINEDOJO Simulator

We design unified observation and action spaces across all tasks to facilitate the development of multi-tasking and continually learning agents that can adapt to novel tasks and scenarios. The codebase is open sourced at github.com/MineDojo/MineDojo.

## B.1 Observation Space

Our observation space contains multiple modalities. The agent perceives the world mainly through the RGB screen. To provide the same information as human players receive, we also supplement the agent with observations about its inventory, location, health, surrounding blocks, etc. The full observation space is shown below. We refer readers to see our code documentation for technical details such as data type for each observable item.

We also support a LIDAR sensor that returns the groundtruth type of the blocks that the agent sees, however this is considered privileged information and does not go into the benchmarking specification. However, it is still useful for hand engineering the dense reward function, which we use in our experiments (Sec. 5). Amounts and directions of LIDAR rays can be arbitrarily configured at the cost of a lower simulation throughput.

| Modality | Shape | Description |
|---|---|---|
| RGB | `(3, H, W)` | Ego-centric RGB frames. |
| Equipment | `(6,)` | Names, quantities, variants, and durabilities of equipped items. |
| Inventory | `(36,)` | Names, quantities, variants, and durabilities of inventory items. |
| Voxel | `(3, 3, 3)` | Names, variants, and properties of $3 \times 3 \times 3$ surrounding blocks. |
| Life statistics | `(1,)` | Agent's health, oxygen, food saturation, etc. |
| GPS | `(3,)` | GPS location of the agent. |
| Compass | `(2,)` | Yaw and pitch of the agent. |
| Nearby tools | `(2,)` | Indicate if crafting table and furnace are nearby the agent. |
| Damage source | `(1,)` | Information about the damage on the agent. |
| Lidar | `(Num rays,)` | Ground-truth lidar observation. |

### B.2 Action Space

We design a compound action space. At each step the agent chooses one movement action (forward, backward, camera actions, etc.) and one optional functional action as listed in the table below. Some functional actions such as `craft` take one argument, while others like `attack` does not take any argument. This compound action space can be modelled in an autoregressive manner [126]. We refer readers to our code documentation for example usages of our action space.

| Name | Description | Argument |
|---|---|---|
| `no_op` | Do nothing. | $\varnothing$ |
| `use` | Use the item held in the main hand. | $\varnothing$ |
| `drop` | Drop the item held in the main hand. | $\varnothing$ |
| `attack` | Attack with barehand or tool held in the main hand. | $\varnothing$ |
| `craft` | Execute a crafting recipe to obtain new items. | Index of recipe |
| `equip` | Equip an inventory item. | Slot index of the item |
| `place` | Place an inventory item on the ground. | Slot index of the item |
| `destroy` | Destroy an inventory item. | Slot index of the item |

### B.3 Customizing the Environment

Environments in MINECLIP simulator can be easily and flexibly customized. Through our simulator API, users can control terrain, weather, day-night condition (different lighting), the spawn rate and range of specified entities and materials, etc. We support a wide range of terrains, such as desert, jungle, taiga, and iced plain, and special in-game structures, such as ocean monument, desert temple, and End city. Please visit our website for video demonstrations.

## C MINEDOJO Task Suite

In this section, we explain how we collect the Programmatic (Sec. 2.1) and Creative tasks (Sec. 2.2).

### C.1 Programmatic Tasks

Programmatic tasks are constructed by filling manually written templates for four categories of tasks, namely "Survival", "Harvest", "Tech Tree", and "Combat". The task specifications are included in our codebase. Please refer to Fig. A.1 for a few samples. We briefly explain each task category:

**Survival.** This task group tests the ability to stay alive in the game. It is nontrivial to survive in Minecraft, because the agent grows hungry as time passes and the health bar drops gradually. Hostile mobs like zombie and skeleton spawn at night, which are very dangerous if the agent does not have the appropriate armor to protect itself or weapons to fight back. We provide two tasks with different levels of difficulty for Survival. One is to start from scratch without any assistance. The other is to start with initial weapons and food.

```yaml
survival_sword_food:
  category: survival
  prompt: survive as long as possible given a sword and some food

harvest_wool_with_shears_and_sheep:
  category: harvest
  prompt: harvest wool from a sheep with shears and a sheep nearby

techtree_from_barehand_to_wooden_sword:
  category: tech-tree
  prompt: find material and craft a wooden sword

combat_zombie_pigman_nether_diamond_armors_diamond_sword_shield:
  category: combat
  prompt: combat a zombie pigman in nether with a diamond sword,
    shield, and a full suite of diamond armors
```

Figure A.1: Example specifications.

**Harvest.** This task group tests the agent's ability to collect useful resources such as minerals (iron, diamond, obsidian), food (beef, pumpkin, carrots, milk), and other useful items (wool, oak wood, coal). We construct these tasks by enumerating the Cartesian product between target items to collect, initial inventory, and world conditions (terrain, weather, lighting, etc.) so that they cover a spectrum of difficulty. For instance, if the task is to harvest wool, then it is relatively easy if the agent has a shear in its initial inventory with a sheep nearby, but more difficult if the agent has to craft the shear from raw material and explore extensively to find a sheep. We filter out combinations that are impossible (such as farming certain plants in the desert) from the Cartesian product.

**Tech Tree.** Minecraft includes several levels of tools and armors with different properties and difficulties to unlock. To progress to a higher level of tools and armors, the agent needs to develop systematic and compositional skills to navigate the technology tree (e.g. wood $\rightarrow$ stone $\rightarrow$ iron $\rightarrow$ diamond). In this task group, the agent is asked to craft and use a hierarchy of tools starting from a less advanced level. For example, some task asks the agent to craft a wooden sword from bare hand. Another task may ask the agent to craft a gold helmet. An agent that can successfully complete these tasks should have the ability to transfer similar exploration strategies to different tech levels.

**Combat.** We test agent's reflex and martial skills to fight against various monsters and creatures. Similar to how we develop the Harvest task group, we generate these tasks by enumerating the Cartesian product between the target entity to combat with, initial inventory, and world conditions to cover a spectrum of difficulty.

### C.2 Creative Tasks

We construct Creative tasks using three approaches: 1) manual brainstorming, 2) mining from YouTube tutorial videos, and 3) generate by querying GPT-3 API. We elaborate the second and third approaches below.

**Task Mining from YouTube Tutorial Videos.** Our YouTube dataset serves the dual purpose of a rich task source, as many human players demonstrate and narrate creative missions in the tutorial playlists. To collect high-quality tasks and accompanying videos, we design a 3-stage pipeline that makes it easy to find and annotate interesting tasks.

Stage 1: We search for YouTube playlists with the key phrases, "Minecraft Tutorial" and "Minecraft Guide". Then we apply heuristic rules (see Sec. D.1) to filter out low-quality videos;

Stage 2: We only show the title of the video to a human annotator through a command-line interface, who makes a binary decision to accept or reject it as a potential task. This step is typically very fast, taking a few seconds on average;

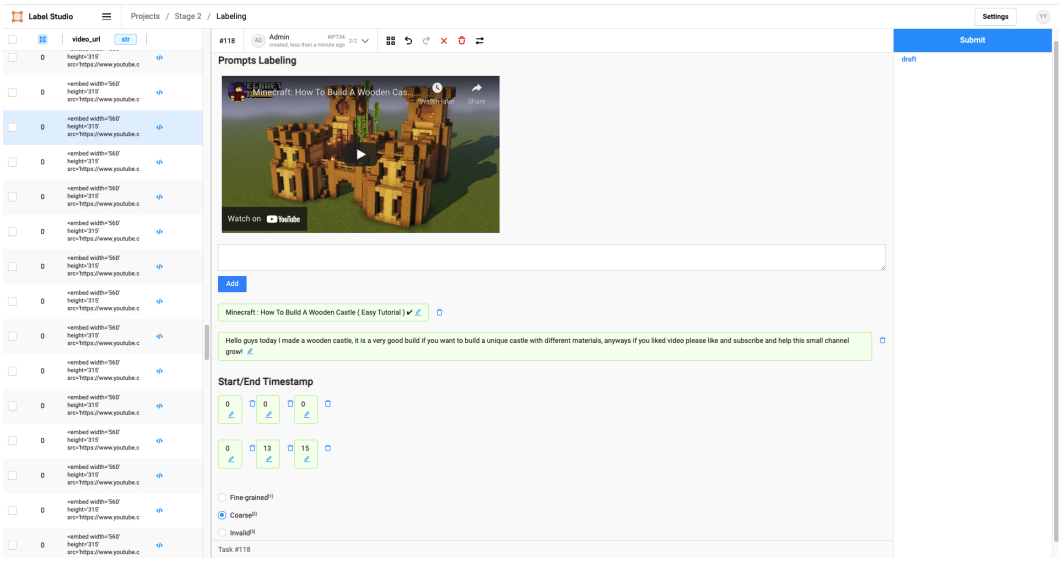



Figure A.2: Labeling UI to mine tasks from YouTube. A human annotator can choose to reject the video (*Invalid*), adjust the timestamps, select the title, or edit and expand the original description to be the new task goal.

Stage 3: For the accepted tasks in stage 2, we design a labeling UI using Label Studio [70] that displays the full video and YouTube description. A human annotator can choose to reject the video, adjust the timestamps, select the title, or refine the description to be the task goal (Fig. A.2). Through this pipeline, we extract 1,042 task ideas from the common wisdom of a huge number of veteran Minecraft gamers. Some examples are "*make an automated mining machine*" and "*grow cactus up to the sky*".

### C.3 GPT-3 Guidance

We leverage OpenAI's `GPT-3-davinci` API to automatically generate detailed guidance for a subset of the tasks. Inspired by [67], we adopt the following template to prompt GPT-3: `How to {task goal} in Minecraft? Let's think step by step.` Here are some examples:

The guidance for the task "*find material and craft a gold pickaxe*" is `1) Find a place with a lot of trees; 2) Cut down the trees and gather the wood; 3) Find a place with a lot of stone; 4) Mine the stone and gather the cobblestone; 5) Find a place with a lot of iron; 6) Mine the iron and gather the iron ingots; 7) Find a place with a lot of gold; 8) Mine the gold and gather the gold ingots; 9) Craft a gold pickaxe.`

The guidance for the task "*sail on boat with a sheep*" is `1) Find a boat; 2) Place the sheep in the boat; 3) Right-click on the boat with an empty hand to get in; 4) Use the WASD keys to move the boat. The sheep should stay in the boat.`

### C.4 Playthrough: Defeat the Ender Dragon

Our benchmarking suite includes a special task called "Playthrough". The agent is initialized bare-handed in a freshly created world and aims to defeat the *Ender dragon*, which is considered the final boss of Minecraft. This task holds a unique position in our benchmark because killing the dragon means "beating the game" in the traditional sense of the phrase, and is considered the most significant achievement for a new player. This boss is optional and plenty of people choose to skip it without affecting their open-ended game experience.

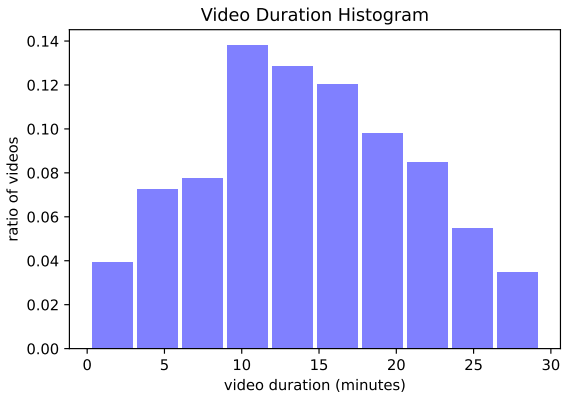



Figure A.3: Distribution of YouTube video duration. The histogram is trimmed by the 85th percentile to hide much longer videos that can run for many hours.

"Playthrough" is technically a programmatic task, because we can check the simulator state for the defeat of the Ender dragon. However, we decide to create its own category due to the uniqueness as well as the sheer difficulty of the task. The mission requires lots of preparation, exploration, agility, and trial-and-error, which may take a regular human player many days to complete. It would be extremely long horizon (hundreds of thousands of steps) and difficult for an agent to tackle. We consider this one of the moonshot goals in MINEDOJO.

# D Internet-Scale Database

We upload our databases to `zenodo.org`, which is an open repository platform operated by CERN. The data DOIs, URLs, and licenses are listed below. In this section, we describe our database properties and data collection process in details.

| Database | DOI | License |
|---|---|---|
| YouTube | `10.5281/zenodo.6641142` | Creative Commons Attribution 4.0 International (CC BY 4.0) |
| Wiki | `10.5281/zenodo.6640448` | Creative Commons Attribution Non Commercial Share Alike 3.0 Unported |
| Reddit | `10.5281/zenodo.6641114` | Creative Commons Attribution 4.0 International (CC BY 4.0) |

## D.1 YouTube Videos and Transcripts

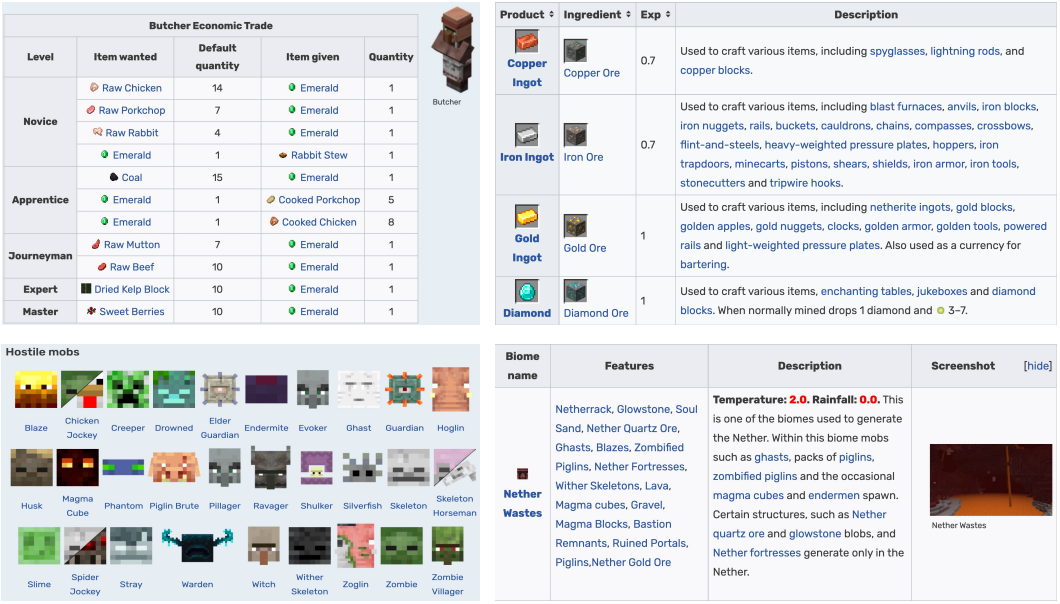
Minecraft is among the most streamed games on YouTube [41]. Human players have demonstrated a stunning range of creative activities and sophisticated missions that take hours to complete. We collect 33 years worth of video and 2.2B words in the accompanying English transcripts. The distribution of video duration is shown in Fig. A.3. The time-aligned transcripts enable the agent to ground free-form natural language in video pixels and learn the semantics of diverse activities without laborious human labeling.

We use the official YouTube Data API [140] to collect our database, following the procedure below:

a) Search for *channels* that contain Minecraft videos using a list of keywords, e.g., "Minecraft", "Minecraft Guide", "Minecraft Walkthrough", "Minecraft Beginner". We do not directly search for videos at this step because there is a limit of total results returned by the API;

b) Search for all the video IDs uploaded by each channel that we obtain at the previous step. There are many false positives at this step because some channels (like gaming news channel) may cover a range of topics other than Minecraft;

c) To remove the false positives, we rely on the video category chosen by the user when the video was uploaded and filter out all the videos that do not belong to the Minecraft category;



**Butcher Economic Trade**

| Level | Item wanted | Default quantity | Item given | Quantity |
|---|---|---|---|---|
| Novice | Raw Chicken | 14 | Emerald | 1 |
| | Raw Porkchop | 7 | Emerald | 1 |
| | Raw Rabbit | 4 | Emerald | 1 |
| | Emerald | 1 | Rabbit Stew | 1 |
| Apprentice | Coal | 15 | Emerald | 1 |
| | Emerald | 1 | Cooked Porkchop | 5 |
| | Emerald | 1 | Cooked Chicken | 8 |
| Journeyman | Raw Mutton | 7 | Emerald | 1 |
| | Raw Beef | 10 | Emerald | 1 |
| Expert | Dried Kelp Block | 10 | Emerald | 1 |
| Master | Sweet Berries | 10 | Emerald | 1 |

Butcher

| Product | Ingredient | Exp | Description |
|---|---|---|---|
| Copper Ingot | Copper Ore | 0.7 | Used to craft various items, including spyglasses, lightning rods, and copper blocks. |
| Iron Ingot | Iron Ore | 0.7 | Used to craft various items, including blast furnaces, anvils, iron blocks, iron nuggets, rails, buckets, cauldrons, chains, compasses, crossbows, flint-and-steels, heavy-weighted pressure plates, hoppers, iron trapdoors, minecarts, pistons, shears, shields, iron armor, iron tools, stonecutters and tripwire hooks. |
| Gold Ingot | Gold Ore | 1 | Used to craft various items, including netherite ingots, gold blocks, golden apples, gold nuggets, clocks, golden armor, golden tools, powered rails and light-weighted pressure plates. Also used as a currency for bartering. |
| Diamond | Diamond Ore | 1 | Used to craft various items, enchanting tables, jukeboxes and diamond blocks. When normally mined drops 1 diamond and 3–7. |

Hostile mobs

Blaze, Chicken Jockey, Creeper, Drowned, Elder Guardian, Endermite, Evoker, Ghast, Guardian, Hoglin, Husk, Magma Cube, Phantom, Piglin Brute, Pillager, Ravager, Shulker, Silverfish, Skeleton, Skeleton Horseman, Slime, Spider Jockey, Stray, Warden, Witch, Wither Skeleton, Zoglin, Zombie, Zombie Villager

| Biome name | Features | Description | Screenshot [hide] |
|---|---|---|---|
| Nether Wastes | Netherrack, Glowstone, Soul Sand, Nether Quartz Ore, Ghasts, Blazes, Zombified Piglins, Nether Fortresses, Wither Skeletons, Lava, Magma cubes, Gravel, Magma Blocks, Bastion Remnants, Ruined Portals, Piglins,Nether Gold Ore | **Temperature: 2.0. Rainfall: 0.0.** This is one of the biomes used to generate the Nether. Within this biome mobs such as ghasts, packs of piglins, zombified piglins and the occasional magma cubes and endermen spawn. Certain structures, such as Nether quartz ore and glowstone blobs, and Nether fortresses generate only in the Nether. | Nether Wastes |

Figure A.4: Wiki dataset examples. Closewise order: Villager trade table, mineral ingredient descriptions, monster gallery, and terrain explanation.

d) To curate a language-grounded dataset, we favor videos that have English transcripts, which can be manually uploaded by the user, automatically transcribed from audio, or automatically translated from another language by the YouTube engine. For each video, we filter it out if 1) the view count is less than 100; or 2) the aspect ratio is less 1; or 3) the duration is less than 1 minute long; or 4) marked as age-restricted.

e) To further clean the dataset and remove potentially harmful contents, we employ the Detoxify [51] tool to process each video title and description. Detoxify is trained on Wikipedia comments to predict multiple types of toxicity like threats, obscenity, insults, and identity-based hate speech. We delete a video if the toxicity probability in any category is above 0.5.

We release all the video IDs along with metadata such as video titles, view counts, like counts, duration, and FPS. In line with prior practice [64], we do not release the actual MP4 files and transcripts due to legal concerns.

### D.2 Minecraft Wiki

The Wiki pages cover almost every aspect of the game mechanics, and supply a rich source of unstructured knowledge in multimodal tables, recipes, illustrations, and step-by-step tutorials (example screenshots in Fig. A.4 and Fig. A.5). We use Selenium [103] to scrape 6,735 pages that interleave text, images, tables, and diagrams. We elaborate the details of each web element scraped by Selenium:

a) **Screenshot.** Using Selenium's built-in function, we take a full screenshot of the rendered Wiki page in order to preserve the human-readable visual formatting. We also record the bounding boxes of each salient web element on the page.

b) **Text.** We hand-select several HTML tags that likely contain meaningful text data, such as `p, h1, h2, ul, dl`.

c) **Images and Animations.** We download the raw source file of each image element (JPG, PNG, GIF, etc.), as well as the corresponding caption if available. There are also animation effects enabled by JavaScript on the Wiki. We save all image frames in the animation.

d) **Sprites.** Sprite elements are micro-sized image icons that are typically embedded in text to create multimodal tutorials and explanations. We save all the sprites and locate their bounding boxes within the text too.

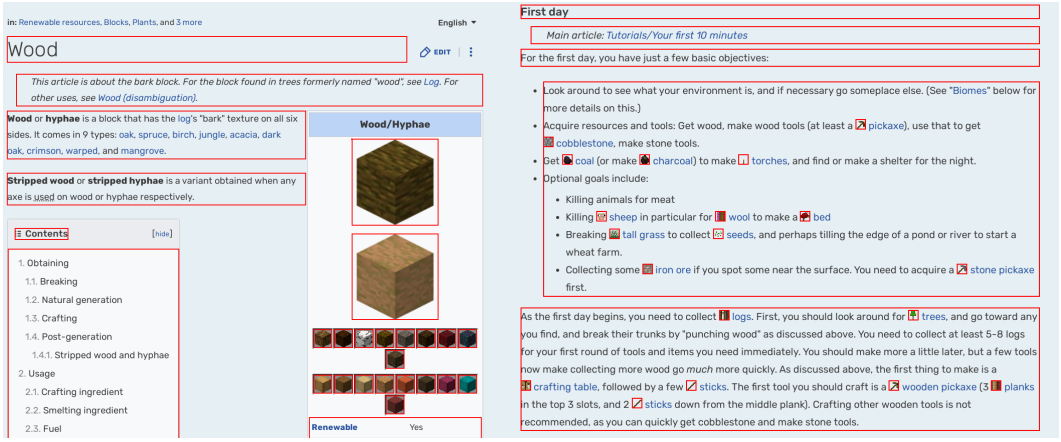



Figure A.5: More Wiki database examples with bounding boxes (annotated in red). Left: wood block introduction; right: first day tutorial.

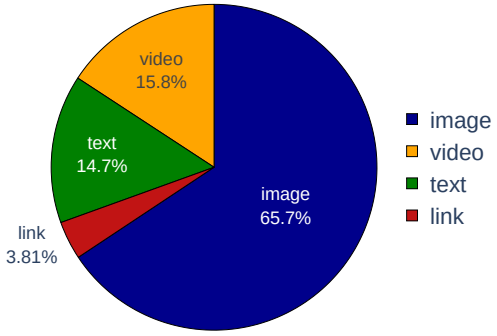



Figure A.6: Distribution of Reddit post types.

e) **Tables.** We save the text content and bounding box of each cell that a table element contains. We store the header cells separately as they carry the semantic meaning of each column. A table can be easily reconstructed with the stored text strings and bounding boxes.

### D.3 Reddit

There are more than 1M subreddits (i.e., Reddit topics) where people can discuss a wide range of themes and subjects. Prior works use Reddit data for conversational response selection [5, 139, 54] and abstractive summarization [127, 65]. The r/Minecraft subreddit contains free-form discussions of game strategies and images/videos showcases of Minecraft builds and creations (examples in Fig. A.7). The distribution of post types is shown in Fig. A.6.

To scrape the Reddit contents, we use PRAW [88], a Python wrapper on top of the official Reddit API. Our procedure is as follows:

a) Obtain the ID and metadata (e.g. post title, number of comments, content, score) of every post in the "r/Minecraft" subreddit since it was created. For quality control, we only consider posts with scores (upvotes) $\geq 5$ and not marked as NSFW.

b) Determine each post's type. There are 4 native post types - text, image/video, link, and poll. We group text and poll posts together as text posts, and store their body text. For image/video and link posts, we store the source file URLs on external media hosting sites like Imgur and Gfycat. Based on the URL of each link post, we classify it as an image post, a video post or a general link post.

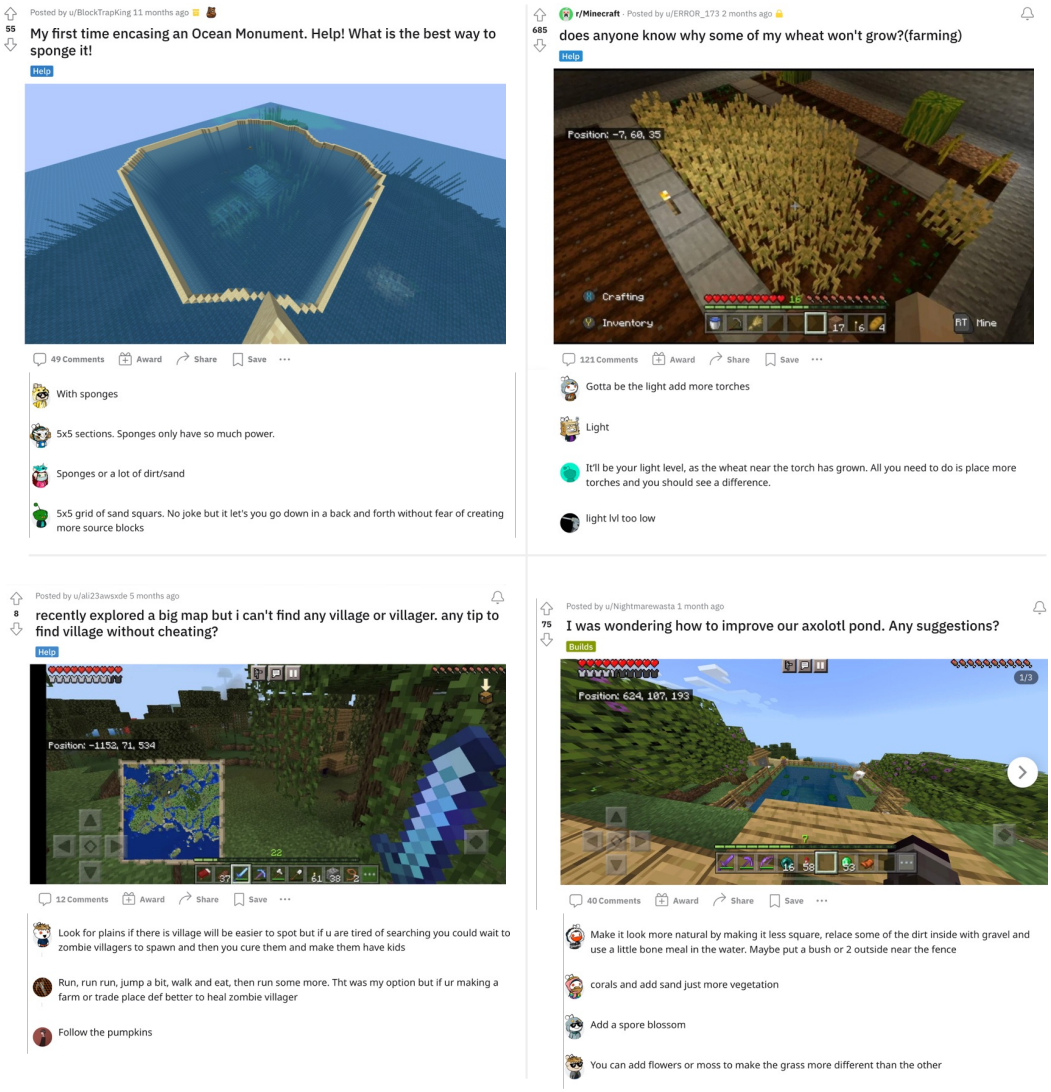



Figure A.7: Examples of posts and comment threads from the Reddit database.

c) Scrape the comments and store the parent ID of each comment so that we can reconstruct the threaded discussion.

d) Similarly to our YouTube database (Sec. D.1), we run Detoxify [51] on the scraped Reddit contents to filter out potentially toxic and harmful posts.

We release all post IDs and their corresponding metadata. We also provide a Python function based on PRAW for researchers to download the post contents after obtaining a license key for the official Reddit API.

# E MINECLIP Algorithm Details

We implement all our neural networks in PyTorch v1.11 [86]. Training MINECLIP uses the PyTorch-Lightning framework [32], pre-trained models hosted on HuggingFace [132], and the `x-transformers` library for Transformer variants [128].

Table A.2: Training hyperparameters for MINECLIP.

| Hyperparameter | Value |
|---|---|
| LR schedule | Cosine with warmup [73] |
| Warmup steps | 500 |
| Peak LR | 1.5e-4 |
| Final LR | 1e-5 |
| Weight decay | 0.2 |
| Layerwise LR decay | 0.65 |
| Pre-trained layers LR multiplier | $0.5\times$ |
| Batch size per GPU | 64 |
| Parallel GPUs | 8 |
| Video resolution | $160 \times 256$ |
| Number of frames | 16 |
| Image encoder | `ViT-B/16` [28] |

### E.1 Video-Text Pair Extraction

Similar to VideoCLIP [136], we sample 640K pairs of 16-second video snippets and time-aligned English transcripts by the following procedure:

1) Collect a list of keywords corresponding to the supported entities, blocks, and items in Minecraft;
2) Perform string matching over our YouTube video transcripts to obtain 640K text segments;
3) For each matched transcript segment, randomly grow it to $16 \sim 77$ tokens (limited by CLIP's context length);
4) Randomly sample a timestamp within the start and end time of the matched transcript as the center for the video clip;
5) Randomly grow the video clip from the center timestamp to $8 \sim 16$ seconds.

### E.2 Architecture

MINECLIP architecture is composed of three parts:

**Frame-wise image encoder $\phi_I$** We use the ViT-B/16 architecture [28] to compute a 512-D embedding for each RGB frame. We initialize the weights from OpenAI CLIP's public checkpoint [92] and only finetune the last two layers during training. The input resolution is $160 \times 256$, which is different from CLIP's default $224 \times 224$ resolution. We adapt the positional embeddings via bicubic interpolation, which does not introduce any new learnable parameters.

**Temporal aggregator $\phi_a$** Given a sequence of frame-wise RGB features, a temporal aggregator network summarizes the sequence into one video embedding. After the aggregator, we insert two extra layers of residual CLIP Adapter [38]. The residual weight is initialized such that it is very close to an identity function at the beginning of training. We consider two variants of $\phi_a$:

1. Average pooling (MINECLIP[avg]): a simple, parameter-free operator. It is fast to execute but loses the temporal information, because average pooling is permutation-invariant.
2. Self-Attention (MINECLIP[attn]): a 2-layer transformer encoder with 512 embedding size, 8 attention heads, and Gated Linear Unit variant with Swish activation [106, 22]. The transformer sequence encoder is relatively slower, but captures more temporal information and achieves better performance in our experiments (Table 1).

**Text encoder $\phi_G$** We use a 12-layer 512-wide GPT model with 8 attention heads [90, 91]. The input string is converted to lower-case byte pair encoding with a 49,152 vocabulary size, and capped at 77 tokens. We exactly follow the text encoder settings in CLIP and initialize the weights from their public checkpoint. Only the last two layers of $\phi_G$ is finetuned during training.

**Algorithm 1:** PPO-SI Interleaved Training

**Input:** policy $\pi_\theta$, value function $VF(\cdot)$, SI buffer threshold $\Delta$, SI frequency $\omega$

Initialize empty SI buffers for all tasks $\mathbf{D}_{SI} \leftarrow \{\emptyset, \forall T \in \text{training tasks}\}$;
Initialize a counter for simulator steps $counter \leftarrow 0$;
**while** *not done* **do**
  Collect set of trajectories for all tasks $\{\tau_T, \forall T \in \text{training tasks}\}$ by running policy $\pi_\theta$ in (parallel) environments;
  **forall** $\mathcal{D}_{SI,T}$ **do**
    **if** $\tau_T$ *is successful* **then**
      $\mathcal{D}_{SI,T} \leftarrow \mathcal{D}_{SI,T} \cup \tau_T$
    **else if** $\tau_T$*'s episode return* $\geq \mu_{return}(\mathcal{D}_{SI,T}) + \Delta \times \sigma_{return}(\mathcal{D}_{SI,T})$ **then**
      $\mathcal{D}_{SI,T} \leftarrow \mathcal{D}_{SI,T} \cup \tau_T$
  **end**
  Increase $counter$ accordingly;
  Update $\pi_\theta$ following Equation 2;
  Fit $VF(\cdot)$ by regression on mean-squared error;
  **if** $\mathbb{1}(counter \bmod \omega = 0)$ **then**
    Determine the number of trajectories to sample from each buffer $\#_{\text{sample}} = \min(\{|\mathcal{D}_{SI,T}|, \forall T \in \text{training tasks}\})$;
    Sample $\#_{\text{sample}}$ trajectories from each buffer in a prioritized manner to construct $\mathcal{D}_{SI}$;
    Update $\pi_\theta$ on $\mathcal{D}_{SI}$ with supervised objective;
**end**

### E.3 Training

We train MINECLIP on the 640K video-text pairs for 2 epochs. We sample 16 RGB frames from each video uniformly, and apply temporally-consistent random resized crop [17, 33] as data augmentation. We use Cosine learning rate annealing with 500 gradient steps of warming up [73]. We apply a lower learning rate ($\times 0.5$) on the pre-trained weights and layer-wise learning rate decay for better finetuning [53]. Training is performed on 1 node of $8\times$ V100 GPUs with FP16 mixed precision [76] via the PyTorch native `amp` module. All hyperparameters are listed in Table A.2.

## F Policy Learning Details

In this section, we elaborate how a trained MINECLIP can be adapted as a reward function with two different formulations. We then discuss the algorithm for policy learning. Finally, we demonstrate how we combine self imitation learning and on-policy learning to further improve sample efficiency.

### F.1 Adapt MINECLIP as Reward Function

We investigate two ways to convert MINECLIP output to scalar reward, dubbed DIRECT and DELTA. The ablation results for `Animal-Zoo` task group are presented in Table A.3.

**Direct.** For a task $T$ with the goal description $G$, MINECLIP outputs the probability $P_G$ that the observation video semantically corresponds to $G$, against a set of negative goal descriptions $\mathcal{G}^-$. Note that we omit timestep subscript for simplicity. As an example, for the task "shear sheep", $G$ is "shear a sheep" and $\mathcal{G}^-$ may include negative prompts like "milk a cow", "hunt a sheep", "hunt a cow", etc. To compute the DIRECT reward, we further process the raw probability using the formula $r = \max(P_G - \frac{1}{N_T}, 0)$ where $N_T$ is the number of prompts passed to MINECLIP. $\frac{1}{N_T}$ is the baseline probability of randomly guessing which text string corresponds to the video. We threshold $r$ at zero to avoid highly uncertain probability estimates below the random baseline. We call the variant without the post-processing DIRECT-Naive: $r = P_G$ as the reward signal for every time step.

**Delta.** The DIRECT formulation yields strong performance when the task is concerned with moving creatures, e.g. farm animals and monsters that run around constantly. However, we discover that DIRECT is suboptimal if the task deals with static objects, e.g., "find a nether portal". Simply using the

Table A.3: Ablation on different MINECLIP reward formulations.

| Group | Tasks | DIRECT | DIRECT-Naive | DELTA |
|---|---|---|---|---|
|  | Milk Cow | $\mathbf{64.5 \pm 37.1}$ | $8.6 \pm 1.2$ | $7.6 \pm 5.2$ |
| | Hunt Cow | $\mathbf{83.5 \pm 7.1}$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ |
| | Shear Sheep | $\mathbf{12.1 \pm 9.1}$ | $0.8 \pm 0.6$ | $1.8 \pm 1.5$ |
| | Hunt Sheep | $\mathbf{8.1 \pm 4.1}$ | $0.1 \pm 0.2$ | $0.0 \pm 0.0$ |

raw probability from MINECLIP as reward can cause the learned agent to stare at the object of interest but fail to move closer and interact. Therefore, we propose to use an alternative formulation, DELTA, to remedy this issue. Concretely, the reward value at timestep $t$ becomes $r_t = P_{G,t} - P_{G,t-1}$. We empirically validate that this formulation provides better shaped reward for the task group with static entities.

### F.2 Policy Network Architecture

Our policy architecture consists of three parts: an input feature encoder, a policy head, and a value function. To handle multimodal observations (Sec. B.1), the feature extractor contains several modality-specific components:

- RGB frame: we use the frozen frame-wise image encoder $\phi_I$ in MINECLIP to optimize for compute efficiency and provide the agent with good visual representations from the beginning (Sec. 4.2).
- Task goal: $\phi_G$ computes the text embedding of the natural language task goal.
- Yaw and Pitch: compute $\sin(\cdot)$ and $\cos(\cdot)$ features respectively, then pass through an MLP.
- GPS: normalize and featurize via MLP.
- Voxel: to process the $3 \times 3 \times 3$ surrounding voxels, we embed discrete block names to dense vectors, flatten them, and pass through an MLP.
- Past action: our agent is conditioned on its immediate past action, which is embedded and featurized by MLP.

Features from all modalities are concatenated, passed through another fusion MLP, and finally fed into the policy head and value function head. We use an MLP to model the policy head that maps from the input feature vectors to the action probability distribution. We use another MLP to estimate the value function, conditioned on the same input features.

### F.3 RL Training

**PPO.** We use the popular PPO algorithm [102] (Proximal Policy Optimization) as our RL training backbone. PPO is an on-policy method that optimizes for a surrogate objective while ensuring that the deviation from the previous policy is relatively small. PPO updates the policy network by

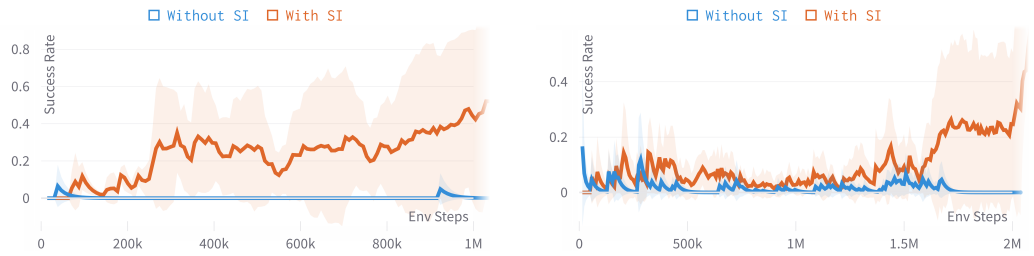
$$\underset{\theta}{\text{maximize}}\ \mathbb{E}_{s,a\sim\pi_{\theta_{\text{old}}}} L(s, a, \theta_{\text{old}}, \theta), \tag{1}$$

where

$$L(s, a, \theta_{\text{old}}, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}}(s, a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_{\text{old}}}}(s, a)\right). \tag{2}$$

$A$ is an estimator of the advantage function (GAE [101] in our case) and $\epsilon$ is a hyperparameter that controls the deviation between the new policy and the old one.

**Self Imitation Learning.** We apply self-imitation learning [84] (SI) to further improve sample efficiency because computing the reward with MINECLIP in the loop makes the training more expensive. Self-imitation learning is essentially supervised learning on a buffer $\mathcal{D}_{SI}$ of good trajectories generated by the agent's past self. In our case, the trajectories are generated by the behavior policy during PPO rollouts, and only added to $\mathcal{D}_{SI}$ if it is a *successful* trial or if the episodic return exceeds a certain threshold. Self imitation optimizes $\pi_\theta$ for the objective $\mathcal{J}_{SI} = \mathbb{E}_{s,a\sim\mathcal{D}_{SI}} \log \pi_\theta(a|s)$ with respect to $\theta$.




(a) "Milk Cow" (b) "Shear Sheep"

Figure A.8: Adding the self imitation technique [84] significantly improves the performance of RL training in MINEDOJO.

We alternate between the PPO phase and the SI phase. A pseudocode of our interleaved training procedure is given in Algorithm 1. We use a *prioritized* strategy to sample trajectories from the buffer $\mathcal{D}_{SI}$. Specifically, we assign equal probability to all successful trajectories. Unsuccessful trajectories can still be sampled but with lower probabilities proportional to their episodic returns.

In Fig. A.8, we demonstrate that adding self-imitation dramatically improves the stability, performance, and sample efficiency of RL training in MINEDOJO.

# G Experiment Details

## G.1 Task Details

We experiment with three task groups with four tasks per group. We train one multi-task agent for each group. In this section, we describe each task goals, initial setup, and the manual dense-shaping reward function.

**Animal Zoo:** 4 Programmatic tasks on hunting or harvesting resource from animals. We spawn various animal types (pig, sheep, and cow) in the same environment to serve as distractors. It is considered a failure if the agent does not take action on the correct animal specified by the prompt.

- **Milk Cow**: find and approach a cow, then obtain milk from it with an empty bucket. The prompt is `milk a cow`. We initialize the agent with an empty bucket to collect milk. We also spawn sheep, cow, and pig nearby the agent. The manual dense reward shaping is a navigation reward based on geodesic distance obtained from privileged LIDAR. The combined reward passed to PPO can be formulated as $r_t = \lambda_{\text{nav}} \max(d_{\text{min},t-1} - d_{\text{min},t}, 0) + \lambda_{\text{success}} \mathbb{1}(\text{milk collected})$, where $\lambda_{\text{nav}} = 10$ and $\lambda_{\text{success}} = 200$. $d_{\text{min},t} = \min(d_{\text{min}}, d_t)$ where $d_{\text{min}}$ denotes the minimal distance to the cow that the agent has achieved so far in the episode history.
- **Hunt Cow**: find and approach a cow, then hunt with a sword. The cow will run away so the agent needs to chase after it. The prompt is `hunt a cow`. We initialize the agent with a diamond sword. The manual dense reward shaping consists of two parts, a valid attack reward and a navigation reward based on geodesic distance obtained from privileged LIDAR. Mathematically, the reward is $r_t = \lambda_{\text{attack}} \mathbb{1}(\text{valid attack}) + \lambda_{\text{nav}} \max(d_{\text{min},t-1} - d_{\text{min},t}, 0) + \lambda_{\text{success}} \mathbb{1}(\text{cow hunted})$, where $\lambda_{\text{attack}} = 5$, $\lambda_{\text{nav}} = 1$, and $\lambda_{\text{success}} = 200$. We additionally reset $d_{\text{min}}$ every time the agent hits the cow to encourage the chasing behavior.
- **Shear Sheep**: find and approach a sheep, then collect wool from the sheep with a shear. The prompt is `shear a sheep`. We initialize the agent with a shear. The manual dense reward shaping is a navigation reward based on geodesic distance obtained from the privileged LIDAR sensor, similar to "Milk Cow".
- **Hunt Sheep**: find and approach a sheep, then hunt with a sword. The sheep will run away so the agent needs to chase after it. An episode will terminate once any entity is hunted. The prompt is `hunt a sheep`. We initialize the agent with a diamond sword. The manual dense

Table A.4: Hyperparameters in RL experiments. "`{state}` MLP" refers to MLPs to process observations of compass, GPS, and voxel blocks. "Embed Dim" denotes the same dimension size used to embed all discrete observations into dense vectors.

| NN Architecture | | Training | | | |
|---|---|---|---|---|---|
| | | Hyperparameter | `Animal-Zoo` | `Mob-Combat` | `Creative` |
| RGB Feature Size | 512 | Learning Rate | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ |
| Task Prompt Feature Size | 512 | Cosine Decay Minimal LR | $5 \times 10^{-6}$ | $5 \times 10^{-6}$ | $5 \times 10^{-6}$ |
| `{state}` MLP Hidden Size | 128 | $\gamma$ | 0.99 | 0.99 | 0.99 |
| `{state}` MLP Output Size | 128 | Entropy Weight (Stage 1) | $5 \times 10^{-3}$ | $5 \times 10^{-3}$ | $5 \times 10^{-3}$ |
| `{state}` MLP Hidden Depth | 2 | Entropy Weight (Stage 2) | $10^{-2}$ | N/A | $10^{-2}$ |
| Embed Dim | 8 | PPO Optimizer | Adam | Adam | Adam |
| Num Feature Fusion Layers | 1 | SI Learning Rate | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ |
| Feature Fusion Output Size | 512 | SI Cosine Decay Minimal LR | $10^{-6}$ | $10^{-6}$ | $10^{-6}$ |
| Prev Action Conditioning | True | SI Epoch | 10 | 10 | 10 |
| Policy Head Hidden Size | 256 | SI Frequency (Env Steps) | 100K | 100K | 100K |
| Policy Head Hidden Depth | 3 | SI Optimizer | Adam | Adam | Adam |
| VF Hidden Size | 256 | SI Buffer Threshold | $2\sigma$ | $2\sigma$ | $0.5\sigma$ |
| VF Hidden Depth | 3 | PPO Buffer Size | 100K | 100K | 100K |
| | | Frame Stack | 1 | 1 | 1 |
| | | VF Loss Weight | 0.5 | 0.5 | 0.5 |
| | | GAE $\lambda$ | 0.95 | 0.95 | 0.95 |
| | | Gradient Clip | 10 | 10 | 10 |
| | | PPO $\epsilon$ | 0.2 | 0.2 | 0.2 |
| | | Action Smooth Weight | $10^{-7}$ | $10^{-7}$ | $10^{-7}$ |
| | | Action Smooth Window Size | 3 | 3 | 3 |
| | | MINECLIP Reward Formulation | DIRECT | DIRECT | DELTA |

reward shaping consists of two parts, a valid attack reward and a navigation reward based on geodesic distance obtained from the privileged LIDAR sensor, similar to "Hunt Cow".

**Mob Combat:** fight 4 different types of hostile monsters: Spider, Zombie, Zombie Pigman (a creature in the *Nether* world), and Enderman (a creature in the *End* world). The prompt template is `"Combat {monster}"`. For all tasks within this group, we initialize the agent with a diamond sword, a shield, and a full suite of diamond armors. The agent is spawned in the *Nether* for Zombie Pigman task, and in the *End* for Enderman. The manual dense-shaping reward can be expressed as $r_t = \lambda_{\text{attack}} \mathbb{1}(\text{valid attack}) + \lambda_{\text{success}} \mathbb{1}(\texttt{\{monster\}}\text{ hunted})$ where $\lambda_{\text{attack}} = 5$ and $\lambda_{\text{success}} = 200$.

**Creative:** 4 tasks that do not have manual dense reward shaping or code-defined success criterion.

- **Find Nether Portal**: find and move close to a Nether Portal, then enter the *Nether* world through the portal. The prompt is `find a nether portal`.
- **Find Ocean**: find and move close to an ocean. The prompt is `find an ocean`.
- **Dig Hole**: dig holes in the ground. The prompt is `dig a hole`. We initialize the agent with an iron shovel.
- **Lay Carpet**: lay down carpets to cover the wooden floor inside a house. The prompt is `put carpets on the floor`. We initialize the agent with a number of carpets in its inventory.

Note that we categorize "Find Nether Portal" and "Find Ocean" as Creative tasks even though they seem similar to object navigation [12]. While finding terrains and other structures is semantically well defined, it is not easy to define a function to evaluate success automatically because the simulator does not have the exact location information of these structures given a randomly generated world. In principle, we can make a sweep by querying each chunk of voxels in the world to recognize the terrains, but that would be prohibitively expensive. Therefore, we opt to use MineCLIP as the reward signal and treat these tasks as Creative.

### G.2 Observation and Action Space

We use a subset of the full observation and action space listed in Sec. B.1 and B.2, because the tasks in our current experiments do not involve actions like crafting or inventory management. Our observation space consists of RGB frame, compass, GPS, and Voxels.

Our action space is a trimmed version of the full action space. It consists of movement control, camera control, "use" action, and "attack" action, which add up to 89 discrete choices. Concretely, it includes 81 actions for discrete camera control ($9 \times 9$ resulted from the Cartesian product between yaw and pitch, each ranges from $-60$ degree to 60 degree with a discrete interval of 15 degree). It also includes 6 movement actions (forward, forward + jump, jump, back, move left, and move right) and 2 functional actions of "use" and "attack". Note that the "no-op" action is merged into the 81 camera actions.

### G.3 RL Training

All hyperparameters used in our RL experiment are listed in Table A.4. We visualize the learned behaviors of 4 tasks in Figure 2. Demos of more tasks can be found on our website `https://minedojo.org`.

**Action smoothing.** Due to the stochastic nature of PPO, we observe a lot of action jittering in the agent's behavior during training. This leads to two negative effects that degrade the learning performance: 1) exploration difficulty due to inconsistent action sequence. For example, the agent may be required to take multiple consecutive `attack` actions in order to complete certain tasks; and 2) rapidly switching different movement and camera motions result in videos that are highly non-smooth and disorienting. This causes a domain gap from the training data of MINECLIP, which are typically smooth human gameplay videos. Therefore, the reward signal quality deteriorates significantly.

To remedy the issue, we impose an action smoothing loss to be jointly optimized with the PPO objective (Eq. 2) during training. Concretely, consider a sliding window $\mathcal{W}$ with window size $|\mathcal{W}|$ that contains $|\mathcal{W}|$ consecutive action distributions $\mathcal{W} = \{\pi_{t-|\mathcal{W}|+1}, \pi_{t-|\mathcal{W}|+2}, \dots, \pi_t\}$, the action smoothing loss is defined as

$$\mathcal{L}_{\text{smooth}} = \frac{1}{|\mathcal{W}|} \sum_{i=1}^{|\mathcal{W}|-1} KL(\pi_t \| \pi_{t-|\mathcal{W}|+i}), \tag{3}$$

where $KL(\cdot)$ denotes Kullback–Leibler divergence.

**Multi-stage training for multi-task RL.** Due to hardware limitations, we are not able to run a large number of parallel simulators for all tasks in a task group. Therefore, we adopt a multi-stage strategy to split the tasks and train them sequentially with a single policy network. For the task groups `Animal-Zoo` and `Creative`, we split the four tasks into two stages of two parallel training tasks each. We carry over the self-imitation buffers when switching to the next stage. We also follow the recommended practice in [83] and reset the policy head at the beginning of stage 2 to encourage exploration and reduce overfitting. We adopt a similar replay buffer balancing strategy as [46] to prevent any task from dominating the training.

### G.4 Evaluation

In this section, we elaborate on our human and automatic evaluation procedure for Creative tasks. We first ask the human annotators to manually label 100 successful and 100 failure trajectories. This produces a combined dataset of 200 trajectories with groundtruth binary labels to evaluate the learned reward functions. On this dataset, we run MINECLIP to produce step-wise rewards and compute a score that averages over each trajectory. We then apply K-means clustering with $K = 2$ to all scores and determine a decision boundary $\delta$ from the mean of the two centroids. A trajectory with a score greater than $\delta$ is classified as successful, and vice versa for failure. In this way, we essentially convert MINECLIP to a binary classifier. The quality of MINECLIP can be measured by the F1 score of its binary classification output against the human labels. We demonstrate that MINECLIP has high agreements with humans (Table 2), and thus qualifies as an effective automatic evaluation metric for Creative tasks in the absence of human judges.

To further investigate MINECLIP's evaluation on more complex Creative tasks, we annotate 50 YouTube video segments each for 5 more tasks that are much more semantically complex: "build a farm", " build a fence", "build a house", "ride a minecart", and "build a swimming pool". We then run MINECLIP evaluation on these videos against a negative set. As shown in Table A.5, though not perfect, MINECLIP generally has a positive agreement with human judgment. We note

Table A.5: MINECLIP's evaluation on more complex Creative tasks. Numbers represent F1 scores between MINECLIP's evaluation on tasks success and human labels. Scaled to percentage for better readability.

| Tasks | Build a Farm | Build a Fence | Build a House | Ride a Minecart | Build a Swimming Pool |
|---|---|---|---|---|---|
| Ours (Attn) | **78.7** | **91.4** | **63.7** | 95.9 | 85.0 |
| Ours (Avg) | 73.4 | 83.1 | 37.4 | **96.9** | **94.7** |
| $\text{CLIP}_{\text{OpenAI}}$ | 62.5 | 24.5 | 52.9 | 70.0 | 71.7 |

that the current MINECLIP is a proof-of-concept step in leveraging internet data for automated evaluation, and further scaling on more training data and parameters may lead to more improvements. Meanwhile, human judgment remains a useful and important alternative [93, 97].

# H Limitations and Potential Societal Impact

Unlike human demonstrations [126] or offline RL datasets [35], our YouTube dataset contains only the video screen observations but not the actual control actions. This allows us to scale up the dataset tremendously, but at the same time poses a challenge to imitation learning algorithms that require observation-action pairs to learn. Our proposed algorithm, MINECLIP, side-steps this problem by learning a reward model, but we believe that directly inferring the human expert policy from YouTube is another important direction complementary to our approach. There are promising techniques that can potentially overcome this limitation, such as the Learning-from-Observation (LfO) family of algorithms [123, 122, 115, 31].

Our database is scraped from the internet, which inevitably contains offensive YouTube videos or toxic Reddit posts. While we have made our best effort to filter out these harmful contents (Sec. D.1), there can still be undesirable biases and toxicity that elude our automatic filters. Furthermore, we advocate the use of large pre-trained language models in our main paper, and MINECLIP is finetuned from the pre-trained weights of OpenAI CLIP [92]. These foundation models are known to contain harmful stereotypes and generate hateful commentary [15, 13, 40]. We ask the researchers who will use our code and database to exercise their best judgment during new model development to avoid any negative social impact.

# I Datasheet

We present a Datasheet [39] for documentation and responsible usage of our internet knowledge databases.

## I.1 Motivation

**For what purpose was the dataset created?** We create this internet-scale multimodal knowledge base to facilitate research towards open-ended, generally capable embodied agents.

**Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?** This knowledge base was created by Linxi Fan (Nvidia), Guanzhi Wang (Caltech), Yunfan Jiang (Stanford), Ajay Mandlekar (Nvidia), Yuncong Yang (Columbia), Haoyi Zhu (SJTU), Andrew Tang (Columbia), De-An Huang (Nvidia), Yuke Zhu (Nvidia and UT Austin), and Anima Anandkumar (Nvidia and Caltech).

## I.2 Distribution

**Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created?** Yes, the dataset is publicly available on the internet.

**How will the dataset will be distributed (e.g., tarball on website, API, GitHub)?** All datasets can be downloaded from `https://zenodo.org/`. Please refer to this table of URL, DOI, and licensing:

| Database | DOI | License |
|---|---|---|
| YouTube | `10.5281/zenodo.6641142` | Creative Commons Attribution 4.0 International (CC BY 4.0) |
| Wiki | `10.5281/zenodo.6640448` | Creative Commons Attribution Non Commercial Share Alike 3.0 Unported |
| Reddit | `10.5281/zenodo.6641114` | Creative Commons Attribution 4.0 International (CC BY 4.0) |

**Have any third parties imposed IP-based or other restrictions on the data associated with the instances?** No.

**Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?** No.

### I.3 Maintenance

**Who will be supporting/hosting/maintaining the dataset?** The authors will be supporting, hosting, and maintaining the dataset.

**How can the owner/curator/manager of the dataset be contacted (e.g., email address)?** Please contact Linxi Fan (`linxif@nvidia.com`), Guanzhi Wang (`guanzhi@caltech.edu`), and Yunfan Jiang (`yunfanj@cs.stanford.edu`).

**Is there an erratum?** No. We will make announcements if there is any.

**Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?** Yes. New updates will be posted on `https://minedojo.org`.

**If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were the individuals in question told that their data would be retained for a fixed period of time and then deleted)?** N/A.

**Will older versions of the dataset continue to be supported/hosted/maintained?** Yes, old versions will be permanently accessible on zenodo.org.

**If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?** Yes, please refer to `https://minedojo.org`.

### I.4 Composition

**What do the instances that comprise the dataset represent?** For YouTube videos, our data is in JSON format with video URLs and metadata. We do not provide the raw MP4 files for legal concerns. For Wiki, we provide the text, images, tables, and diagrams embedded on the web pages. For Reddit, our data is in JSON format with post IDs and metadata, similar to YouTube. Users can reconstruct the Reddit dataset by running our script after obtaining an official Reddit API license key.

**How many instances are there in total (of each type, if appropriate)?** There are more than 730K YouTube videos with 2.2B words of transcripts, 6,735 Wiki pages with 2.2M bounding boxes of visual elements, and more than 340K Reddit posts with 6.6M comments.

**Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?** We provide all instances in our Zenodo data repositories.

**Is there a label or target associated with each instance?** No.

**Is any information missing from individual instances?** No.

**Are relationships between individual instances made explicit (e.g., users' movie ratings, social network links)?** We provide metadata for each YouTube video link and Reddit post ID.

**Are there recommended data splits (e.g., training, development/validation, testing)?** No. The entire database is intended for pre-training.

**Are there any errors, sources of noise, or redundancies in the dataset?** Please refer to Sec. D

**Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)?** We follow prior works [64] and only release the video URLs of YouTube videos due to legal concerns. Researchers need to acquire the MP4 and transcript files separately. Similarly, we only release the post IDs for the Minecraft Reddit database, but we also provide a script that can reconstruct the full Reddit dataset given a free official license key.

**Does the dataset contain data that might be considered confidential?** No.

**Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?** We have made our best efforts to detoxify the contents via an automated procedure. Please refer to Sec. D.

### I.5 Collection Process

The collection procedure, preprocessing, and cleaning are explained in details in Sec. D.

**Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)?** All data collection, curation, and filtering are done by MINEDOJO coauthors.

**Over what timeframe was the data collected?** The data was collected between Dec. 2021 and May 2022.

### I.6 Uses

**Has the dataset been used for any tasks already?** Yes, we have used the MINEDOJO YouTube database for agent pre-training. Please refer to Sec. 5 and Sec. G for algorithmic and training details.

**What (other) tasks could the dataset be used for?** Our knowledge base is primarily intended to facilitate research in open-ended, generally capable embodied agents. However, it can also be broadly applicable to research in video understanding, document understanding, language modeling, multimodal learning, and so on.

**Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?** No.

**Are there tasks for which the dataset should not be used?** We strongly oppose any research that intentionally generates harmful or toxic contents using our YouTube, Wiki, and Reddit data.