Reinforcement learning lab report

Adrian-Gabriel Bălănescu

June 18, 2021

Contents

Chapter 1

Game description

Gravitar is an arcade style game released by Atari in 1982. The game was known for its high level of difficulty [1].

1.1 Gameplay

The game takes place in a fictional solar system. The player controls a spacecraft with the help of five buttons: two to rotate the ship left and right, one to shoot, one to activate the thruster and one for both a tractor beam and a force field. In the side-view levels, the player has to destroy red bunkers that shoot constantly, and can also use the tractor beam to pick up blue fuel tanks. Once all of the bunkers are destroyed, the planet will blow up, and the player will earn a bonus. Once all planets are destroyed, the player will move onto another solar system. The player will lose a life if he crashes into the terrain or gets hit by an enemy's shot, and the game will end immediately if fuel runs out [1].

After completing all 11 planets (or alternatively completing the reactor three times) the player enters the second universe and the gravity will reverse. Instead of dragging the ship towards the planet surface, the gravity pushes it away. In the third universe the landscape becomes invisible and the gravity is positive again. The final, fourth universe, has invisible landscape and reverse gravity. After completing the fourth universe the game starts over [1].

Figure 1.1: Gravitar logo

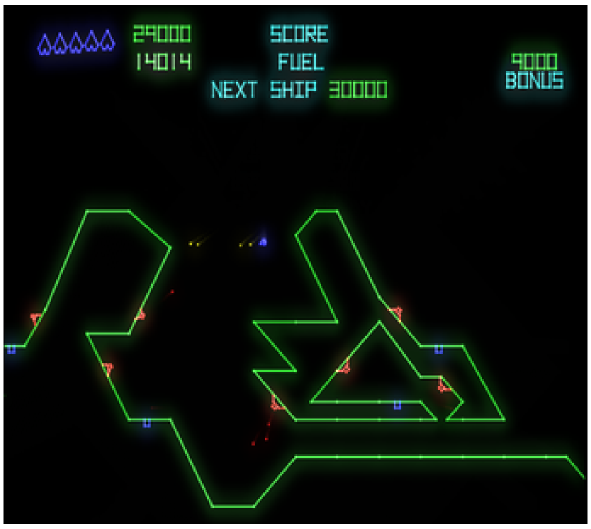

Figure 1.2: A level from Gravitar

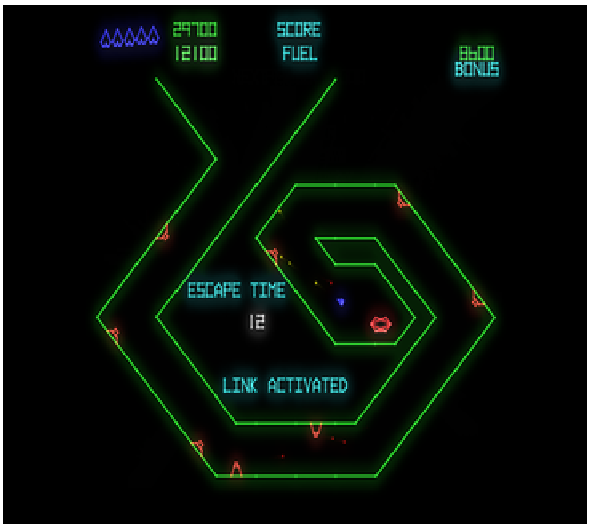

Figure 1.3: The reactor

1.2 Gravitar in SLM-lab

SLM-lab [2] is a software framework for reproducible reinforcement learning (RL) research. It implements most of the canonical RL algorithms in a modular platform that allows for easy experimentation and analysis. It includes the OpenAI Gym [3] environments, which contains a lot of Atari games emulations, one being Gravitar.

In this environment, the observation is an RGB image of the screen, which is an array of shape (210, 160, 3). Each action is repeatedly performed for a duration of k frames, where k is uniformly sampled from $\{2,3,4\}$. The action space is discrete and has 18 available commands like fire, up, down, right, left and so on.

The Gravitar environment can be loaded in SLM-lab by specifying in the .json configuration file, at the "env" section the "name" of the environment. In this case, "GravitarNoFrameskip–v4".

Chapter 2

Algorithms

In this chapter, the algorithms used will be briefly explained.

2.1 Reinforce

Reinforce is a Monte Carlo variant of the policy-gradient algorithm in reinforcement learning. The agent collects samples of an episode using its current policy and uses it to update the policy parameter θ. It is updated as an off-policy algorithm:

$$\nabla_\theta J(\theta) = E_\pi[G_t \nabla_\theta ln \pi_\theta(a_t, s_t)] \tag{2.1}$$

2.2 A2C - Advantage Actor-Critic

The Advantage Actor-Critic algorithm relies on two important concepts:

The Actor-Critic architecture. This type of algorithms learn two parametrized functions simultaneously. The actor learns the actual policy that the agent uses to act in the world, while the critic learns the value function that evaluates the state-action pairs.

Using the critic to provide a reinforcement signal has the advantage of being able to feed a more continuous signal for training that is useful for example in sparse rewards environments. However, training becomes more difficult because the policy is dependent on the quality of the value function. In the beginning of the training process, the value function is not learned, so the evaluation signal provided doesn't help much to select better actions. Therefore, usually an Advantage function is actually learned.

Reinforcing signal based on the Advantage function. The advantage function measures how much an action is better or worse than a policy's average action in a certain state. It is defined as follows:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \tag{2.2}$$

It has the following properties:

- The advantage is a relative measure. It quantifies whether a particular action a in a state s will improve the policy relative to the value $V^{\pi}(s)$.
- If an action is worse than the average action, but the expected return is still positive - the action taken should become less probable. This is the behavior of the advantage function, in contrast with other methods that might actually encourage that action.
- The advantage function is able to capture long-term effects of an action, because it considers all future time steps, while ignoring the effects of all the actions to date.

2.2.1 Estimating the advantage with n-Step returns

The n-Step estimate is formulated as follows:

$$A^{\pi}_{Nstep}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) \approx r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_{t+n} + \gamma^{n+1} \hat{V}^{\pi}(s_{t+n+1}) - \hat{V}^{\pi}(s_t) \tag{2.3}$$

The hyperparameter n - the number of actual rewards, controls the bias-variance trade-off.

2.2.2 Estimating the advantage with Generalized Advantage Estimation

GAE is defined as an exponentially weighted average of all of the n-step forward return advantages:

$$A^{\pi}_{GAE}(s_t, a_t) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{2.4}$$

where, $\delta_t = r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$

The variance is controlled through the decay coefficient λ

2.3 PPO - Proximal Policy Optimization

PPO [4] aims at avoiding the problem of performance collapse through a surrogate objective that is guaranteed to have monotonic improvement. It works by extending either REINFORCE or an actor-critic algorithm with the modified objective. It is data efficient and reliable as TRPO [5] but uses first-order optimization. Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ so r(θ_{old}) $= 1$. TRPO maximizes a "surrogate" objective:

$$L^{CPI}(\theta) = \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] = \hat{E}_t \left[r_t(\theta) \hat{A}_t \right] \tag{2.5}$$

where CPI refers to a conservative policy iteration.[4] Without a constraint, maximization of L^{CPI} would lead to an excessively large policy update, therefore, PPO modifies the objective to penalize changes to the policy that moves $r_t(\theta)$ away from 1:

$$J^{CLIP}(\theta) = \hat{E} \left[min \left(r_t(\theta)\hat{A}, clip\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A} \right) \right] \tag{2.6}$$

where ϵ is a hyperparameter. The intuition is as follows: The $clip$ term modifies the surrogate objective by clipping the probability ratio, which removes the incentive to move r_t outside the interval $[1-\epsilon, 1+\epsilon]$ [4].

Chapter 3

Experimental Results

3.1 Reinforce

For training Reinforce, the following configuration has been used:

```
{
 "reinforce_gravitar": {
  "agent": [{
   "name": "Reinforce",
   "algorithm": {
    "name": "Reinforce",
    "action_pdtype": "default",
    "action_policy": "default",
    "center_return": true,
    "explore_var_spec": null,
    "gamma": 0.99,
    "entropy_coef_spec": {
     "name": "linear_decay",
     "start_val": 0.01,
     "end_val": 0.001,
     "start_step": 1000,
     "end_step": 300000
    },
    "training_frequency": 1
   },
   "memory": {
    "name": "OnPolicyReplay",
    "batch_size": 64
   },
   "net": {
    "type": "ConvNet",
    "shared": true,
    "conv_hid_layers": [
    [32, 8, 4, 0, 1],
    [64, 4, 2, 0, 1],
    [32, 3, 1, 0, 1]
    ],
    "fc_hid_layers": [512],
    "hid_layers_activation": "relu",
    "init_fn": "orthogonal_",
    "normalize": true,
    "batch_norm": false,
    "clip_grad_val": 0.5,
    "loss_spec": {
     "name": "MSELoss"
    },
    "optim_spec": {
     "name": "Adam",
     "lr": 0.002
    },
    "lr_scheduler_spec": null,
    "gpu": true
   }
  }],
  "env": [{
   "name": "GravitarNoFrameskip-v4",
   "frame_op": "concat",
   "frame_op_len": 4,
   "reward_scale": "sign",
   "num_envs": 16,
   "max_t": null,
   "max_frame": 300000
  }],
  "body": {
   "product": "outer",
   "num": 1
  },
  "meta": {
   "distributed": false,
   "eval_frequency": 10000,
   "max_session": 1,
   "max_trial": 1
  }
 }
}
```

The following parameters are set:

- The algorithm is set to Reinforce
- The action policy is the default one
- The agent uses a convolutional network with three layers connected to a single linear layer with ReLU activations.
- The optimizer is Adam.
- Algorithm is trained for 300k steps.
- Evaluation occurs every 10k steps.

  The following returns are obtained:

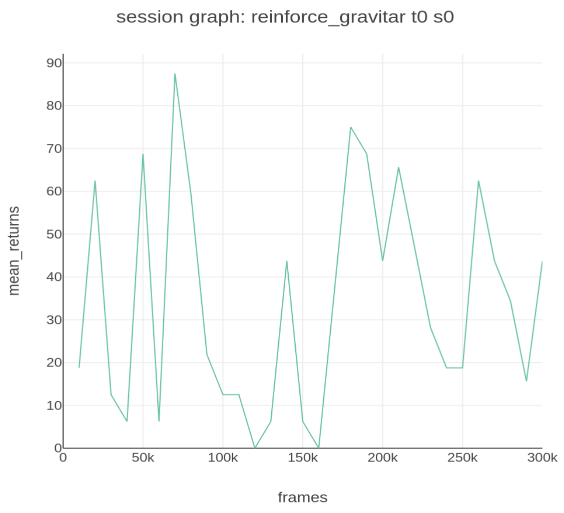

Figure 3.1: Trial graph

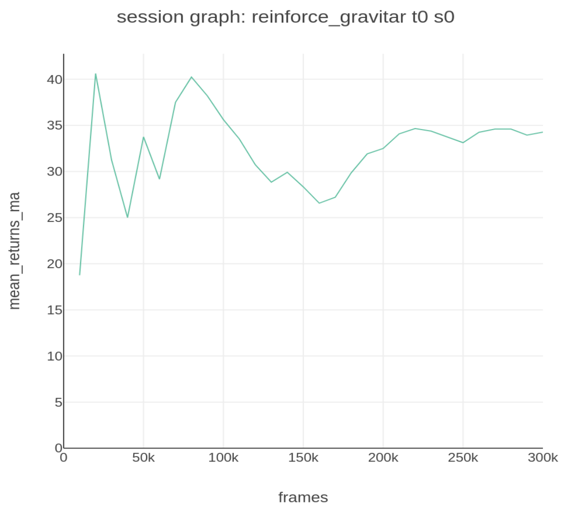

Figure 3.2: Trial graph with moving average

Figure 3.3: Reinforce trial graphs from SLM lab. On the vertical axis the total rewards are shown and on the horizontal axis the number of frames.

3.2 A2C

For training A2C, two experiments have been considered:

1. Training with generalized advantage function (GAE) The config file is as follows:

```
{
 "a2c_gae_gravitar": {
  "agent": [{
   "name": "A2C",
   "algorithm": {
    "name": "ActorCritic",
    "action_pdtype": "default",
    "action_policy": "default",
    "explore_var_spec": null,
    "gamma": 0.99,
    "lam": 0.95,
    "num_step_returns": null,
    "entropy_coef_spec": {
     "name": "no_decay",
     "start_val": 0.01,
     "end_val": 0.01,
     "start_step": 0,
     "end_step": 0
    },
    "val_loss_coef": 0.5,
    "training_frequency": 32
   },
   "memory": {
    "name": "OnPolicyBatchReplay"
   },
      "net": {
        "type": "ConvNet",
        "shared": true,
        "conv_hid_layers": [
        [32, 8, 4, 0, 1],
        [64, 4, 2, 0, 1],
        [32, 3, 1, 0, 1]
        ],
        "fc_hid_layers": [512],
        "hid_layers_activation": "relu",
        "init_fn": "orthogonal_",
        "normalize": true,
        "batch_norm": false,
        "clip_grad_val": 0.5,
        "use_same_optim": false,
        "loss_spec": {
          "name": "MSELoss"
        },
        "actor_optim_spec": {
          "name": "RMSprop",
          "lr": 7e-4,
          "alpha": 0.99,
          "eps": 1e-5
        },x
        "critic_optim_spec": {
          "name": "RMSprop",
          "lr": 7e-4,
          "alpha": 0.99,
          "eps": 1e-5
        },
        "lr_scheduler_spec": null,
        "gpu": true
      }
    }],
    "env": [{
      "name": "GravitarNoFrameskip-v4",
      "frame_op": "concat",
      "frame_op_len": 4,
      "reward_scale": "sign",
      "num_envs": 16,
      "max_t": null,
      "max_frame": 1e6
    }],
    "body": {
      "product": "outer",
      "num": 1
    },
    "meta": {
      "distributed": false,
      "eval_frequency": 10000,
      "log_frequency": 10000,
      "rigorous_eval": 0,
      "max_session": 1,
      "max_trial": 1
```

```
            }
          }
        }
```

The following parameters are set:

- The algorithm is A2C
- The action policy is the default one
- GAE is used as an Advantage function estimator
- The agent uses a convolutional network with three layers connected to a single linear layer with ReLU activations.
- The optimizer is RMSProp.
- Algorithm is trained for 1M steps.
- Evaluation occurs every 10k steps.

The following returns are obtained:

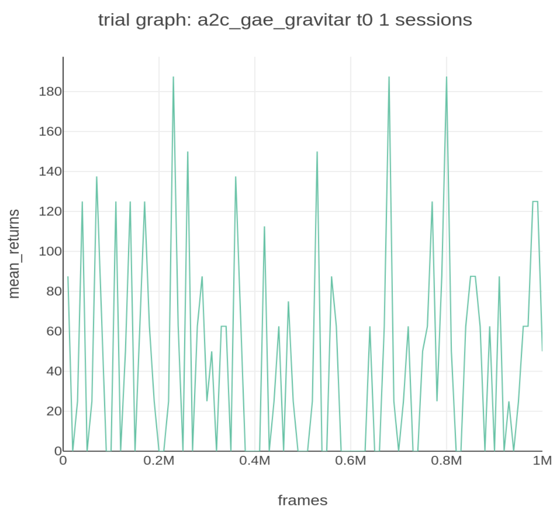

Figure 3.4: Trial graph

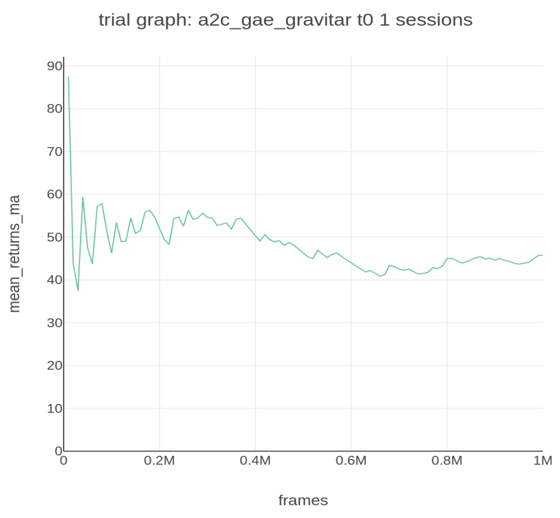

Figure 3.5: Trial graph with moving average

Figure 3.6: A2C trial graphs from SLM lab. On the vertical axis the total rewards are shown and on the horizontal axis the number of frames.

2. Training with n-Step Returns

   The config file is almost the same as the GAE version, it differs at the *num_step_returns* parameter.

   The following returns are obtained:

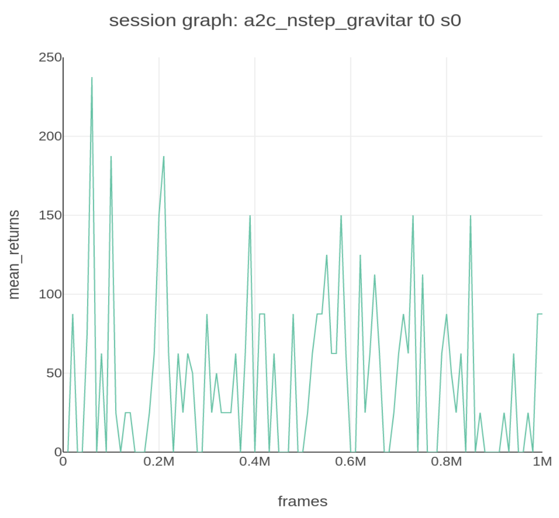

Figure 3.7: Trial graph

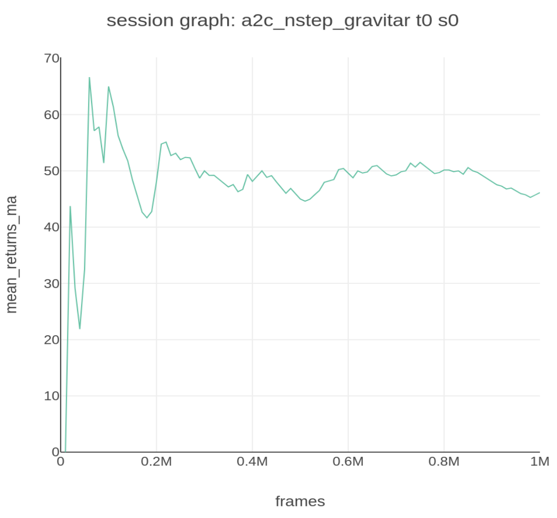

Figure 3.8: Trial graph with moving average

Figure 3.9: A2C trial graphs from SLM lab. On the vertical axis the total rewards are shown and on the horizontal axis the number of frames.

3.3 PPO

For training the Proximal Policy Optimization algorithm, the following configuration has been used:

```
{
 "ppo_gravitar": {
  "agent": [{
   "name": "PPO",
   "algorithm": {
    "name": "PPO",
    "action_pdtype": "default",
    "action_policy": "default",
    "explore_var_spec": null,
    "gamma": 0.99,
    "lam": 0.70,
    "clip_eps_spec": {
     "name": "no_decay",
     "start_val": 0.10,
     "end_val": 0.10,
     "start_step": 0,
     "end_step": 0
    },
        "entropy_coef_spec": {
          "name": "no_decay",
          "start_val": 0.01,
          "end_val": 0.01,
          "start_step": 0,
          "end_step": 0
        },
        "val_loss_coef": 0.5,
        "time_horizon": 128,
        "minibatch_size": 256,
        "training_epoch": 4
      },
      "memory": {
        "name": "OnPolicyBatchReplay",
      },
      "net": {
        "type": "ConvNet",
        "shared": true,
        "conv_hid_layers": [
        [32, 8, 4, 0, 1],
        [64, 4, 2, 0, 1],
        [32, 3, 1, 0, 1]
        ],
        "fc_hid_layers": [512],
        "hid_layers_activation": "relu",
        "init_fn": "orthogonal_",
        "normalize": true,
        "batch_norm": false,
        "clip_grad_val": 0.5,
        "use_same_optim": false,
        "loss_spec": {
          "name": "MSELoss"
        },
        "actor_optim_spec": {
          "name": "Adam",
          "lr": 2.5e-4,
        },
        "critic_optim_spec": {
          "name": "Adam",
          "lr": 2.5e-4,
        },
        "lr_scheduler_spec": {
          "name": "LinearToZero",
          "frame": 1e7
        },
        "gpu": true
      }
    }],
    "env": [{
      "name": "GravitarNoFrameskip-v4",
      "frame_op": "concat",
      "frame_op_len": 4,
      "reward_scale": "sign",
      "num_envs": 4,
73        "max_t": null,
74        "max_frame": 1e6
75      }],
76      "body": {
77        "product": "outer",
78        "num": 1
79      },
80      "meta": {
81        "distributed": false,
82        "eval_frequency": 10000,
83        "log_frequency": 10000,
84        "rigorous_eval": 0,
85        "max_session": 1,
86        "max_trial": 1
87      }
88    }
89  }
```

The following parameters are set:

- The action policy is the default policy for a discrete action space.
- Generalized advantage estimation is used.
- The clipping parameters are set in lines 12-17
- In the lines 34-43 the network architecture is set as a 3 layer convolutional network with a hidden fully connected layer and ReLU activations.
- The opimizer is set to Adam both for the actor and the critic networks.
- The environment is Gravitar.
- Evaluation is done every 10k steps.
- The algorithm is trained for 1M steps.

The following returns are obtained:

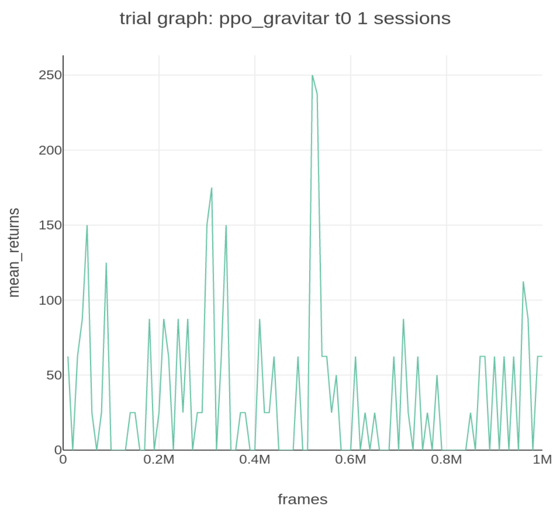

Figure 3.10: Trial graph

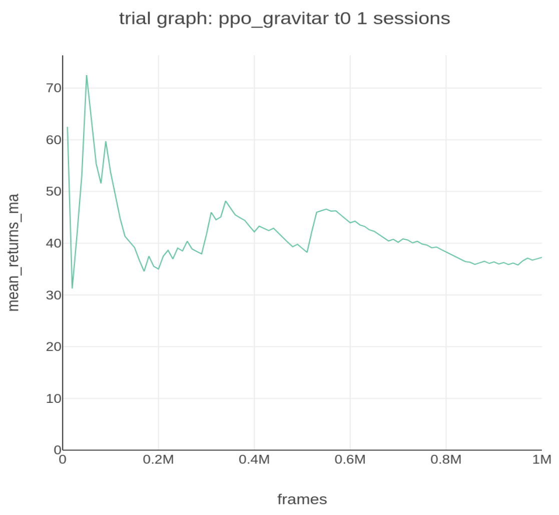

Figure 3.11: Trial graph with moving average

Figure 3.12: PPO trial graphs from SLM lab. On the vertical axis the total rewards are shown and on the horizontal axis the number of frames.

Chapter 4

Conclusions

To conclude, Gravitar is a very difficult game to solve. Not only due to it's more complex physics and controls, but also due to the fact that the levels are increasingly difficult and quite different from one to another. No algorithms from the ones tested was able to decently solve this environment. As a comparison, apart from Reinforce, which (with no surprise) performed the poorest, the other algorithms performed similarly, with the n-step A2C having the best mean returns average of approx. 50. Also, all the algorithms suffered performance degradation as more steps were taken.

Bibliography

[1] *Gravitar*. `https://en.wikipedia.org/w/index.php?title=Gravitar&oldid=1018480060`, April 2021. Page Version ID: 1018480060. 1, 1.1

[2] Wah Loon Keng and Laura Graesser. Slm lab. `https://github.com/kengz/SLM-Lab`, 2017. 1.2

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 1.2

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, August 2017. arXiv: 1707.06347. 2.3, 2.3, 2.3

[5] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust Region Policy Optimization. *arXiv:1502.05477 [cs]*, April 2017. arXiv: 1502.05477. 2.3