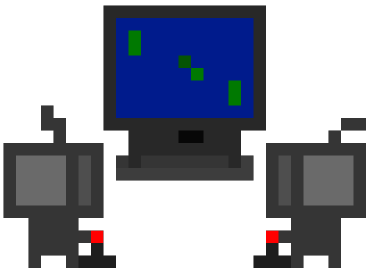
piSTAR Lab

Release 0.0.1.dev0

Brandyn Kusenda

Sep 22, 2021

CONTENTS



piSTAR Lab is an open source platform built to make AI experimentation accessible and fun.

- Github: http://github.com/pistarlab/pistarlab

Note: piSTAR Lab is in **early development**.

CHAPTER

ONE

WHO IS PISTAR LAB FOR?

- **AI Enthusiasts and Students:**
  - Have fun experimenting with state-of-the-art AI Agents. No coding required.
  - Gain a better intuition about reinforcement learning by creating your own agents.
- **Reinforcement Learning Researchers and Engineers:**
  - Showcase your algorithms by creating extensions for other users to enjoy. No restrictive framework/library requirements.
  - Import your externally trained agents.
  - Use piSTAR Lab in your development workflow.
  - Try your algorithm on newly available environments with just a few clicks
- **Game and Environment Developers:**
  - Create Extensions" which make your game playable by piSTAR Agents.
  - Train your own Agents for your game's AI

CHAPTER

TWO

FEATURES

- Intuitive UI
- Extension System for adding new agents, environments or tasks types
- No framework requirements. Use any framework available in python (other languages coming)
- Real-time streaming of observations during training
- Single and Multi player environment support
- Experiment tracking
- Python API, anything you can do in the UI, you can do in Python as well
- Uses Ray Project (https://ray.io/) under the hood for distributed processing

CHAPTER

THREE

COMING SOON (UNORDERED)

- Simpler installation and better support for Windows and Mac
- Easily share your trained agents with friends or publicly
- Docker based agents and environments
- Remote agents and environments
- Data environments for agents with support for learning from offline data
- API for Data Interfaces
- Support for browser based games and environments so human can interact directly with their agents.
- Support for other programming languages
- Human control mode for testing environments via the UI
- Better workspace integration with VSCode and Jupyter
- Interactive Debugger
- Testing and quality verification of extensions

CHAPTER

FOUR

CONTRIBUTING

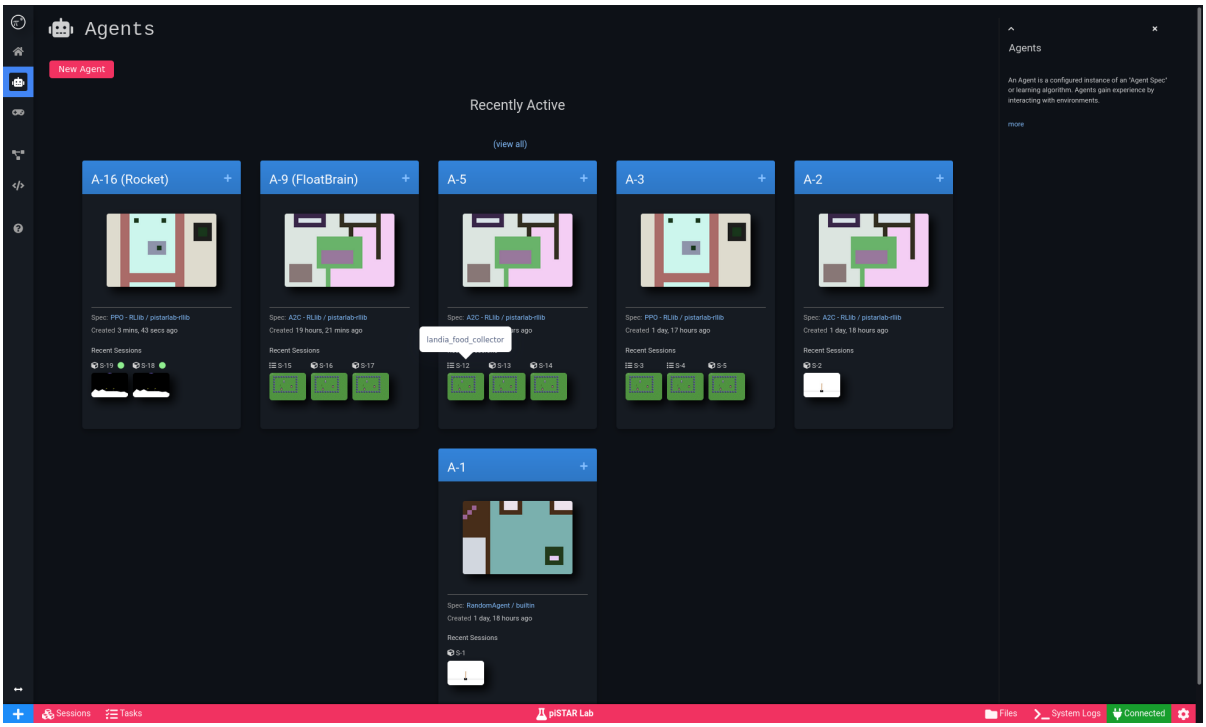
piSTAR Lab is **Open Source** and welcomes your contributions.

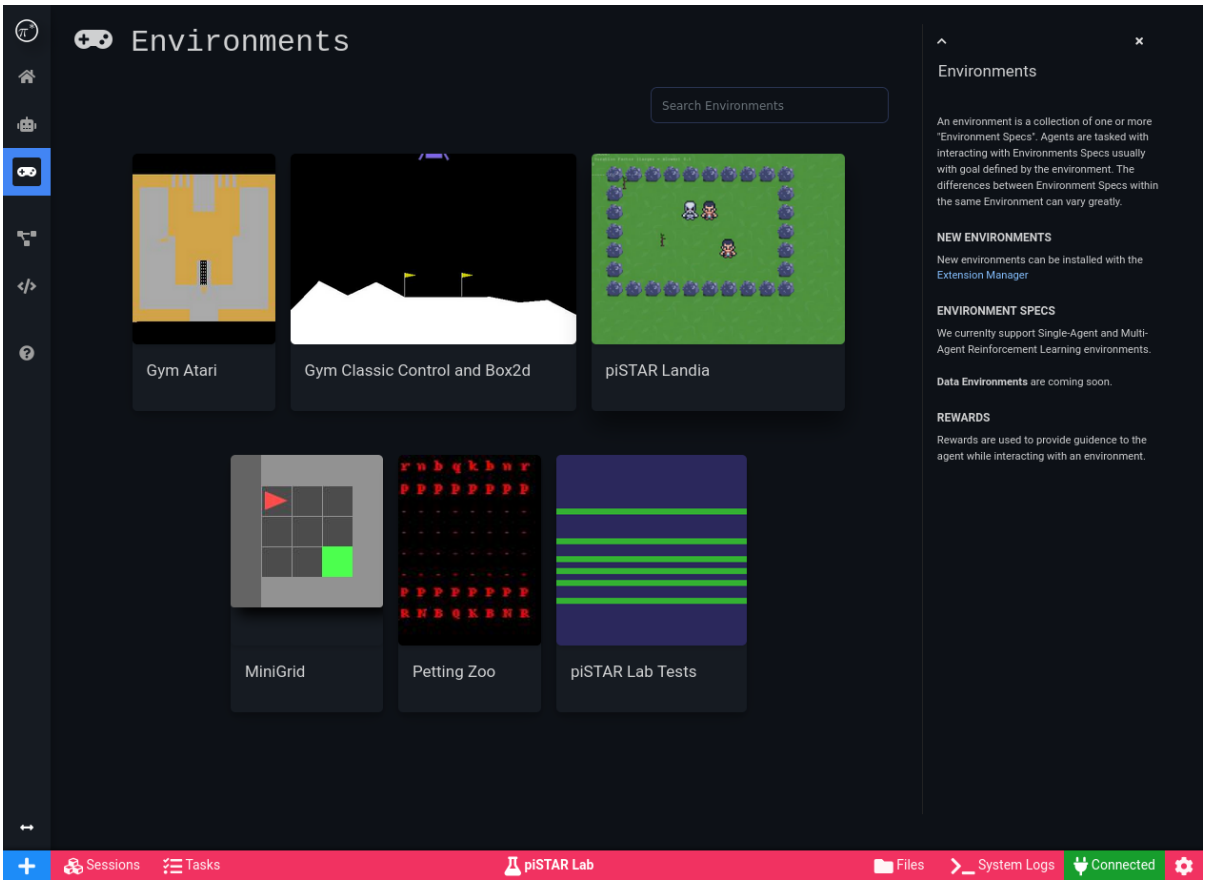
4.1 Overview

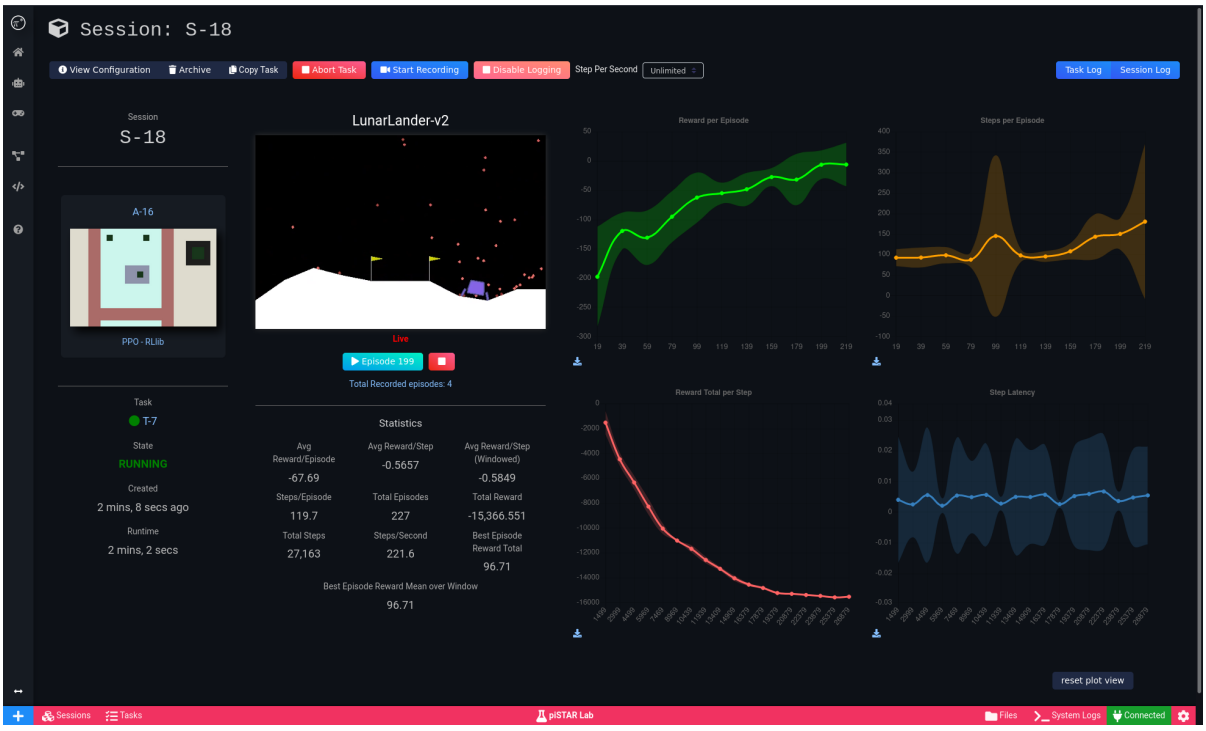
piSTAR Lab is modular Agent Development Environment that can run on your PC or in the cloud.

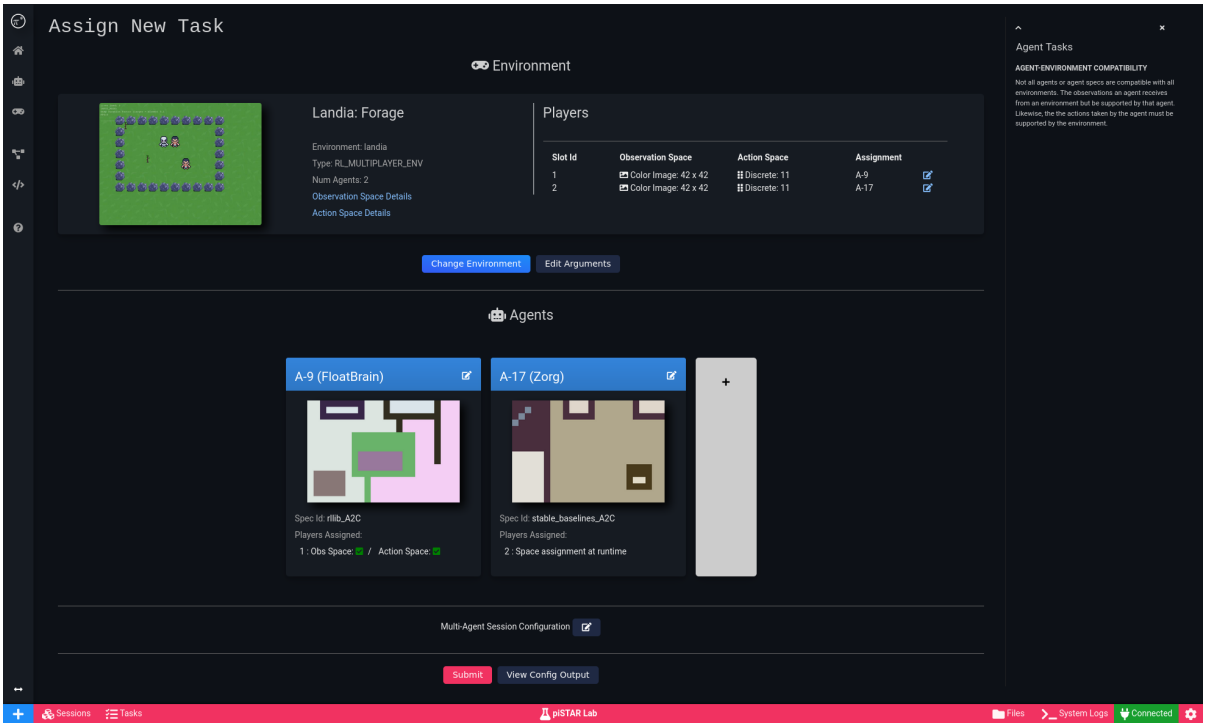
- Manage your own library of Agents and Environments.
- Run experiments with a click of a button.
- Watch your agent learn in realtime.

4.2 Screenshots

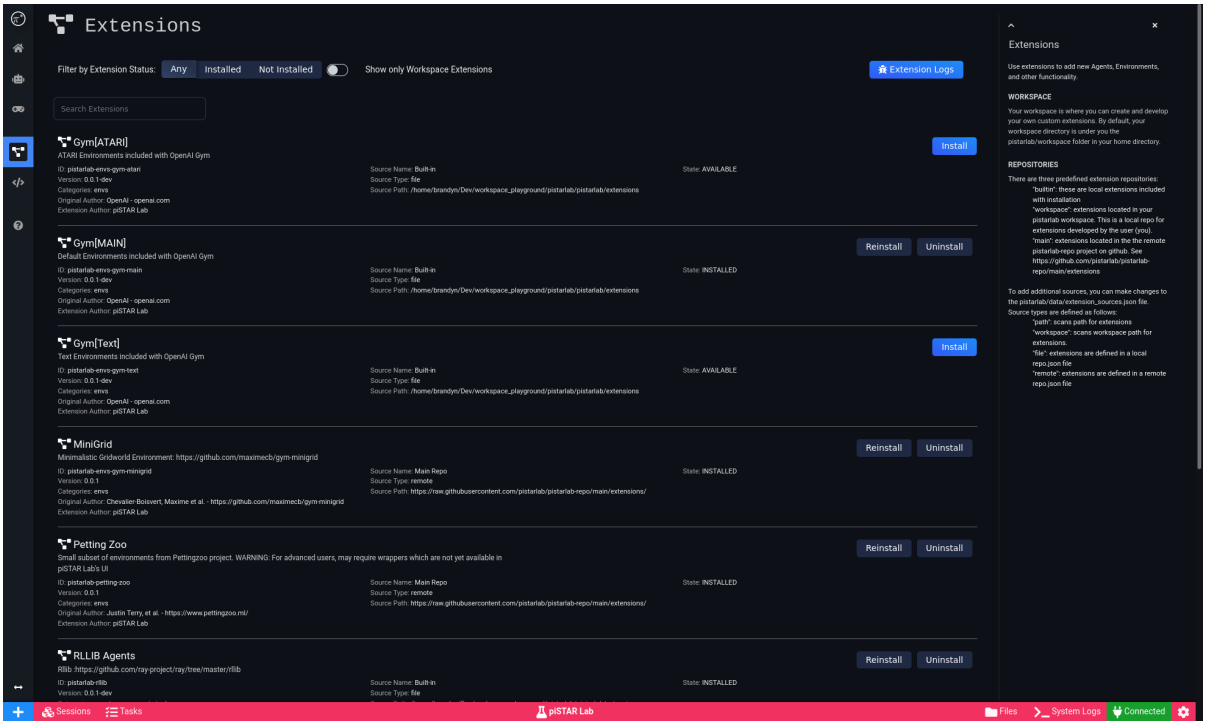



Environments
Search Environments
Gym Atari
Gym Classic Control and Box2d
piSTAR Landia
MiniGrid
Petting Zoo
piSTAR Lab Tests
Environments
An environment is a collection of one or more "Environment Specs". Agents are tasked with interacting with Environments Specs usually with goal defined by the environment. The differences between Environment Specs within the same Environment can vary greatly.
NEW ENVIRONMENTS
New environments can be installed with the Extension Manager
ENVIRONMENT SPECS
We currenlty support Single-Agent and Multi-Agent Reinforcement Learning environments.
Data Environments are coming soon.
REWARDS
Rewards are used to provide guidence to the agent while interacting with an environment.
Sessions
Tasks
piSTAR Lab
Files
System Logs
Connected


Session: S-18
View Configuration
Archive
Copy Task
Abort Task
Start Recording
Disable Logging
Step Per Second
Unlimited
Task Log
Session Log
Session
S-18
A-16
PPO - RLlib
Task
T-7
State
RUNNING
Created
2 mins, 8 secs ago
Runtime
2 mins, 2 secs
LunarLander-v2
Live
Episode 199
Total Recorded episodes: 4
Statistics
Avg Reward/Episode
-67.69
Avg Reward/Step
-0.5657
Avg Reward/Step (Windowed)
-0.5849
Steps/Episode
119.7
Total Episodes
227
Total Reward
-15,366.551
Total Steps
27,163
Steps/Second
221.6
Best Episode Reward Total
96.71
Best Episode Reward Mean over Window
96.71
Reward per Episode
Steps per Episode
Reward Total per Step
Step Latency
reset plot view
Sessions
Tasks
piSTAR Lab
Files
System Logs
Connected


Assign New Task
Environment
Landia: Forage
Environment: landia
Type: RL_MULTIPLAYER_ENV
Num Agents: 2
Observation Space Details
Action Space Details
Players
Slot Id
Observation Space
Action Space
Assignment
1
Color Image: 42 x 42
Discrete: 11
A-9
2
Color Image: 42 x 42
Discrete: 11
A-17
Change Environment
Edit Arguments
Agents
A-9 (FloatBrain)
A-17 (Zorg)
Spec Id: rllib_A2C
Players Assigned:
1 : Obs Space: / Action Space:
Spec Id: stable_baselines_A2C
Players Assigned:
2 : Space assignment at runtime
Multi-Agent Session Configuration
Submit
View Config Output
Agent Tasks
AGENT-ENVIRONMENT COMPATIBILITY
Not all agents or agent specs are compatible with all environments. The observations an agent receives from an environment but be supported by that agent. Likewise, the the actions taken by the agent must be supported by the environment.
Sessions
Tasks
piSTAR Lab
Files
System Logs
Connected


Multi-Agent Session: S-22
Copy Task
View Configuration
Abort Task
Start Recording
Disable Logging
Step Per Second
Unlimited
Task Log
Session Log
Session Id
S-22
Task Id
T-11
State
RUNNING
Creation Time
2021-08-26T00:02:37
Runtime (seconds)
0 secs
Landia: Forage
Live
Agent Sessions
SESSION ID
AGENT
TOTAL STEPS
EPISODE COUNT
S-23
A-1 (RandomAgent)
1925
21
S-24
A-9 (rllib_A2C)
4383
21
Statistics
Avg Reward/Episode
-30.00
Avg Reward/Step
-0.1435
Avg Reward/Step (Windowed)
-0.3689
Steps/Episode
209.0
Completed Episodes
21
Total Reward
-630
Total Steps
4389
Steps/Second
103.9
ep_stats - episode_reward_total
ep_stats - episode_step_count
step_stats - reward_total
step_stats - step_latency_mean_windowed
Sessions
Tasks
piSTAR Lab
Files
System Logs
Connected



4.3 Installation

NOTE: Tested on Ubuntu and Windows 10. If you run into issues, we recommend using Docker.

4.3.1 with Pip

1. Install Anaconda or Miniconda

   Visit https://www.anaconda.com/products/individual for instructions

2. **Install additional dependencies (Ubuntu only)**

   - XVFB to render without display (No MS Windows Support)
   - ffmpeg for video processing

```
sudo apt-get install -y xvfb ffmpeg
```

3. Create Conda Virtual Environment

```
conda create -n pistarlab python=3.7
conda activate pistarlab
conda install pip
```

4. Option 1 (Package only)

```
pip install https://github.com/pistarlab/pistarlab/archive/refs/heads/main.zip
↪#egg=pistarlab[all]
```

5. Option 2 (Package + Source)

```
git clone  --single-branch --depth=1 http://github.com/pistarlab/pistarlab/
cd pistarlab
pip install -e .[all]
```

4.3.2 with Docker

1. **Install Docker:** Visit: https://docs.docker.com/engine/install/
2. Clone Repo

```
docker pull docker.pkg.github.com/pistarlab/pistarlab/image:latest
```

4.4 Usage

To launch piSTAR Lab, run:

```
pistarlab_launcher
```

Open your browser to http://localhost:7777

To launch the piSTAR Lab docker container, run:

```
mkdir pistarlab_docker ; docker run --rm -i -t -u `id -u` \
    --shm-size=2g \
    -p 7776:7776  \
    -p 7777:7777  \
    -p 7778:7778  \
    -p 8265:8265 \
    -v ${HOME}/pistarlab_docker:/home/ray/pistarlab pistarlab/pistarlab:latest  --
↪launcher_host="0.0.0.0"
```

Open your browser to http://localhost:7777

4.5 Extensions

Extensions are the primary mechanism for adding new Agents, Environments, Tasks and Components.

Below are extensions available via piSTAR Lab and the repo (https://github.com/pistarlab/pistarlab-repo/). Future versions will allow user created extensions to be used and shared as well.

- **Gym Atari Environments:**
  - Extension: https://github.com/pistarlab/pistarlab/tree/main/pistarlab/extensions/pistarlab-envs-gym-atari
  - Project Page: https://gym.openai.com/envs/#atari
- **Gym Text Environments:**
  - Extension: https://github.com/pistarlab/pistarlab/tree/main/pistarlab/extensions/pistarlab-envs-gym-text
  - Project Page: https://gym.openai.com/envs/#text

- **Gym 2D Box and Classic Control Environments:**
  - Extension: https://github.com/pistarlab/pistarlab/tree/main/pistarlab/extensions/pistarlab-envs-gym-main
  - Project Pages: https://gym.openai.com/envs/#box2d, https://gym.openai.com/envs/#classic_control
- **piSTAR Landia Environment**
  - Extension: https://github.com/pistarlab/pistarlab-repo/tree/main/src/pistarlab-landia
  - Project Page: https://github.com/pistarlab/landia/
- **Ray RLlib Agents**
  - Extension: https://github.com/pistarlab/pistarlab/tree/main/pistarlab/extensions/pistarlab-rllib
  - Project Page: https://docs.ray.io/en/latest/rllib.html
  - Note: extension only includes a subset of agent algorithms
- **MiniGrid Envirnoments:**
  - Extension: https://github.com/pistarlab/pistarlab-repo/tree/main/src/pistarlab-envs-gym-minigrid
  - Project Page: https://github.com/maximecb/gym-minigrid
- **Petting Zoo Environments (WIP)**
  - Extension: https://github.com/pistarlab/pistarlab-repo/tree/main/src/pistarlab-petting-zoo
  - Project Page: https://www.pettingzoo.ml/
  - Note: extension only includes a small subset of environments
- **Stable Baselines Agents (WIP)**
  - Extension: https://github.com/pistarlab/pistarlab-repo/tree/main/src/pistarlab-stable-baselines
  - Project Page:
  - Note: extension only includes a small subset of agents available and currently does not support parameterization
- **Unity ML-Agent Environments (Planned)**
  - Extension: https://github.com/pistarlab/pistarlab-repo/tree/main/src/pistarlab-unity-envs
  - Project Page: https://github.com/Unity-Technologies/ml-agents
  - Note: currently unavailable in extension repo
- **Minecraft RL Environments (Planned)**
  - Extension: https://github.com/pistarlab/pistarlab-repo/tree/main/src/pistarlab-envs-minecraft
  - Project Page: https://github.com/minerllabs/minerl
  - Note: currently unavailable in extension repo
- **MultiGrid MARL Environments (Planned)**
  - Extension: https://github.com/pistarlab/pistarlab-repo/tree/main/src/pistarlab-multigrid
  - Project Page: https://github.com/ArnaudFickinger/gym-multigrid
  - Note: currently unavailable in extension repo

4.5.1 Development Notes

Creating a manifest

Manfiest files are used to speed up the installation of Extensions. They are especially useful for Environments where probing is required.

Example of creating a manifest

```
xvfb-run python pistarlab/extensions_tools.py --action=save_manifest --extension_path␣
↪PATH_TO_EXTENSION/pistarlab-envs-gym-main
```

4.6 Advanced

4.6.1 Settings

Data and Configuration Path

By default pistarlab stores data and configuration in the **$HOME/pistarlab/** directory. (eg: /home/$USER/pistarlab) This can be changed by using the **PISTARLAB_ROOT** environment variable

If you are using *Docker*, PISTARLAB_ROOT is under **$HOME/pistarlab_docker/**. This has been changed for Docker to avoid permission issues that Docker may instroduce.

Configuration is stored in the **config.yaml** file located at PISTARLAB_ROOT

Data is stored under PISTARLAB_ROOT/data

4.6.2 Debugging

Our current solution uses the remote_pdb python module to enable remote debuging over telnet.

To use debugging, do the following:

1. Insert the following code snip where you want the debugger to started and wait for your input

```
from ..utils.remote_pdb import set_trace
set_trace() # you'll see the port number in the logs
```

1. When the above code is executed, it will print the port number to connect to in the logs.
2. use telnet to connect and use pdb as usual: for example `telnet 127.0.0.1 PORTNUM_HERE`

4.6.3 Cluster Mode

> **Warning:** Experimental

Cluster mode is handled by Ray. For more information visit ray.io.

WIP, this documentation is incomplete and more testing needed.

Requirements * PostgreSQL to communicate over network (Temporary) * Python Version should be same on all nodes

4.6.4 with Docker

Switch to python env to 3.7.7

This is needed because the docker version of Ray uses 3.7.7 and multiple versions of python create problems when pickling.

```
conda install python=3.7.7
```

Docker Related Issue with using Ray

- file permission issues when using docker. files copied using rsync get permissions
- /tmp/ray_tmp_mount

4.6.5 without Docker

- **Install MINICONDA**
    1. wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
    2. bash ~/Miniconda3-latest-Linux-x86_64.sh
    3. login logout
    4. conda install python=3.7.7

** Configuration **

Config file path: ~/pistarlab/cluster.yaml

TODO, see for details

Startup

1. Starting cluster

    ```
    ray up -vvvv ~/pistarlab/cluster.yaml
    ```

2. Launch piSTAR Lab on head node

    ```
    python pistarlab/launcher.py --ray_address="IP_ADDRESS:6379"
    ```

Install extensions on nodes

```
ray exec $HOME/pistarlab/cluster.yaml  "pip install --user -e /home/pistarlabuser/app/
↪pistarlab/extensions/pistarlab-envs-gym-main"
```

Sync Data with Nodes

Before each task run

```
ray rsync-up -vvv ~/pistarlab/cluster.yaml
```

After each task (or as needed)

```
ray rsync-down -vvv ~/pistarlab/cluster.yaml /home/pistarlabuser/pistarlab/data/ $HOME/
↪pistarlab/data/
```

4.7 Troubleshooting

4.7.1 Missing .so error when running tensorflow

ensure LD_LIBRARY_PATH is correct

4.7.2 My GPU is suddenly not avilable (Ubuntu)

If your GPU was working previously, but suddenly is not accessable the the system. The following script may help the script at `scripts/fix_gpu.sh` may help.

4.7.3 Testing GPU

Check if torch is detecting the GPU

python -c "import torch; print(torch.cuda.is_available());"

4.7.4 Running Atari Environment on Windows

If you get this error:

```
FileNotFoundError: Could not find module '..\atari_py\ale_interface\ale_c.dll' (or one␣
↪of its dependencies). Try using the full path with constructor syntax.------------
```

Reinstall Atari from here: https://github.com/Kojoley/atari-py/releases

4.8 Design

piSTAR Lab is build from on serveral excellent opensource technologies including:

- Redis (redis.io) - in-memory data store
- Ray (ray.io) - distributed processing framework to handle remote task execution
- SQLite (sqlite.org)/ PostgreSQL (postgresql.org)
- Vuejs (vuejs.org) - web development framework built on (nodejs)
- Theia IDE (theia-ide.org) - web based IDE

4.9 Developer Notes

4.9.1 Setting up your environment

Tested on Ubuntu

1. Install Anaconda or Miniconda

   Visit https://www.anaconda.com/products/individual for instructions

2. Create Conda Virtual Environment

```
conda create -n pistarlab python=3.7
```

3. Install PIP

```
conda install pip
```

4. Clone Repo and install

```
git clone https://github.com/pistarlab/pistarlab
cd pistarlab
pip install -e .[all]
```

5. **Install additional dependencies**

   - XVFB to render without display (No MS Windows Support)
   - ffmpeg for video processing

```
sudo apt-get install -y xvfb ffmpeg
```

6. Install NPM/Nodejs for UI and Theia IDE

```
bash ./install_node.sh
```

7. Build Web IDE (optional)

```
bash ./build_ide.sh
```

4.9.2 on Windows [Experimental]

Limitation: no headless mode for many environments so rendering may open a window

1. Install Miniconda
2. Install GitBash
3. Follow Ubuntu Instructions

Troubleshooting

Building Theia IDE on Windows. * https://github.com/eclipse-theia/theia/blob/master/doc/Developing.md#building-on-windows

Install Scoop

See: https://github.com/lukesampson/scoop#installation

```
Invoke-Expression (New-Object System.Net.WebClient).DownloadString('https://
↪get.scoop.sh')

# or shorter
iwr -useb get.scoop.sh | iex
# IF SCOOP doesn't get added to path
$env:Path += ";C:\Users\${USER}\scoop\shims"
```

4.9.3 Making changes to the UI

The UI is build using Vuejs cli and requires npm to run. Once setup, changes to the ui source code will be reflected immidiately in the browser.

Option 1: # Start pistarlab with enable_dev_ui argument. Eg: `pistarlab_launcher --enable_dev_ui`

Option 2: #. Run the UI using `npm run serve` #. By default, changes will be reflected at http://localhost:8080

4.9.4 Building for Readonly Viewing

```
pip install -e . -no-deps
pip install -r requirements-webreadonly.txt
```

4.9.5 Building for PiPy

1. Build Redis Server Binary and copy to pistarlab/thirdparty_lib

```
bash ./install_redis.sh_
```

2. Build and deploy UI in pistarlab/uidist/ package directory

```
bash ./build_ui.sh
```

3. Run Tests with tox

```
pip install tox
tox
```

4. Building wheel and source distribution and view files

```
rm -rf build dist *.egg-info &&
python setup.py bdist_wheel && python -m build --sdist --wheel && unzip -l dist/*.
↪whl
```

5. Uploading to PiPy

```
pip install twine
twine upload dist/*
```

4.9.6 Building the Documentation

1. Install dependencies

```
pip install -r docs/requirements.txt
```

2. Rebuild API Docs

   From the project root, run:

```
cd docs
sphinx-apidoc -o source ../pistarlab
```

3. Update the HTML

```
cd docs
make html
```

4.9.7 Building and Publishing a new Docker Image

Instructions on how to create a docker image from an Ubuntu environment

1. Make changes to docker file
2. Update requirements.txt

   ```
   ```

   conda create -n pistarlab377 python=3.7.7 conda activate pistarlab377 pip install -e . pip freeze > requirements.txt

3. Run Docker Build

   ```
   ```

   ./build_docker

4.10 Roadmap

Warning: Under development

4.10.1 Planned Features

- Composable Agents - agents built from reusable components
- Public repositories for agents and component snapshots
- Filter by observation/action space

- Persistent environments
- UI for easy hyper parameter exploration
- Learning resource: videos, documentation, and tutorials
- Missions which help users gain intuition about different aspects of reinforcement learning
- Hosted competitions

4.11 pistarlab

4.11.1 pistarlab package

Subpackages

pistarlab.agents package

Submodules

pistarlab.agents.random module

Module contents

pistarlab.envs package

Submodules

pistarlab.envs.ma_test_envs module

Module contents

pistarlab.tasks package

Submodules

pistarlab.tasks.matask module

pistarlab.tasks.tune_default module

Module contents

pistarlab.thirdparty_lib package

Module contents

pistarlab.utils package

Submodules

pistarlab.utils.agent_helpers module

pistarlab.utils.env_helpers module

pistarlab.utils.gym_importer module

pistarlab.utils.logging module

pistarlab.utils.misc module

pistarlab.utils.param_helpers module

pistarlab.utils.pkghelpers module

pistarlab.utils.pyson module

pistarlab.utils.remote_pdb module

pistarlab.utils.serialize module

pistarlab.utils.snapshot_helpers module

pistarlab.utils.timer module

Module contents

pistarlab.wrappers package

Submodules

pistarlab.wrappers.default module

Module contents

Submodules

pistarlab.agent module

pistarlab.api_schema module

pistarlab.app_backend module

pistarlab.common module

pistarlab.component module