MineRL

Release 0.4.0

William H. Guss, Brandon Houghton

Aug 23, 2023

TUTORIALS AND GUIDES

Welcome to documentation for the MineRL project and its related repositories and components!

CHAPTER

ONE

WHAT IS MINERL

MineRL is a rich Python 3 library which provides a OpenAI Gym interface for interacting with the video game Minecraft, accompanied with datasets of human gameplay.

Started as a research project at Carnegie Mellon University, MineRL aims to assist in the develpment of various aspects of artificial intelligence within Minecraft.

1.1 Installation

Welcome to MineRL! This guide will get you started.

To start using the MineRL dataset and Gym environments comprising MineRL, you'll need to install the main python package, `minerl`.

1. First **make sure you have JDK 8** installed on your system.
   a. Windows installer – On windows go this link and follow the instructions to install JDK 8. Install x64 version.
   b. On Mac, you can install Java 8 using homebrew and AdoptOpenJDK (an open source mirror, used here to get around the fact that Java8 binaries are no longer available directly from Oracle). If you encounter errors installing MineRL, try these steps for Mac:

```
brew tap AdoptOpenJDK/openjdk
brew install --cask adoptopenjdk8
```

   c. On Debian based systems (Ubuntu!) you can run the following:

```
sudo add-apt-repository ppa:openjdk-r/ppa
sudo apt-get update
sudo apt-get install openjdk-8-jdk

# Verify installation
java -version # this should output "1.8.X_XXX"
# If you are still seeing a wrong Java version, you may use
# the following line to update it
# sudo update-alternatives --config java
```

2. If you are using Windows, you will also need `bash` command. The best way to do is to install Windows Subsystem for Linux (WSL. Tested on WSL 2). Note that installing MineRL may seem especially slow/stuck, but it is not; it is just a bit slow. You can also install MineRL on the WSL system itself, but you may need `xvfb` to run the environment. Note that you need to install correct Java version on the WSL, too!
3. Now install the `minerl` package!:

```
pip install git+https://github.com/minerllabs/minerl
```

Note: You may need the user flag: `pip install git+https://github.com/minerllabs/minerl --user` to install properly.

1.2 Hello World: Your First Agent

With the `minerl` package installed on your system you can now make your first agent in Minecraft!

To get started, let's first import the necessary packages

```
import gym
import minerl
```

1.2.1 Creating an environment

Now we can choose any one of the many environments included in the `minerl` package. To learn more about the environments checkout the environment documentation.

For this tutorial we'll choose the `MineRLBasaltFindCave-v0` environment. In this task, the agent is placed to a new world and its (subjective) goal is to find a cave, and end the episode.

To create the environment, simply invoke `gym.make`

```
env = gym.make('MineRLBasaltFindCave-v0')
```

Caution: Currently `minerl` only supports environment rendering in **headed environments** (servers with monitors attached).

In order to run `minerl` environments without a head use a software renderer such as `xvfb`:

```
xvfb-run python3 <your_script.py>
```

Note: If you're worried and want to make sure something is happening behind the scenes install a logger **before** you create the envrionment.

```
import logging
logging.basicConfig(level=logging.DEBUG)

env = gym.make('MineRLBasaltFindCave-v0')
```

1.2.2 Taking actions

As a warm up let's create a random agent.

Now we can reset this environment to its first position and get our first observation from the agent by resetting the environment.

```
# Note that this command will launch the MineRL environment, which takes time.
# Be patient!
obs = env.reset()
```

The `obs` variable will be a dictionary containing the following observations returned by the environment. In the case of the `MineRLBasaltFindCave-v0` environment, only one observation is returned: `pov`, an RGB image of the agent's first person perspective.

```
{
    'pov': array([[[ 63,  63,  68],
        [ 63,  63,  68],
        [ 63,  63,  68],
        ...,
        [ 92,  92, 100],
        [ 92,  92, 100],
        [ 92,  92, 100]],,

        ...,


        [[ 95, 118, 176],
        [ 95, 119, 177],
        [ 96, 119, 178],
        ...,
        [ 93, 116, 172],
        [ 93, 115, 171],
        [ 92, 115, 170]]], dtype=uint8)
}
```

Now let's take actions through the environment until time runs out or the agent dies. To do this, we will use the normal OpenAI Gym `env.step` method.

```
done = False

while not done:
    # Take a random action
    action = env.action_space.sample()
    # In BASALT environments, sending ESC action will end the episode
    # Lets not do that
    action["ESC"] = 0
    obs, reward, done, _ = env.step(action)
    env.render()
```

With the `env.render` call, you should see the agent move sporadically until `done` flag is set to true, which will happen when agent runs out of time (3 minutes in the FindCave task).

1.3 More Tutorials

Here we aim to collect links to projects done by us and others, which can be helpful for you getting started.

- MineRL in Colab
- OpenAI VPT in Colab
- MineRL Diamond in Colab
- Tutorials on MineRL BASALT (by mdda)
- First steps with MineRL (by mdda)
- Installing MineRL using containers (by mdda)

1.4 General Information

The `minerl` package includes several environments as follows. This page describes each of the included environments, provides usage samples, and describes the exact action and observation space provided by each environment!

Note: All environments offer a default no-op action via `env.action_space.no_op()` and a random action via `env.action_space.sample()`.

1.4.1 Observation Space

Most environments use the same observation space (just an RGB image):

```
Dict(pov:Box(low=0, high=255, shape=(360, 640, 3)))
```

1.4.2 Action Space

Most environments use the same action space, which is a dictionary containing a multitude of different actions. Note that `Discrete` and `Box` are actions spaces defined by Gym.

```
Dict({
    "ESC": "Discrete(2)",
    "attack": "Discrete(2)",
    "back": "Discrete(2)",
    "camera": "Box(low=-180.0, high=180.0, shape=(2,))",
    "drop": "Discrete(2)",
    "forward": "Discrete(2)",
    "hotbar.1": "Discrete(2)",
    "hotbar.2": "Discrete(2)",
    "hotbar.3": "Discrete(2)",
    "hotbar.4": "Discrete(2)",
    "hotbar.5": "Discrete(2)",
    "hotbar.6": "Discrete(2)",
    "hotbar.7": "Discrete(2)",
    "hotbar.8": "Discrete(2)",
    "hotbar.9": "Discrete(2)",
    "inventory": "Discrete(2)",
```

```
    "jump": "Discrete(2)",
    "left": "Discrete(2)",
    "pickItem": "Discrete(2)",
    "right": "Discrete(2)",
    "sneak": "Discrete(2)",
    "sprint": "Discrete(2)",
    "swapHands": "Discrete(2)",
    "use": "Discrete(2)"
})
```

Here is an example action:

```
{"ESC":0, "camera":[10, 45], "swapHands":1}
```

ESC

The ESC action may be used in some environments to end the episode (e.g., BASALT environments). Otherwise it does nothing.

inventory

The `inventory` opens the inventory GUI. This will yield an observation image something like the following:

camera

This action changes the orientation of the agent's heading by the corresponding number of degrees. The head changes its orientation pitch by the first component and its yaw by the second component. Both components are limited to [-180, 180] inclusive.

pickItem

When an agent looks at a block and executes this action, if that block type is in the agents inventory, it will be put it into the agent's main hand.

swapHands

Swaps the items in agent main hand and secondary slot.

1.5 MineRL BASALT Competition Environments

Warning: The videos on this page represent the v0.4 dataset, not the v1.0 dataset.

In the Benchmark for Agents that Solve Almost-Lifelike Task (BASALT) competition, your task is to solve tasks based on human judgement, instead of pre-defined reward functions. The goal is to produce agents that are judged by real humans to be effective at solving a given task. This calls for training on human-feedback, whether it is training from demonstrations, training on human preferences or using humans to correct agents' actions.

1.5.1 MineRLBasaltFindCave-v0

After spawning in a plains biome, explore and find a cave. When inside a cave, end the episode by setting the "ESC" action to 1.

You are not allowed to dig down from the surface to find a cave.

Starting Inventory

```
Dict({    })
```

Max Episode Steps

3600

Usage

```
env = gym.make("MineRLBasaltFindCave-v0")
```

1.5.2 MineRLBasaltCreateVillageAnimalPen-v0

After spawning in a village, build an animal pen next to one of the houses in a village. Use your fence posts to build one animal pen that contains at least two of the same animal. (You are only allowed to pen chickens, cows, pigs, or sheep.) There should be at least one gate that allows players to enter and exit easily. The animal pen should not contain more than one type of animal. (You may kill any extra types of animals that accidentally got into the pen.)

Do not harm villagers or existing village structures in the process.

Send 1 for "ESC" key to end the episode.

Starting Inventory

```
Dict({
    "carrot": 1,
    "oak_fence": 64,
    "oak_fence_gate": 64,
    "wheat": 1,
    "wheat_seeds": 1
})
```

Max Episode Steps

6000

Usage

```
env = gym.make("MineRLBasaltCreateVillageAnimalPen-v0")
```

1.5.3 MineRLBasaltMakeWaterfall-v0

After spawning in an extreme hills biome, use your waterbucket to make a beautiful waterfall. Then take an aesthetic "picture" of it by moving to a good location, positioning player's camera to have a nice view of the waterfall, and ending the episode by setting "ESC" action to 1.

Starting Inventory

```
Dict({
    "cobblestone": 20,
    "stone_pickaxe": 1,
    "stone_shovel": 1,
    "water_bucket": 1
})
```

Max Episode Steps

```
6000
```

Usage

```
env = gym.make("MineRLBasaltMakeWaterfall-v0")
```

1.5.4 MineRLBasaltBuildVillageHouse-v0

Build a house in the style of the village without damaging the village. It should be in an appropriate location (e.g. next to the path through the village) Then, give a brief tour of the house (i.e. spin around slowly such that all of the walls and the roof are visible). Finally, end the episode by setting the "ESC" action to 1.

Tip: You can find detailed information on which materials are used in each biome-specific village (plains, savannah, taiga, desert) here: https://minecraft.fandom.com/wiki/Village/Structure/Blueprints

---

Starting Inventory

```
Dict({
    "acacia_log": 64,
    "black_dye": 64,
    "blue_dye": 64,
    "brown_dye": 64,
    "cactus": 64,
    "cobblestone": 64,
    "cobweb": 64,
    "dirt": 64,
    "flower_pot": 64,
    "glass_pane": 64,
    "grass_block": 64,
    "green_dye": 64,
    "jungle_log": 64,
    "lantern": 64,
    "oak_log": 64,
    "packed_ice": 64,
    "poppy": 64,
    "red_dye": 64,
    "sand": 64,
    "sandstone": 64,
    "smooth_sandstone": 64,
    "snow_block": 64,
    "spruce_log": 64,
    "stone_axe": 1,
    "stone_pickaxe": 1,
    "terracotta": 64,
    "torch": 64,
    "white_dye": 64,
    "white_wool": 64,
    "yellow_dye": 64
})
```

Max Episode Steps

```
14400
```

Usage

```
env = gym.make("MineRLBasaltBuildVillageHouse-v0")
```

1.6 MineRL Obtain Diamond Environments

The goal of these environments is to obtain diamonds and build items from them.

Note: ESC is not used in Diamond environments

Note: Diamond environments also receive direct inventory observations (e.g. `"apple":5` if the agent has five apples)

1.6.1 MineRLObtainDiamondShovel-v0

In this environment the agent is required to obtain a diamond shovel. The agent begins in a random starting location on a random survival map without any items, matching the normal starting conditions for human players in Minecraft.

During an episode the agent is rewarded according to the requisite item hierarchy needed to obtain a diamond shovel. The rewards for each item are given here:

```
<Item reward="1" type="log" />
<Item reward="2" type="planks" />
<Item reward="4" type="stick" />
<Item reward="4" type="crafting_table" />
<Item reward="8" type="wooden_pickaxe" />
<Item reward="16" type="cobblestone" />
<Item reward="32" type="furnace" />
<Item reward="32" type="stone_pickaxe" />
<Item reward="64" type="iron_ore" />
<Item reward="128" type="iron_ingot" />
<Item reward="256" type="iron_pickaxe" />
<Item reward="1024" type="diamond" />
<Item reward="2048" type="diamond_shovel" />
```

Max Episode Steps

18000

Usage

```
env = gym.make("MineRLObtainDiamondShovel-v0")
```

1.7 Performance tips

1.7.1 Faster alternative to xvfb

Running MineRL on xvfb will slow it down by 2-3x as the rendering is done on CPU, not on the GPU. A potential alternative is to use a combination of VirtualGL and virtual displays from nvidia tools.

Note that this may interfere with your display/driver setup, and may not work on cloud VMs (`nvidia-xconfig` is not available).

Following commands outline the procedure. You may need to adapt it to suit your needs. After these commands, run `export DISPLAY=:0` and you should be ready to run MineRL. The Minecraft window will be rendered in a virtual display.

All credits go to Tencent researchers who kindly shared this piece of information!

```
sudo apt install lightdm libglu1-mesa mesa-utils xvfb xinit xserver-xorg-video-dummy

sudo nvidia-xconfig -a --allow-empty-initial-configuration --virtual=1920x1200 --busid␣
↪PCI:0:8:0
cd /tmp
wget https://nchc.dl.sourceforge.net/project/virtualgl/2.6.3/virtualgl_2.6.3_amd64.deb
sudo dpkg -i virtualgl_2.6.3_amd64.deb

sudo service lightdm stop
sudo vglserver_config
sudo service lightdm start
```

1.7.2 Docker images for headless rendering with GPU

The above instructions might not work with a server without root access. You may use this docker file (alternatives can be found here and here instead).

To begin with, build & run this docker on your server.

```
git clone https://github.com/jeasinema/egl-docker && cd egl-docker
docker build . -t <docker_name>
docker run --gpus all -it <docker_name>:latest /bin/bash
```

Inside the container, use the following command to verify if the GPU rendering is working. If you can see something like `OpenGL Renderer: NVIDIA GeForce RTX 3090/PCIe/SSE2`, congratulations. Otherwise output like `OpenGL Renderer: llvmpipe (LLVM 12.0.0, 256 bits)` indicates you're still using CPU. Feel free to post to this repo if you have any issues.

```
vglrun /opt/VirtualGL/bin/glxspheres64
```

You're good to go! Just prepend your commands with `vglrun` to enable GPU rendering.

Acknowledgements: This docker image is brought to you by Xiaojian Ma and the MineDoJo team, and it is developed upon this project by Seungmin Kim.

1.7.3 Singularity container for headless rendering with GPU

There is also a Singularity container based on the docker image above here. All the information on how to build and run the container are specified inside this github repo, plus some things you will have to take into consideration before being able to use it.

Acknowledgements: This singularity container is brought you by David Sanfelix.

1.8 Links to papers and projects

Here you can find useful links to the presentations, code and papers of the finalists in previous MineRL competitions, as well as other publications and projects that use MineRL.

To see all papers that cite MineRL, check Google Scholar. You can also create alerts there to get notified whenever a new citation appears.

If you want to add your paper/project here, do not hesitate to create a pull request in the main repository!

1.8.1 Presentations

- MineRL 2019 - Finalists presentations at NeurIPS 2019
- MineRL 2019 - 1st place winners presentation, longer one (slides in English, talk in Russian)
- MineRL 2020 - Round 1 finalists presentations at NeurIPS 2020
- MineRL 2020 - Round 2 finalists presentations at Microsoft AI and Gaming Research Summit 2021

1.8.2 MineRL papers

- MineRL: A Large-Scale Dataset of Minecraft Demonstrations
- The MineRL 2019 Competition on Sample Efficient Reinforcement Learning using Human Priors
- Retrospective Analysis of the 2019 MineRL Competition on Sample Efficient Reinforcement Learning
- The MineRL 2020 Competition on Sample Efficient Reinforcement Learning using Human Priors
- Towards robust and domain agnostic reinforcement learning competitions: MineRL 2020

1.8.3 2019 competitor code/papers

- 1st place: paper.
- 2nd place: paper, code.
- 3rd place: paper, code.
- 4th place: code.
- 5th place: paper, code.

1.8.4 2020 competitor code/papers

- 1st place: paper.
- 2nd place: code.
- 3rd place: code.

1.8.5 Other papers that use the MineRL environment

- PiCoEDL: Discovery and Learning of Minecraft Navigation Goals from Pixels and Coordinates (CVPR Embodied AI Workshop, 2021)
- Universal Value Iteration Networks: When Spatially-Invariant Is Not Universal (AAAI, 2020)
- Multi-task curriculum learning in a complex, visual, hard-exploration domain: Minecraft
- Follow up paper from the #1 team in 2019 (obtains diamond): paper, code.
- Align-RUDDER: Learning From Few Demonstrations by Reward Redistribution (obtains diamond): paper, code.
- IGLU: Interactive Grounded Language Understanding in a Collaborative Environment: paper (NEURIPS, 2022)

1.8.6 Other

- Data analysis for vector obfuscation/kmeans
- Malmo and MineRL tutorial

1.9 General FAQ

1.9.1 For Version 1.x

Why does this MineRL version take so long to install?

Previous versions compiled the game binary when `reset` gets called. v1.0.0 compiles the binary during install, so MineRL wont have to do so on `reset` calls. This makes `reset` faster. Also see the Windows FAQ for slow install info.

Failed to initialize GLFW or GLX problems

This can occur when attempting to run on a headless system without using something like xvfb.

Try `xvfb-run -a python [path to your code]`

When trying to run MineRL, why do I get Java or JDK related errors?

Make sure you are using the correct JDK version for MineRL (must be Java JDK 8, the x64 version) On Windows, the best option may be to remove all Javas from machine with the uninstall utility, and then install JDK 8 from the Oracle website.

1.9.2 For Version 0.4.x

When I run MineRL, a tiny window pops up and I cant see what my agent is doing. Is something wrong?

No, this is the proper way that MineRL runs. Try useing `env.render()` if you need to watch your agent.

Why is MineRL giving timeout errors or agents with `Connection timed out!` errors?

If a MineRL Window is not :code:`step`ed within X seconds, it will automatically crash. This is to prevent MineRL from hanging if Minecraft stops working properly.

Why do MineRL windows sometimes just crash?

Unfortunately, there are bugs in Minecraft which sometimes cause crashes :(

1.10 Windows FAQ

This note serves as a collection of fixes for errors which may occur on the Windows platform.

1.10.1 The `The system cannot find the path specified` error (installing)

If during installation you get errors regarding missing files or unspecified paths, followed by a long path string, you might be limited by the `MAX_PATH` setting on windows. Try removing this limitation with these instructions.

1.10.2 The `freeze_support` error (multiprocessing)

```
RuntimeError:
       An attempt has been made to start a new process before the
       current process has finished its bootstrapping phase.

   This probably means that you are not using fork to start your
   child processes and you have forgotten to use the proper idiom
   in the main module:

       if __name__ == '__main__':
           freeze_support()
           ...
```

(continues on next page)

(continued from previous page)

```
  The "freeze_support()" line can be omitted if the program
  is not going to be frozen to produce an executable.
```

The implementation of `multiprocessing` is different on Windows, which uses `spawn` instead of `fork`. So we have to wrap the code with an if-clause to protect the code from executing multiple times. Refactor your code into the following structure.

```
import minerl
import gym

def main()
    # do your main minerl code
    env = gym.make('MineRLTreechop-v0')

if __name__ == '__main__':
    main()
```

1.11 MineRL Versions

Over time, there have been a few major MineRL versions. Here is a summary of their functionalities.

Version #	Competitions used in	Details	Install commands
0.3.7	MineRL 2019 and 2020	[Minecraft: 1.11.2] Fastest version, but does not have the BASALT environments	`pip install minerl==0.3.7`
0.4.4	MineRL Diamond and BASALT 2021	[Minecraft: 1.11.2] ~30% slower than 0.3.7, but has the BASALT environments	`pip install minerl==0.4.4`
1.0.0	BASALT 2022	[Minecraft: 1.16.5] Slower than previous versions, but has more realistic action spaces (e.g. no autocrafting) and Minecraft Nether Update version.	`pip install git+https://github.com/minerllabs/minerl@v1.0.0`

1.12 Other Minecraft Interfaces

This page lists and compares a number of different Minecraft Interfaces other than MineRL.

	Created	Dependency	Maintained	Competition	Unified obs/action spaces
Project Malmo	2016	Minecraft	No	No	No
MarLÖ	2018	Malmo	No	Formerly	No
MalmoEnv	~2018/2019	Malmo	No	No	No
MineRL	2019	[1.0.0 Minecraft] [<1.0.0 MalmoEnv]	Yes	Yes	[1.0.0 Mostly] [<1.0.0 No]
IGLU 2021	2021	MineRL	?	Yes	?
MineDojo	2022	MineRL 0.4.4	Just Released	No	Yes

1.12.1 Project Malmo

Original Minecraft Java platform/mod built by Microsoft for AI research. Many other Minecraft libraries depend on Malmo.

1.12.2 MarLÖ "Multi-Agent Reinforcement Learning in MalmÖ"

A Python library which provides a gym-like interface for interacting with Malmo/Minecraft, specifically for multi-agent scenarios.

1.12.3 MalmoEnv

A Python library which provides a gym-like interface for interacting with Malmo/Minecraft.

1.12.4 IGLU "Interactive Grounded Language Understanding in a Collaborative Environment"

A Python library for building interactive agents for natural language grounding tasks. Note: the 2022 competition depends on a faster Minecraft Clone, while the 2021 competition depends on MineRL.

1.12.5 MineDojo

A framework built on MineRL, which "features a simulation suite with 1000s of open-ended and language-prompted tasks".

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search