

Welcome to the AIM Sandbox Environment to rapidly test new machine learning, testing, and evaluation approaches.

It serves two goals:

1) Give you a glimpse of the types of problems we are solving (admittedly quite simplified in this toy interview setup to focus on core concepts).

2) Give us a chance to see how you approach and solve such problems.

This sandbox is used to evaluate your skills to work on production ML code with the team and therefore is expected to be fully functional, documented, and tested. Aiming for the most realistic interview possible, we will evaluate the results as if this was your mini starter project on day one at AIM.

In addition to achieving high-quality results with elegant production code, we value creative and innovative solutions that generalize well.

The base reinforcement learning algorithm implemented here is a bare-bones [Deep Deterministic Policy Gradient](https://arxiv.org/pdf/1509.02971.pdf) (DDPG), which is an actor-critic architecture that allows model-free off-policy learning. It can efficiently handle continuous action spaces by using parameterized function approximators (via neural net models) to represent both the Q-function (critic) and the policy (actor). Iteratively, the actor acts like a policy network: given a state, it picks the action with (what it currently thinks yields) the highest expected reward. The critic model -- which acts like the Q-value network -- predicts the value of that action.

While this approach is pretty good and learns this task well in a few minutes, there are several aspects to improve from the perspectives of machine learning, reinforcement learning, production code hardening and testing, thorough evaluation.

Few starter ideas to get you started on the **ML/RL side**:

1. Besides DDPG, a [Stochastic Latent Actor-Critic](https://arxiv.org/pdf/1907.00953.pdf) (SLAC) approach can be added, which offers a number of benefits over vanilla DDPG.
2. With SLAC, we can leverage machine vision to measure joints/hand/goal positions instead of simply reading them from the 2D environment. SLAC is well poised for this by learning a latent representation of complex signals (screen pixel values in this case). Right now the code "cheats" by always knowing where things are directly from code, but in reality such measures are hard to get and need to be estimated from noisy signals, such as video. Pixel frames can be easily saved/accessed with pyglet in the current setup.
3. Optimize architecture and hyperparameters to achieve more efficient learning, inference, faster convergence, model quality, and stability
4. Weave in classical control theory, such as embedding a PID-type control smoothing in the reward function to encourage smooth behavior and improve resistance to noise
5. Curriculum learning?
6. …

And a few starter ideas for **testing and production perspectives**:

1. Is the system rigorously tested to ensure it performs well in a full range of conditions in the wild?
2. What is the optimal testing process for this system?
3. What should a release plan and process look like to make sure a larger team can concurrently develop the system?
4. How would you drive putting these in place for a streamlined test and release pipeline?
5. How do you leverage simulation, hardware in the loop testing, and other methods to have the most efficient platform to test and release autonomy features?
6. …

**Anything else you think has a high impact here? Often, disproportionate impact stems from nailing multiple aspects simultaneously.**

We are curious where you take this – it’s very open ended by design. Using any available tools, papers, code, references, existing packages/libraries or github projects with proper attribution is fair game here. **The goal is to get the sandbox running at a next-level, using production quality tested code, and arrive at a compelling result.**

To install Dependencies:

conda env create -f environment.yml

Or install manually:

pip install tensorflow==1.15.4

pip install pyglet (with pillow dependencies)

To run in inference mode: python main.py TEST

To run in learning mode: python main.py TRAIN

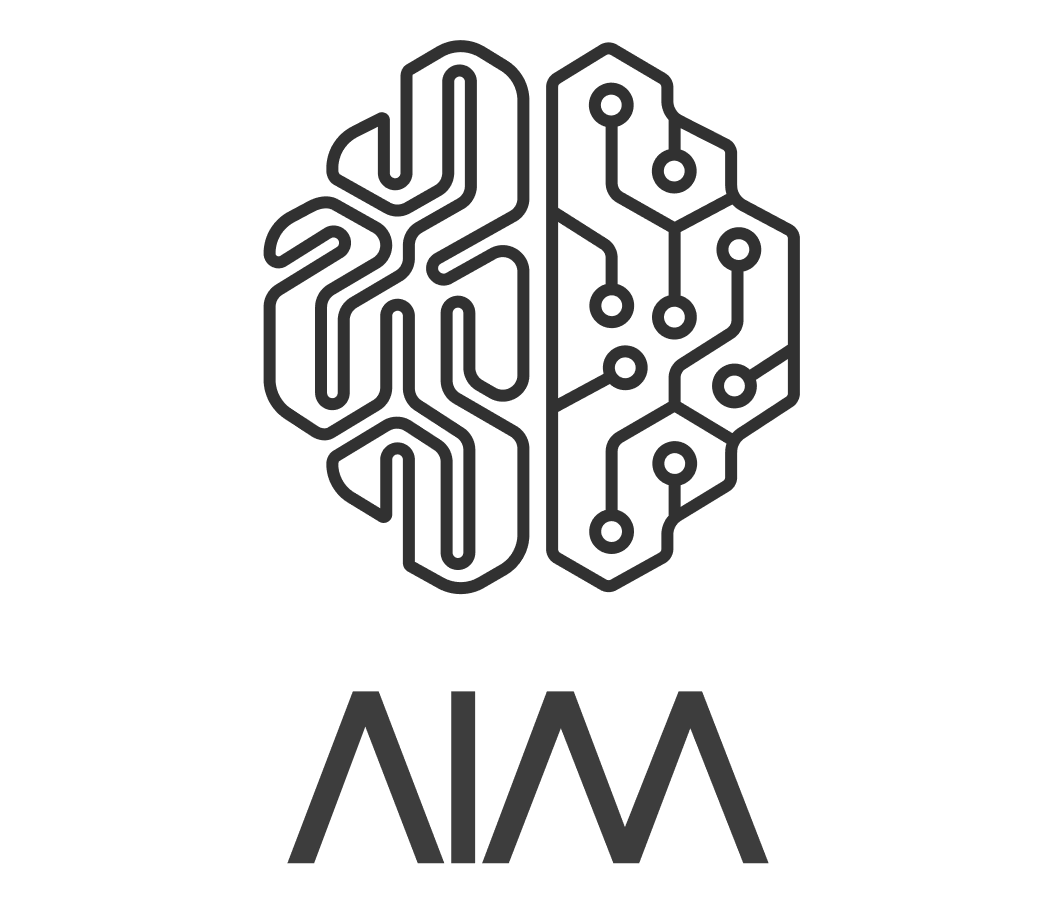
Note the API of the toy env.py is compatible with standard OpenAI gym environments. This makes plugging in existing ML libraries and visualization tools easy.

You have 7 days to complete this exercise from the time this packet was sent to you, submissions will be accepted any time prior to the end of that window.

When done, please share with us an offline git repo including a .pdf file with the documentation of your solution and results.

Feel free to reach out with questions at interviews@machines.run

Happy coding!



The AIM Team