MINEDOJO: Supplementary Material

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`

1 Overview

Our supplementary includes the following material:

1. This document (`supplementary.pdf`): we present the Minecraft framework feature comparison table, MINEDOJO simulator specification, task curation process, internet data collection process (YouTube, Wiki, and Reddit), algorithm and experimental details of MINECLIP and RL agents. We also discuss potential limitations and societal impact at the end.
2. `datasheet.pdf`: we follow the Datasheet format [14] to explain the composition, collection, recommended use case, and other details of our databases. The document also includes **Author Statement**, **Licensing**, and **Maintenance Plan**.

Following NeurIPS Dataset and Benchmark track guidelines, we have uploaded our data to `zenodo.org` for DOI and structured metadata:

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)

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

2 Minecraft Framework Comparison

Table 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 [20]	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) [2]	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 [35]	Cooperative and competitive multiagent tasks; parameterizable environments	✓	14			
Malmo [25]	First comprehensive release of a Gym-style agent API for Minecraft	✓	N/A			
CraftAssist [17]	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 [28]	Interactive dialogues with humans; aimed at building structures described by natural language		157	✓		
EvoCraft [18]	Aimed at generating creative artifacts; allows for direction manipulation of blocks		N/A			
Crafter [21]	2D clone of Minecraft; fast experimentation		22		Human experts dataset	100 episodes

3 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.

3.1 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 (main paper, Sec. 6). Amounts and directions of LIDAR rays can be arbitrarily configured at the cost of a lower simulation throughput.

Table 2: Schema of YAML files for Programmatic and Creative tasks.

	Programmatic Tasks	Creative Tasks
Prompt	Natural language description of the task goal.	
Guidance	Step-by-step guidance generated by GPT-3.	
Category	`{survival, harvest, tech-tree, combat}`	N/A
Collection	N/A	`{manual, youtube, gpt3}`
Source	N/A	If the task is mined from YouTube, include the URL, start, and end timestamp

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.

3.2 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 [49]. 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

3.3 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.

4 MINEDOJO Task Suite

In this section, we explain how we collect the Programmatic and Creative tasks (main paper, Sec. 3).

4.1 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. 1 for a few samples. Table 2 lists the fields in the specification file for Programmatic tasks, which we include as `programmatic_tasks.yaml` in the supplementary material zip. 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.

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.

4.2 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. Table 2 shows the specifications of Creative tasks. Interested readers can refer to `creative_tasks.yaml` for a full listing.

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. 5.1) 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;

Stage 3: For the accepted tasks in stage 2, we design a labeling UI using Label Studio [30] 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. 2). Through this pipeline, we extract 1,042 task ideas from the common wisdom of a

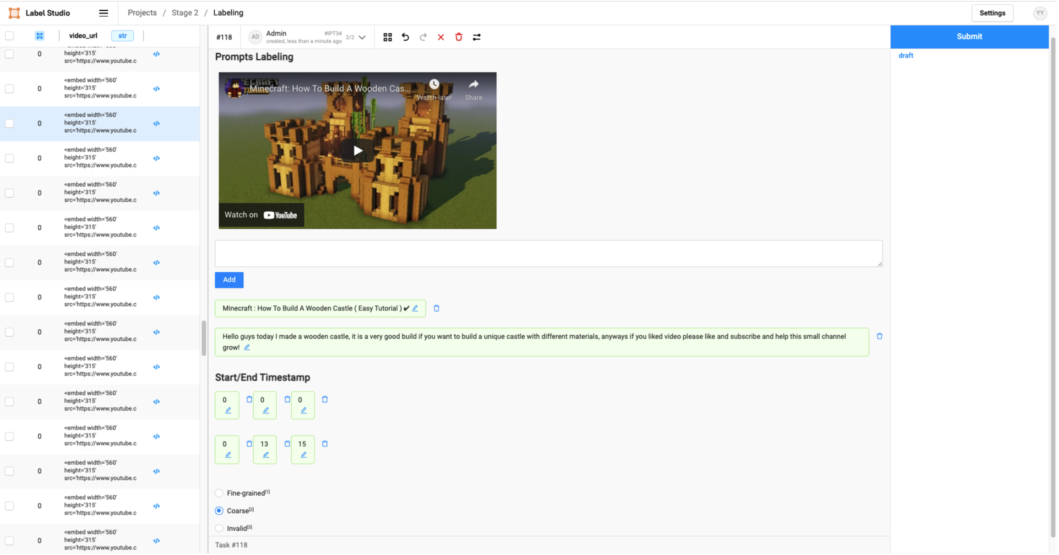
```
survival_sword_food:
  category: survival
  prompt: survive as long as possible given a sword and some food

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 1: Example specifications. Please refer to `programmatic_tasks.yaml` in the supplementary for a complete listing.



Minecraft : How To Build A Wooden Castle (Easy Tutorial) ✔

Hello guys today I made a wooden castle, it is a very good build if you want to build a unique castle with different materials, anyways if you liked video please like and subscribe and help this small channel grow!

0 0 0

0 13 15

○ Fine-grained[1]
● Coarse[2]
○ Invalid[3]

Figure 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.

huge number of veteran Minecraft gamers. Some examples are "*make an automated mining machine*" and "*grow cactus up to the sky*".

4.3 GPT-3 Guidance

We leverage OpenAI's `GPT-3-davinci` API to automatically generate detailed guidance for a subset of the tasks. Inspired by [29], 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;`

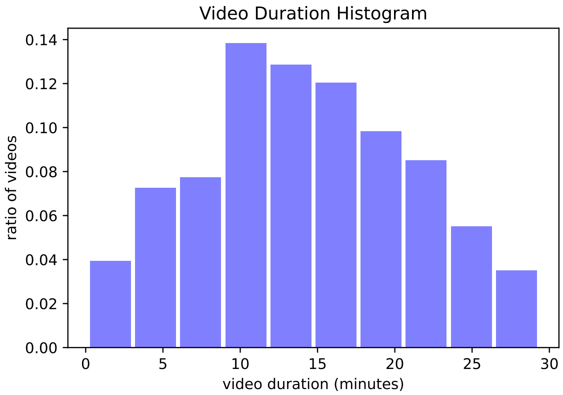


Figure 3: Distribution of YouTube video duration. The histogram is trimmed by the 85th percentile to hide much longer videos that can run for many hours.

`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.`

4.4 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.

"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.

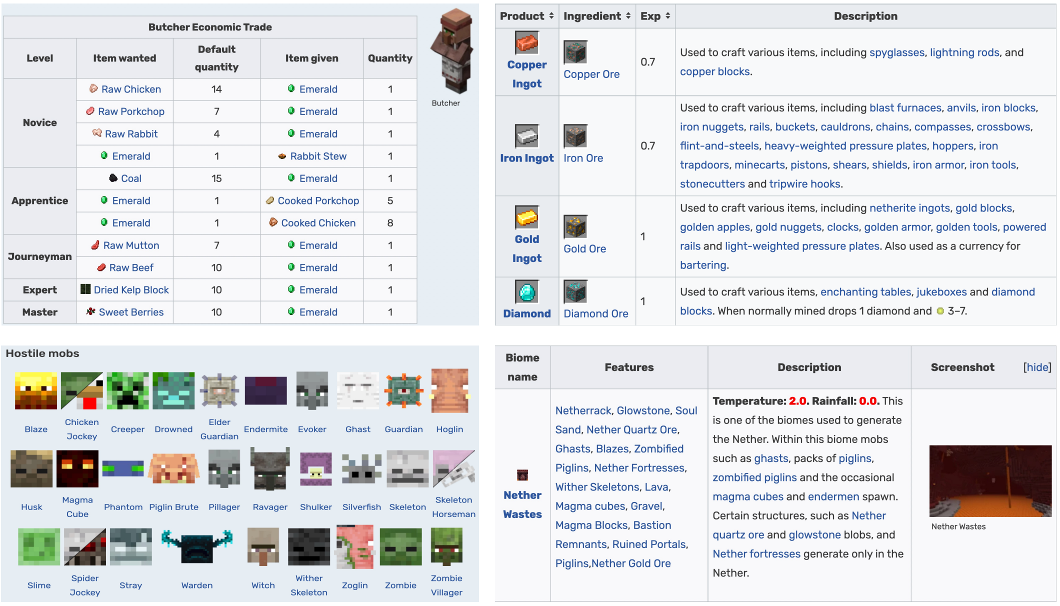
5 Internet-Scale Database

We upload our databases to `zenodo.org`, which is an open repository platform operated by CERN. The persistent DOIs are listed in Sec. 1. In this section, we describe our database properties and data collection process in details.

5.1 YouTube Videos and Transcripts

Minecraft is among the most streamed games on YouTube [16]. 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. 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 [56] to collect our database, following the procedure below:



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

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.

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

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;

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.

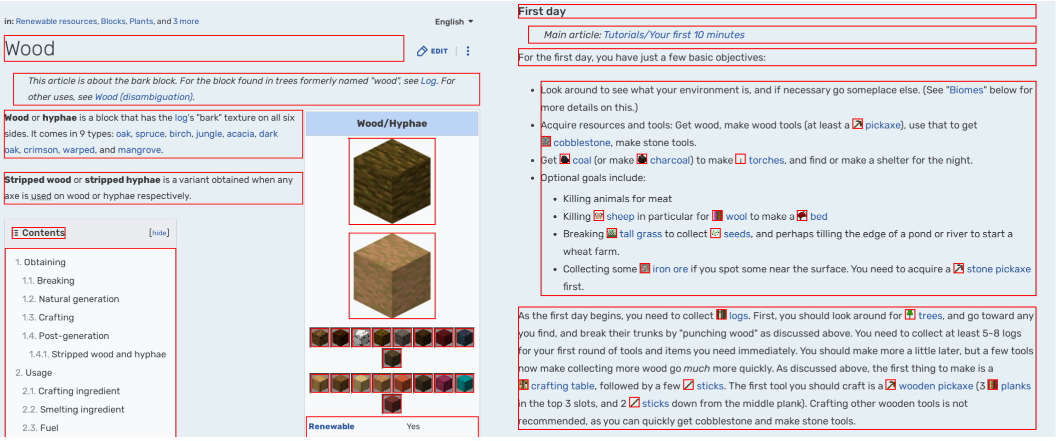
e) To further clean the dataset and remove potentially harmful contents, we employ the Detoxify [22] tool to process each video title and description. Detoxify is trained on Wikipedia comments to predict multiple types of toxicity like threats, obscenity, insults, and identity-based hate speech. We delete 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 practices [26], we do not release the actual MP4 files and transcripts due to legal concerns.

5.2 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. 4 and Fig. 5). We use Selenium [44] 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.



in: Renewable resources, Blocks, Plants, and 3 more

English

Wood

EDIT

This article is about the bark block. For the block found in trees formerly named "wood", see Log. For other uses, see Wood (disambiguation).

Wood or **hyphae** is a block that has the log's "bark" texture on all six sides. It comes in 9 types: oak, spruce, birch, jungle, acacia, dark oak, crimson, warped, and mangrove.

Stripped wood or **stripped hyphae** is a variant obtained when any axe is used on wood or hyphae respectively.

Wood/Hyphae

Renewable	Yes

Contents [hide]

1. Obtaining
   - 1.1. Breaking
   - 1.2. Natural generation
   - 1.3. Crafting
   - 1.4. Post-generation
     - 1.4.1. Stripped wood and hyphae
2. Usage
   - 2.1. Crafting ingredient
   - 2.2. Smelting ingredient
   - 2.3. Fuel

First day

Main article: Tutorials/Your first 10 minutes

For the first day, you have just a few basic objectives:

- Look around to see what your environment is, and if necessary go someplace else. (See "Biomes" below for more details on this.)
- Acquire resources and tools: Get wood, make wood tools (at least a pickaxe), use that to get cobblestone, make stone tools.
- Get coal (or make charcoal) to make torches, and find or make a shelter for the night.
- Optional goals include:
  - Killing animals for meat
  - Killing sheep in particular for wool to make a bed
  - Breaking tall grass to collect seeds, and perhaps tilling the edge of a pond or river to start a wheat farm.
  - Collecting some iron ore if you spot some near the surface. You need to acquire a stone pickaxe first.

As the first day begins, you need to collect logs. First, you should look around for trees, and go toward any you find, and break their trunks by "punching wood" as discussed above. You need to collect at least 5-8 logs for your first round of tools and items you need immediately. You should make more a little later, but a few tools now make collecting more wood go *much* more quickly. As discussed above, the first thing to make is a crafting table, followed by a few sticks. The first tool you should craft is a wooden pickaxe (3 planks in the top 3 slots, and 2 sticks down from the middle plank). Crafting other wooden tools is not recommended, as you can quickly get cobblestone and make stone tools.

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

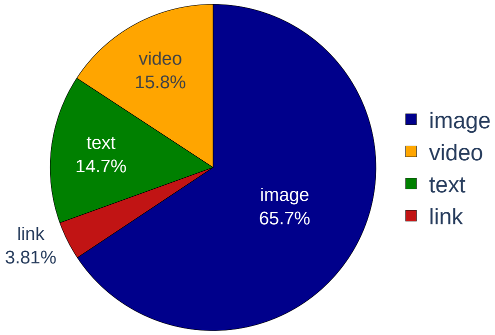


Figure 6: Distribution of Reddit post types.

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.

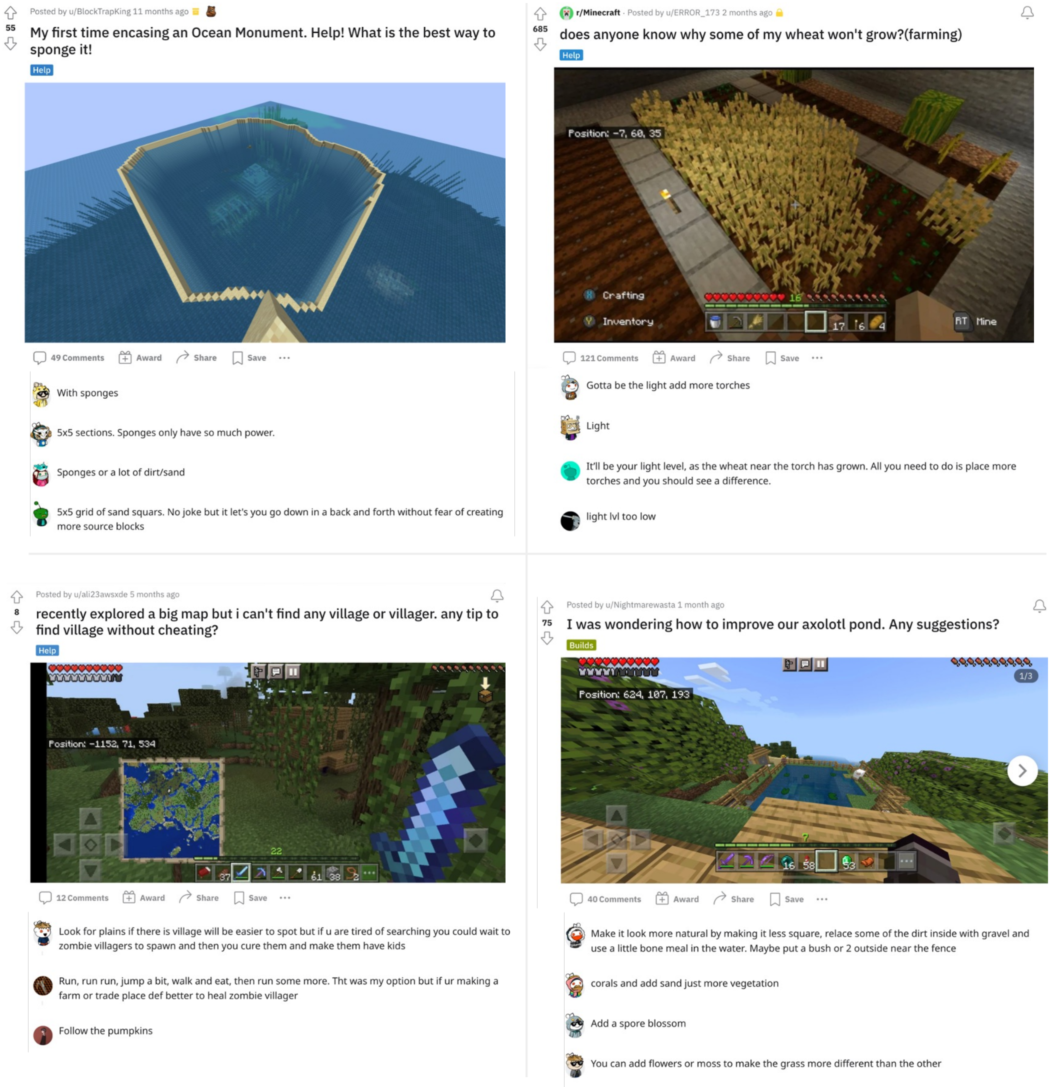
d) **Sprites.** Sprite elements are micro-sized image icons that are typically embedded in text to create multimodal tutorials and explanations. We save all the sprites and locate their bounding boxes within the text too.

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.

5.3 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 [1, 55, 24] and abstractive summarization [50, 27]. The r/Minecraft subreddit contains free-form discussions of game strategies and images/videos showcases of Minecraft builds and creations (examples in Fig. 7). The distribution of post types is shown in Fig. 6.

To scrape the Reddit contents, we use PRAW [36], a Python wrapper on top of the official Reddit API. Our procedure is as follows:



55
Posted by u/BlockTrapKing 11 months ago
My first time encasing an Ocean Monument. Help! What is the best way to sponge it!

With sponges

5x5 sections. Sponges only have so much power.

Sponges or a lot of dirt/sand

5x5 grid of sand squars. No joke but it let's you go down in a back and forth without fear of creating more source blocks

685
r/Minecraft · Posted by u/ERROR_173 2 months ago
does anyone know why some of my wheat won't grow?(farming)

Gotta be the light add more torches

Light

It'll be your light level, as the wheat near the torch has grown. All you need to do is place more torches and you should see a difference.

light lvl too low

8
Posted by u/ali23awsxde 5 months ago
recently explored a big map but i can't find any village or villager. any tip to find village without cheating?

Look for plains if there is village will be easier to spot but if u are tired of searching you could wait to zombie villagers to spawn and then you cure them and make them have kids

Run, run run, jump a bit, walk and eat, then run some more. Tht was my option but if ur making a farm or trade place def better to heal zombie villager

Follow the pumpkins

75
Posted by u/Nightmarewasta 1 month ago
I was wondering how to improve our axolotl pond. Any suggestions?

Make it look more natural by making it less square, relace some of the dirt inside with gravel and use a little bone meal in the water. Maybe put a bush or 2 outside near the fence

corals and add sand just more vegetation

Add a spore blossom

You can add flowers or moss to make the grass more different than the other

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

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) ≥ 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.

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. 5.1), we run Detoxify [22] 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.

6 MINECLIP Algorithm Details

We implement all our neural networks in PyTorch v1.11 [10]. Training MINECLIP uses the PyTorch-Lightning framework [10], pre-trained models hosted on HuggingFace [52], and the `x-transformers` library for Transformer variants [51].

6.1 Video-Text Pair Extraction

Similar to VideoCLIP [54], 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.

6.2 Architecture

MINECLIP architecture is composed of three parts:

Frame-wise image encoder ϕ_I We use the ViT-B/16 architecture [8] to compute a 512-D embedding for each RGB frame. We initialize the weights from OpenAI CLIP's public checkpoint [39] and only finetune the last two layers during training. The input resolution is 160×256, which is different from CLIP's default 224×224 resolution. We adapt the positional embeddings via bicubic interpolation, which does not introduce any new learnable parameters.

Temporal aggregator ϕ_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 [13]. 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 ϕ_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 [45, 7]. The transformer sequence encoder is relatively slower, but captures more temporal information and achieves better performance in our experiments (main paper, Table 1).

Text encoder ϕ_G We use a 12-layer 512-wide GPT model with 8 attention heads [37, 38]. 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 ϕ_G is finetuned during training.

6.3 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 [6, 11] as data augmentation. We use Cosine learning rate annealing with 500 gradient steps of warming up [31]. We apply a lower learning rate ($\times 0.5$) on the pre-trained weights and layer-wise learning rate decay for better finetuning [23]. Training is performed on 1 node of $8\times$ V100 GPUs with FP16 mixed precision [32] via the PyTorch native `amp` module. All hyperparameters are listed in Table 3.

Table 3: Training hyperparameters for MineCLIP.

Hyperparameter	Value
LR schedule	Cosine with warmup [31]
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	160×256
Number of frames	16
Image encoder	`ViT-B/16` [8]

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 $\mathbf{D}_{SI} \leftarrow \{\emptyset, \forall T \in$ training tasks$\}$;
Initialize a counter for simulator steps $counter \leftarrow 0$;
while not done do
    Collect set of trajectories for all tasks $\{\tau_T, \forall T \in$ 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 $\#_{\text{sample}} = \min(\{|\mathcal{D}_{SI,T}|, \forall T \in$ 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
```

7 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.

7.1 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 4.

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

Table 4: Ablation on different MINECLIP reward formulations.

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

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 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.

7.2 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. 3.1), the feature extractor contains several modality-specific components:

- RGB frame: we use the frozen frame-wise image encoder ϕ_I in MINECLIP to optimize for compute efficiency and provide the agent with good visual representations from the beginning (Sec. 5.2 in main paper).
- Task goal: ϕ_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.

7.3 RL Training

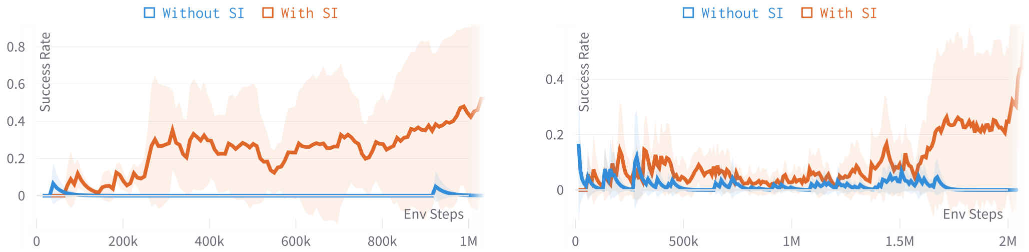
PPO. We use the popular PPO algorithm [43] (Proximal Policy Optimization) as our RL training backbone. PPO is an on-policy method that optimizes for a surrogate objective while ensuring that the deviation from the previous policy is relatively small. PPO updates the policy network by

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

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 [42] in our case) and ϵ is a hyperparameter that controls the deviation between the new policy and the old one.



(a) “Milk Cow”

(b) “Shear Sheep”

Figure 8: Adding the self imitation technique [34] significantly improves the performance of RL training in MINEDOJO.

Self Imitation Learning. We apply self-imitation learning [34] (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 π_θ for the objective $\mathcal{J}_{SI} = \mathbb{E}_{s,a\sim\mathcal{D}_{SI}} \log \pi_\theta(a|s)$ with respect to θ.

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

8 Experiment Details

8.1 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_{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_{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

Table 5: 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	5×10^{-6}	5×10^{-6}	5×10^{-6}
`{state}` MLP Hidden Size	128	γ	0.99	0.99	0.99
`{state}` MLP Output Size	128	Entropy Weight (Stage 1)	5×10^{-3}	5×10^{-3}	5×10^{-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	2σ	2σ	0.5σ
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 λ	0.95	0.95	0.95
		Gradient Clip	10	10	10
		PPO ϵ	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

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 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 [3]. 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.

8.2 Observation and Action Space

We use a subset of the full observation and action space listed in Sec. 3.1 and 3.2, 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 (9×9 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.

8.3 RL Training

All hyperparameters used in our RL experiment are listed in Table 5. 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 [33] 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 [19] to prevent any task from dominating the training.

8.4 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 δ from the mean of the two centroids. A trajectory with a score greater than δ 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 in main paper), and thus qualifies as an effective automatic evaluation metric for Creative tasks in the absence of human judges.

Table 6: 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	**91.5 ± 0.7**	64.5 ± 37.1	↑
	Hunt Cow	46.8 ± 3.7	**83.5 ± 7.1**	↓
	Shear Sheep	**73.5 ± 0.8**	12.1 ± 9.1	↑
	Hunt Sheep	**27.0 ± 1.0**	8.1 ± 4.1	↑
	Combat Spider	72.1 ± 1.3	**80.5 ± 13.0**	↓
	Combat Zombie	27.1 ± 2.7	**47.3 ± 10.6**	↓
	Combat Pigman	**6.5 ± 1.2**	1.6 ± 2.3	↑
	Combat Enderman	0.0 ± 0.0	0.0 ± 0.0	=
	Find Nether Portal	**99.1 ± 0.4**	37.4 ± 40.8	↑
	Find Ocean	**95.1 ± 1.5**	33.4 ± 45.6	↑
	Dig Hole	85.8 ± 1.2	**91.6 ± 5.9**	↓
	Lay Carpet	96.5 ± 0.8	**97.6 ± 1.9**	=

Table 7: 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 ± 0.6	**46.0 ± 15.3**	0.0 ± 0.0
	Harvest Spider String	1.6 ± 1.3	**36.5 ± 16.9**	12.5 ± 12.7

9 More Experiments and Ablations

9.1 Learning a Single Agent for All 12 Tasks

We have 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 7.3). We summarize the result in Table 6. 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 [57, 53] 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.

9.2 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 7 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.

Table 8: 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

9.3 MINECLIP's Evaluation on 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 8, though not perfect, MINECLIP generally has a positive agreement with human judgment. We note 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 [40, 41].

10 Limitations and Potential Societal Impact

Unlike human demonstrations [49] or offline RL datasets [12], 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 [48, 47, 46, 9].

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. 5.1), 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 [39]. These foundation models are known to contain harmful stereotypes and generate hateful commentary [5, 4, 15]. 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.

11 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] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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`.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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`.

[15] 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.

[16] 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/`.

[17] 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.

[18] 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`.

[19] 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`.

[20] 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.

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

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

[23] 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.

[24] 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.

[25] 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`.

[26] 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.

[27] 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`.

[28] 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.

[29] 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.

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

[31] 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`.

[32] 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`.

[33] 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.

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

[35] 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.

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

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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`.

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

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

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

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

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

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

[49] 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.

[50] 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`.

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

[52] 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.

[53] 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`.

[54] 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.

[55] 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`.

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

[57] 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.