RL – Tasks explanation

TA. Bogyeong Suh

Machine Learning &
Control Systems Lab.

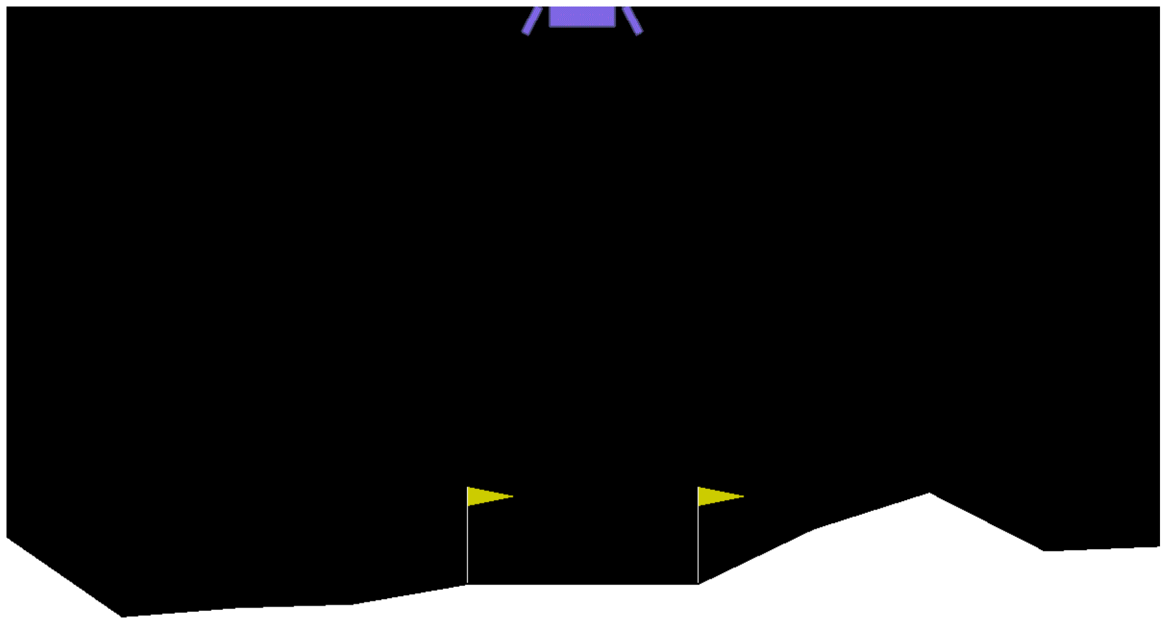
Task 1 & 2

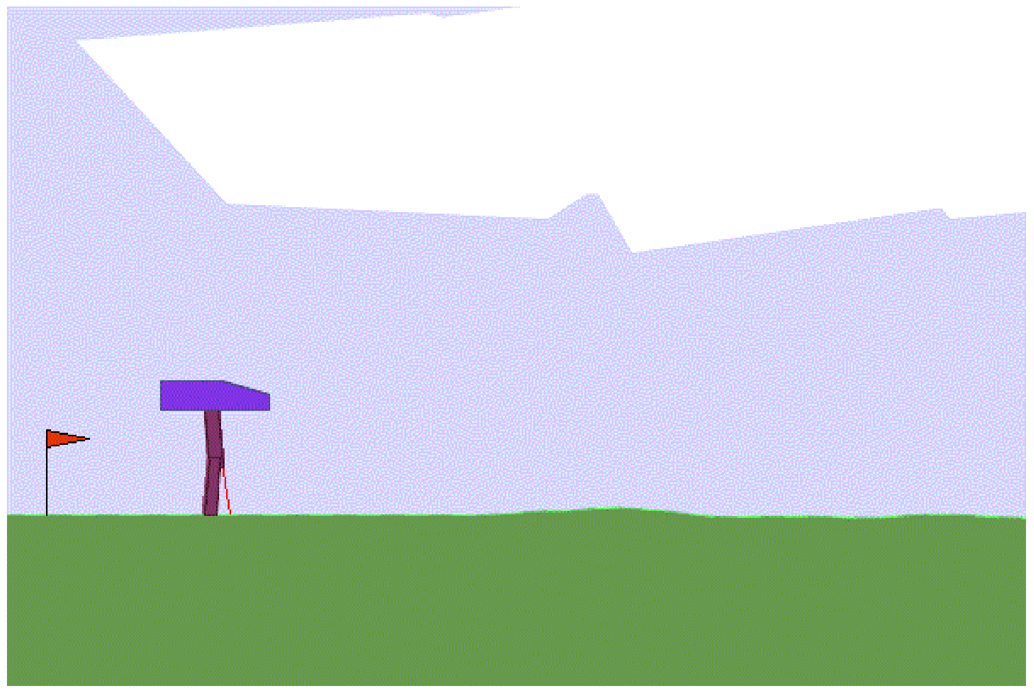
Policy Iteration and Value Iteration with toy text environments

- **Gym** interface is simple, pythonic, and capable of representing general RL problems.
- It provides a variety of standardized environments to test and develop RL agents on, ranging from classic control problems to more sophisticated tasks such as playing Atari games.

Gymnasium

An API standard for reinforcement learning with a diverse collection of reference environments


Lunar Lander

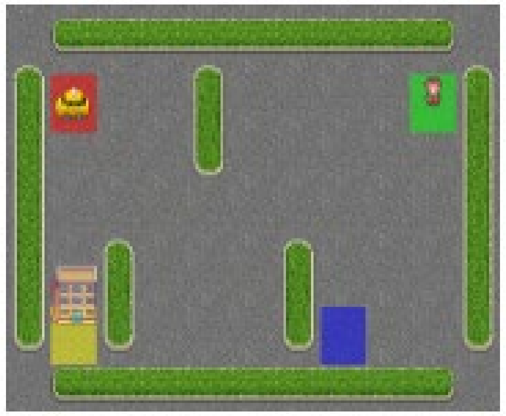

Bipedal Walker

https://www.gymlibrary.dev/

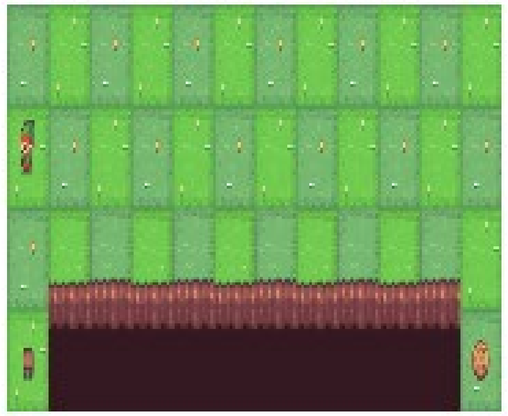
Policy Iteration and Value Iteration with toy text environments

Toy Text

Blackjack



Taxi



Cliff Walking

Frozen Lake

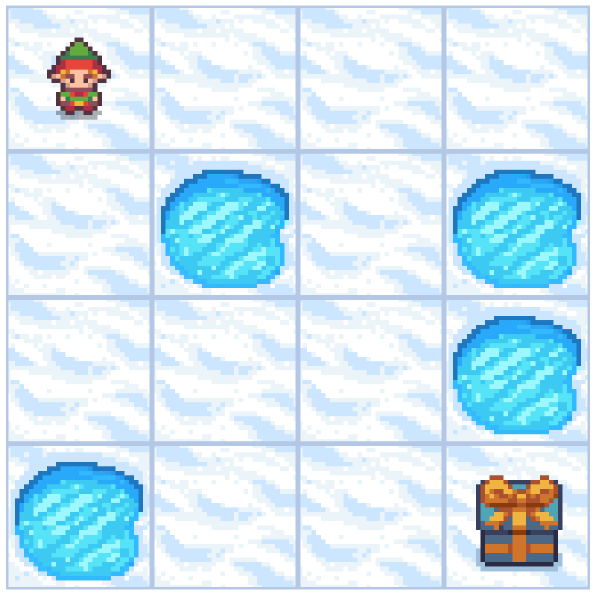
https://gymnasium.farama.org/environments/toy_text/

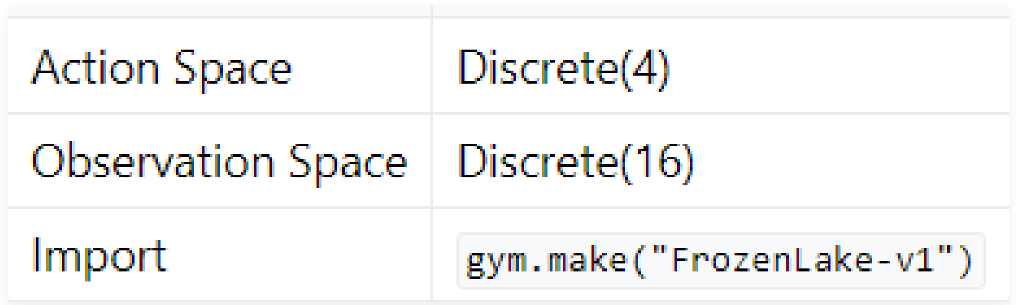
- In these tasks, we will use one of toy text environments from gym library for practicing policy and value iteration methods.
- The toy text environments have discrete state and action spaces, which are suitable for policy iteration and value iteration methods
- **Choose one of the toy text environments, and apply policy iteration method (Task 1) and value iteration method (Task 2), referring to the code files '*policy_iteration.py*' and '*value_iteration.py*'.**

Task 1 & 2

e.g.) Frozen Lake

- Frozen lake involves crossing a frozen lake from start to goal without falling into any holes by walking over the frozen lake. The player may not always move in the intended direction due to the slippery nature of the frozen lake
- Start at [0,0], and goal position located at [3,3]





Action Space	Discrete(4)
Observation Space	Discrete(16)
Import	`gym.make("FrozenLake-v1")`

***Slippery option** `is_slippery=True`
-If True, the player will move in intended direction with probability of 1/3 else will move in other perpendicular direction with equal probability of 1/3 in both directions

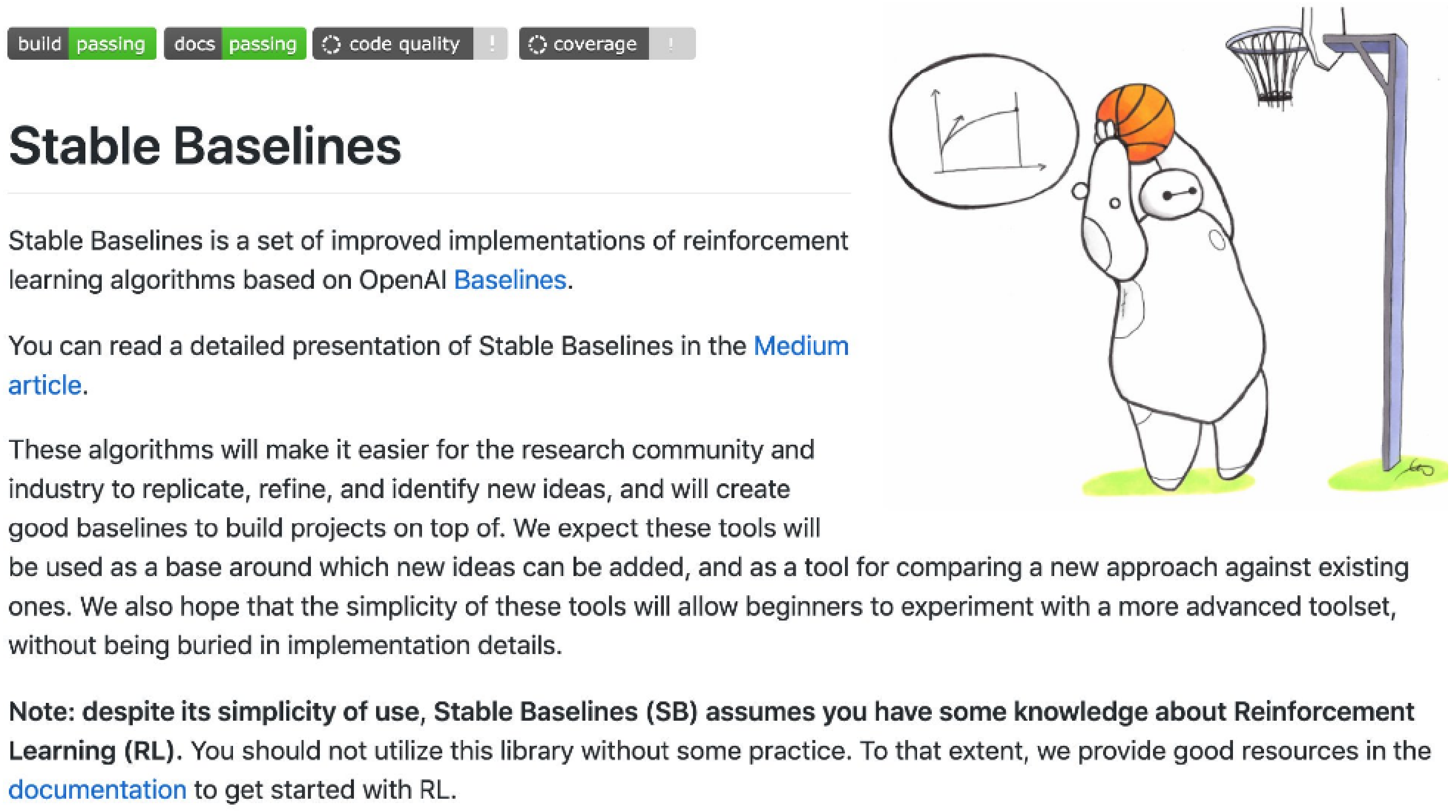
Action Space
-0 (left), 1 (down), 2 (right), 3 (up)

Observation Space
-Value representing the agent's current position as $(current\ row) \times (number\ of\ rows) + (current\ column)$
-For example, the goal position in the 4x4 map can be calculated as 3*4+3=15, and 4x4 map has 16 possible observations.

Reward
- +1 (reach goal), 0 (reach hole), 0 (reach frozen)

Practice using models from Stable baselines

- Use one of the RL algorithms available in stable baselines https://stable-baselines.readthedocs.io/en/master/



Stable Baselines

Stable Baselines is a set of improved implementations of reinforcement learning algorithms based on OpenAI Baselines.

You can read a detailed presentation of Stable Baselines in the Medium article.

These algorithms will make it easier for the research community and industry to replicate, refine, and identify new ideas, and will create good baselines to build projects on top of. We expect these tools will be used as a base around which new ideas can be added, and as a tool for comparing a new approach against existing ones. We also hope that the simplicity of these tools will allow beginners to experiment with a more advanced toolset, without being buried in implementation details.

Note: despite its simplicity of use, Stable Baselines (SB) assumes you have some knowledge about Reinforcement Learning (RL). You should not utilize this library without some practice. To that extent, we provide good resources in the documentation to get started with RL.

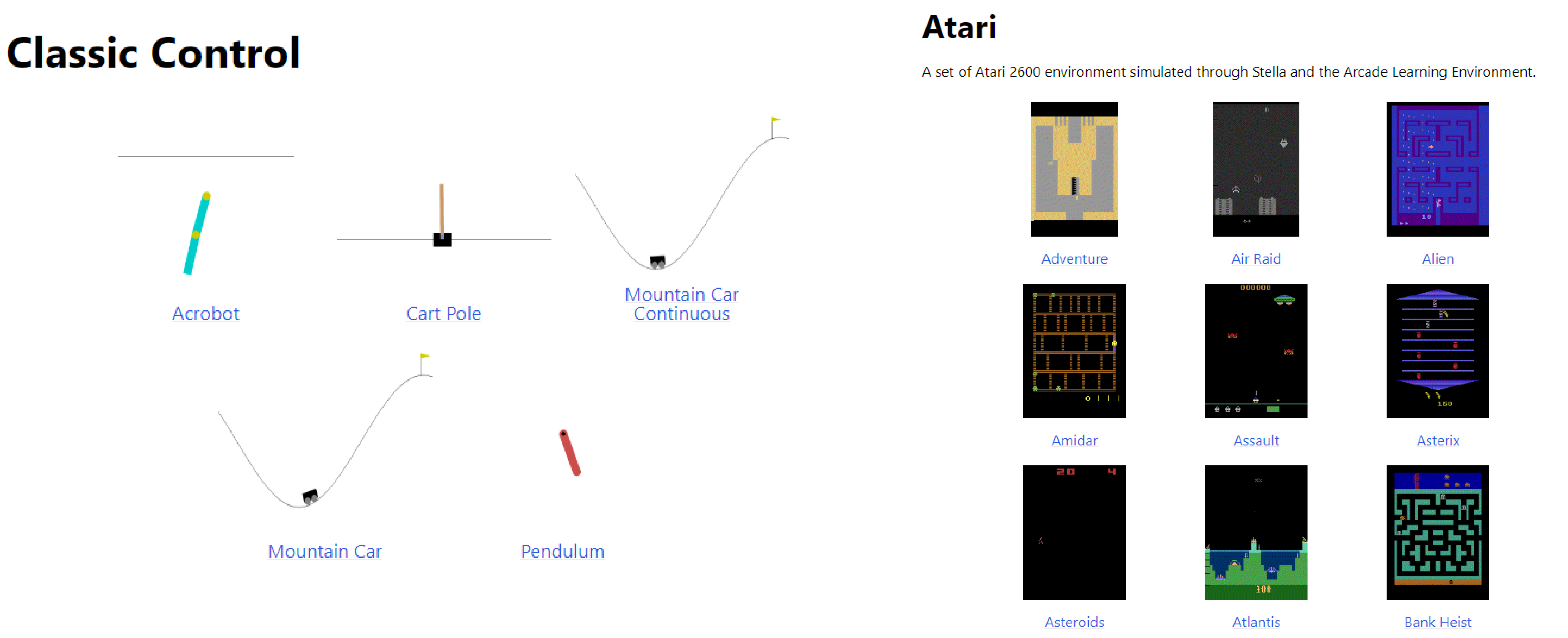
- **Deep Deterministic Policy Gradient (DDPG)**
- Policy Gradient + Q-learning

- **Deep Q Network (DQN)**
- Q-learning + Deep neural networks

- **Soft Actor Critic (SAC)**
- Off-policy maximum entropy deep RL with a stochastic actor
- Actor (policy) learns to optimize a trade-off between expected return and entropy (exploration).
- Critic (Q-function) learns to evaluate the policy

Task 3

Practice using models from Stable baselines



Classic Control

Atari

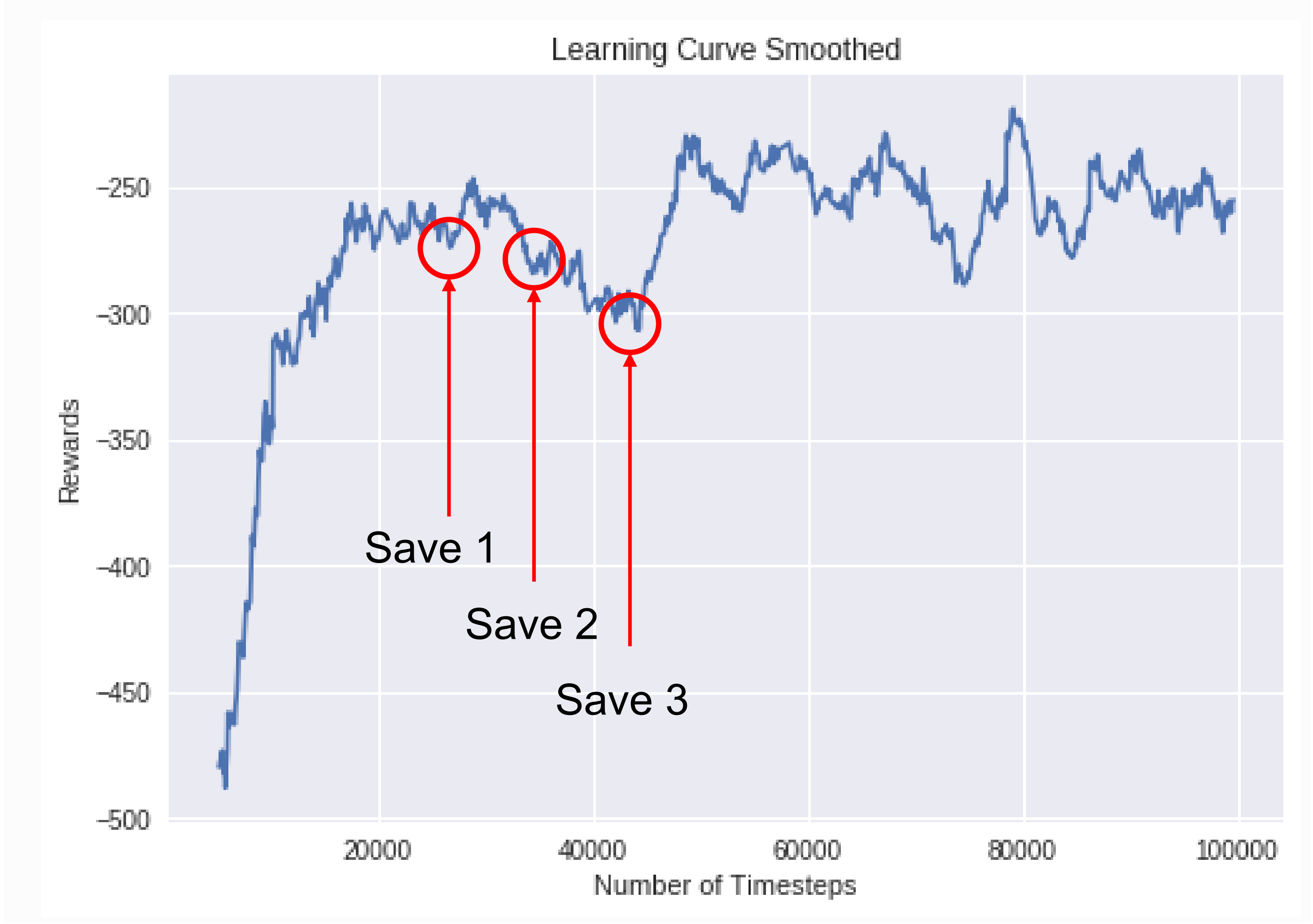
A set of Atari 2600 environment simulated through Stella and the Arcade Learning Environment.

Example environments of gym

- **Choose one of the environments in gym (not chosen on task 1&2), and one RL algorithm in stable baselines(e.g. DQN, DDPG, SAC,…)**

Task 3

Practice using models from Stable baselines



Learning curve of DDPG on LunarLanderContinuous environment

- **Plot the above reward graph of your results on training the model**
- **Use callback function to monitor training, and display points on the graph where the model is saved and reward is updated.**
- (The original callback function will save the first best reward at episode 100 as mean reward over the last 100 episodes, and then save the next best reward when the mean over the last 100 episodes is larger than the previously saved best reward.) When the reward is updated, the model is also saved.

https://stable-baselines.readthedocs.io/en/master/guide/examples.html