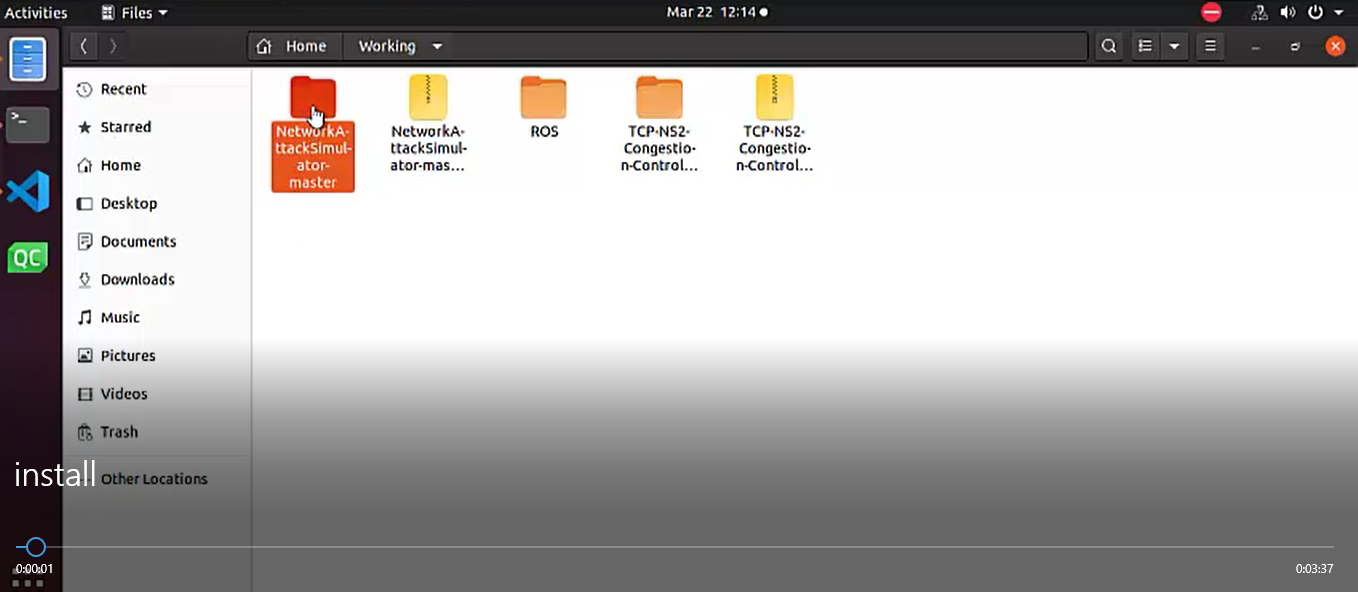
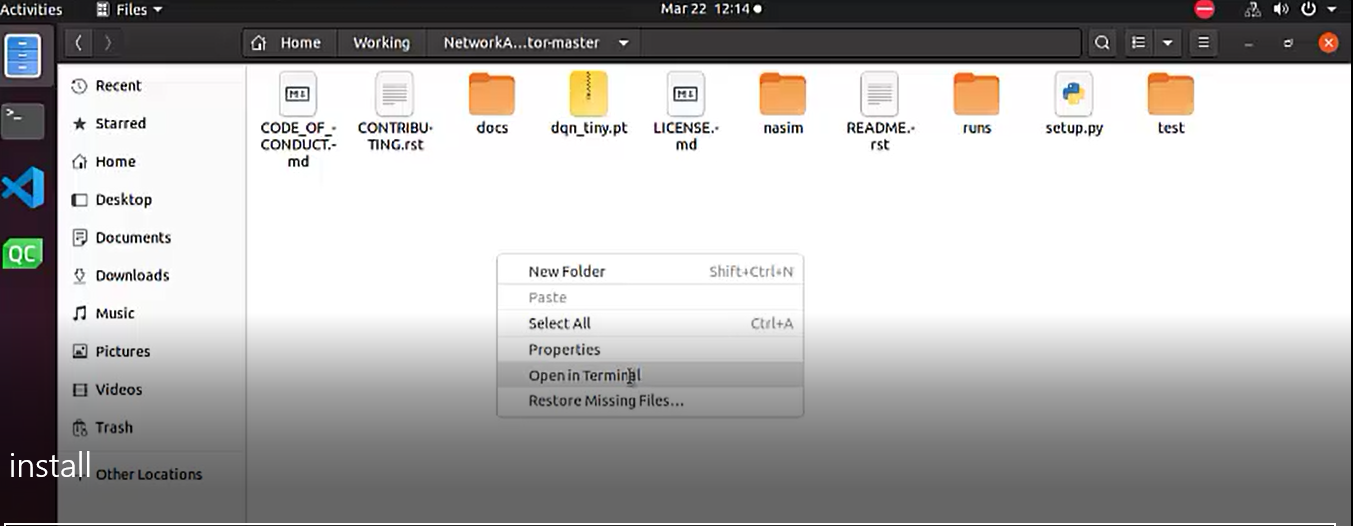
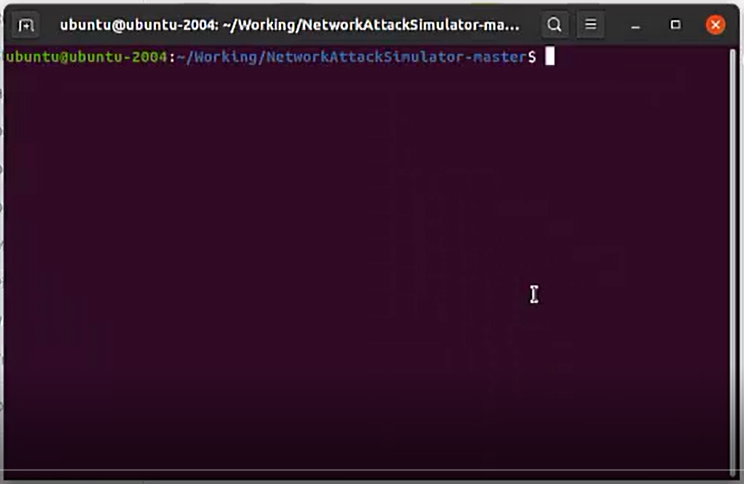
1. Install Steps
2. Input the project

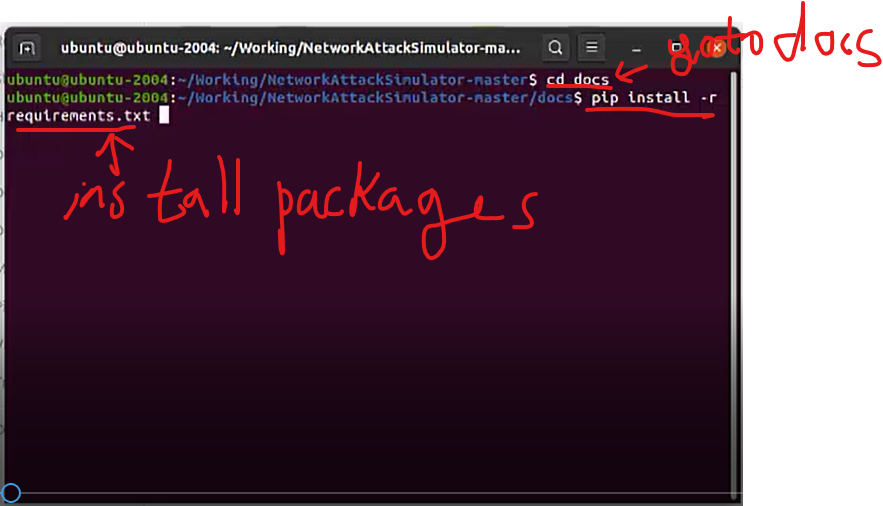


1. Open terminal

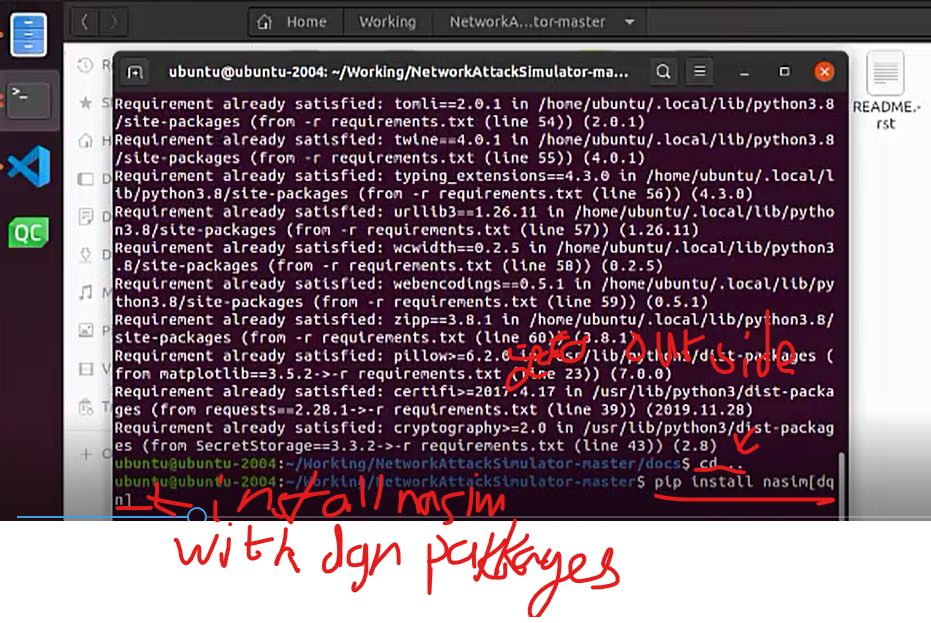




1. Install required packages

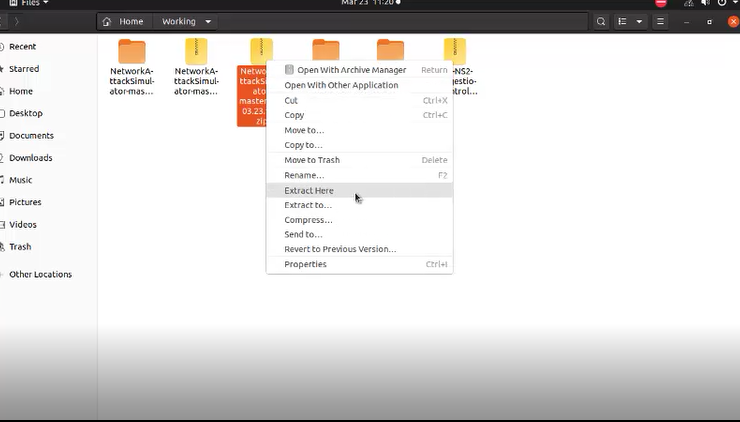


1. Install nasim

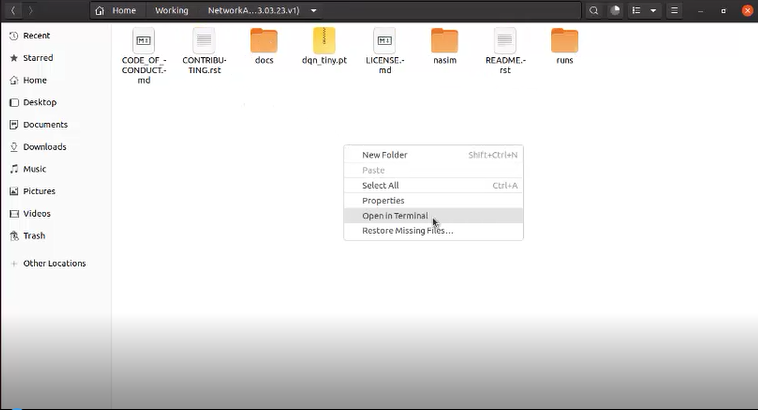


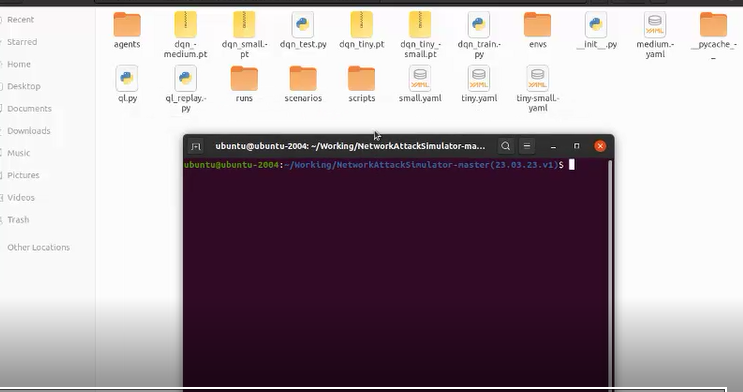
1. Run project
2. Enter the directory of project

* Extract the project



* Enter the project directory and open terminal





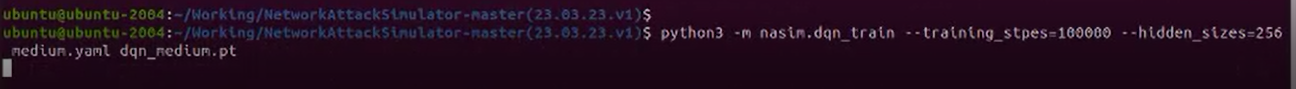
1. Train the dqn agent



-command:

$python3 –m nasim.dqn\_train [--hidden\_sizes] [--training\_steps] <scenario.yaml> <dqn\_policy.pt>

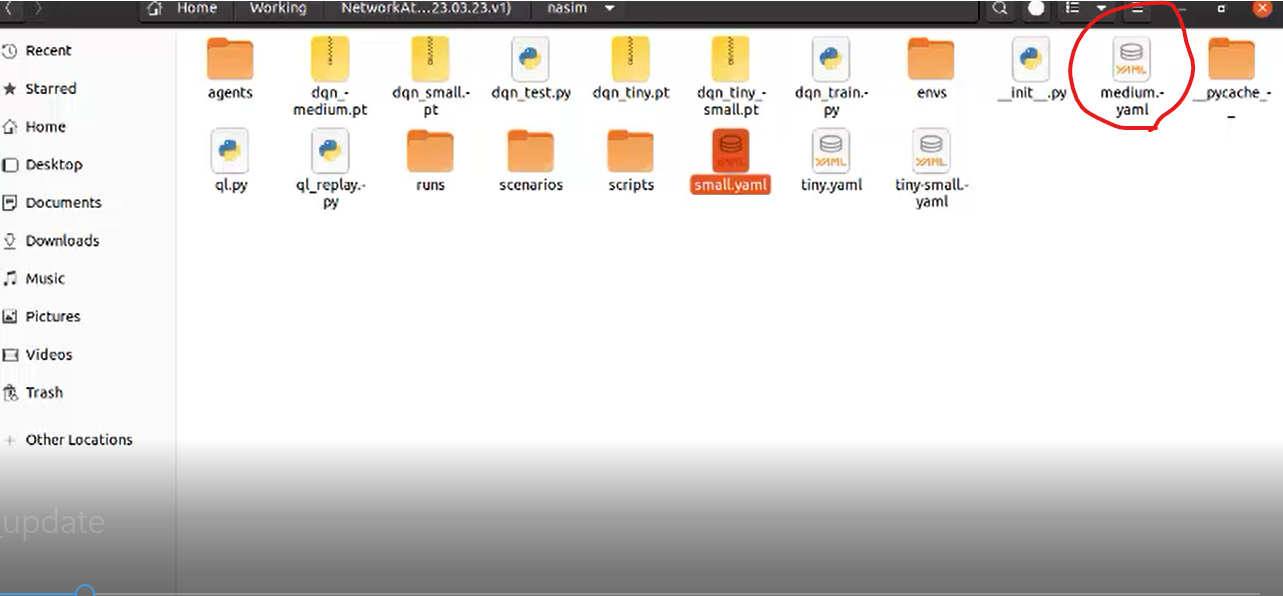
For example,



, where training steps are 100000 and sizes of hidden layer is 256.

This command uses medium.yaml as scenario file and output weight file which has name of dqn\_medium.pt.

medium.yaml file should be in the nasim directory.



Once train is finished, dqn\_train.pt will be produced in the directory.

Note: To increase the episode numbers, you should increase the training steps.

You should select the hidden sizes carefully.

To get good result, it should be set considering the scenario.

The hidden sizes should be set as the value which is greater than the number of actions in the scenario.

In the above example, since the number of actions is 192, I select the hidden sizes as 256.

I showed the commands for training and test in Appendix.

1. Test the dqn agent

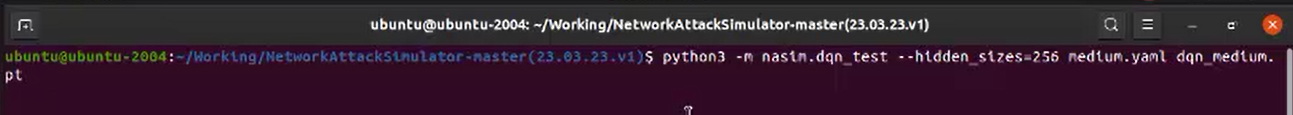


Now you can test the trained agent.

-command:

$python3 –m nasim.dqn\_test [--hidden\_sizes] <scenario.yaml> <policy.pt>

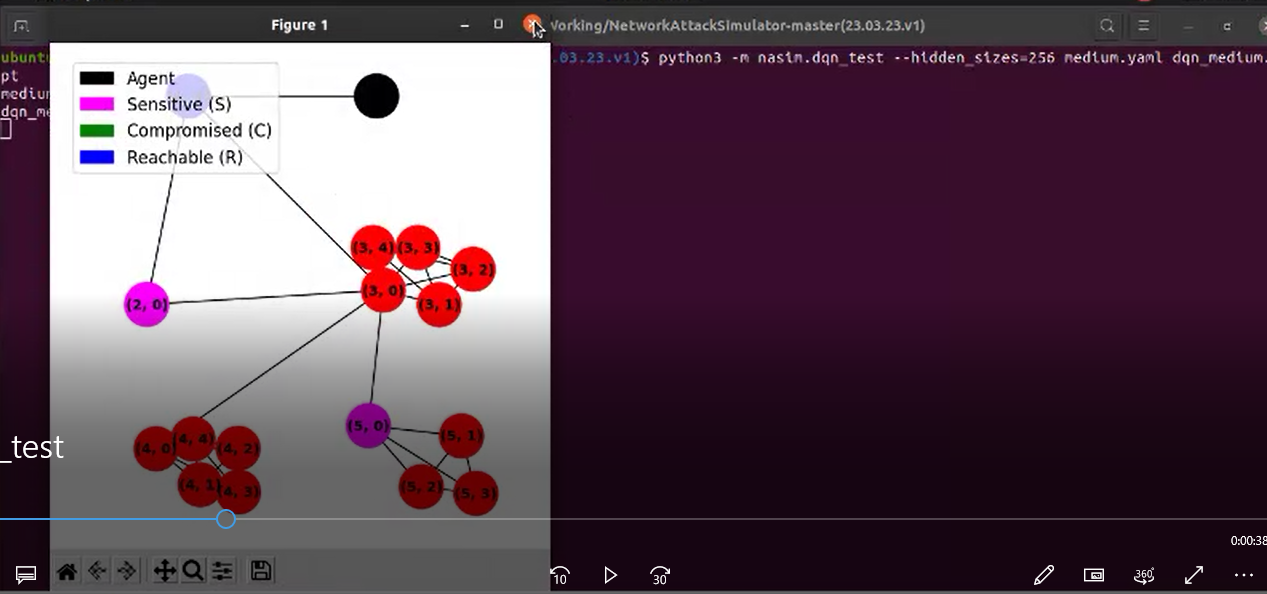
For example,



Here, you should use the same hidden sizes as in the training.

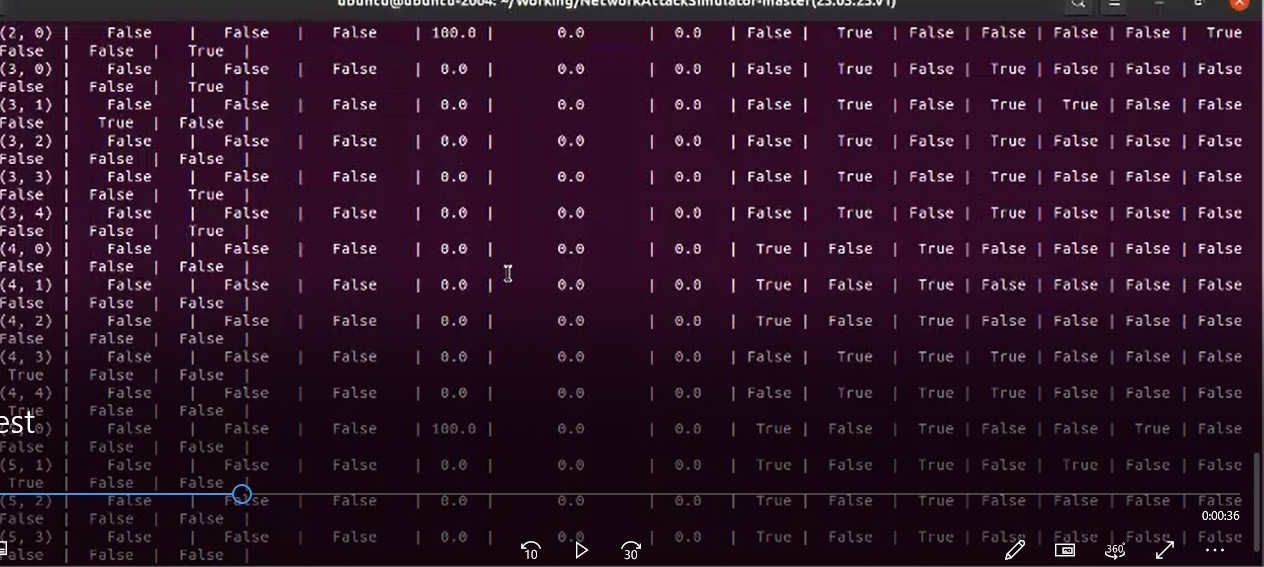
Scenario file name and policy name should be same in the training.

Once you run this command you can see the graph which represents structure and initial state of network.



You should close the figure to continue the test.

Then you can see the initial state of network in the terminal.



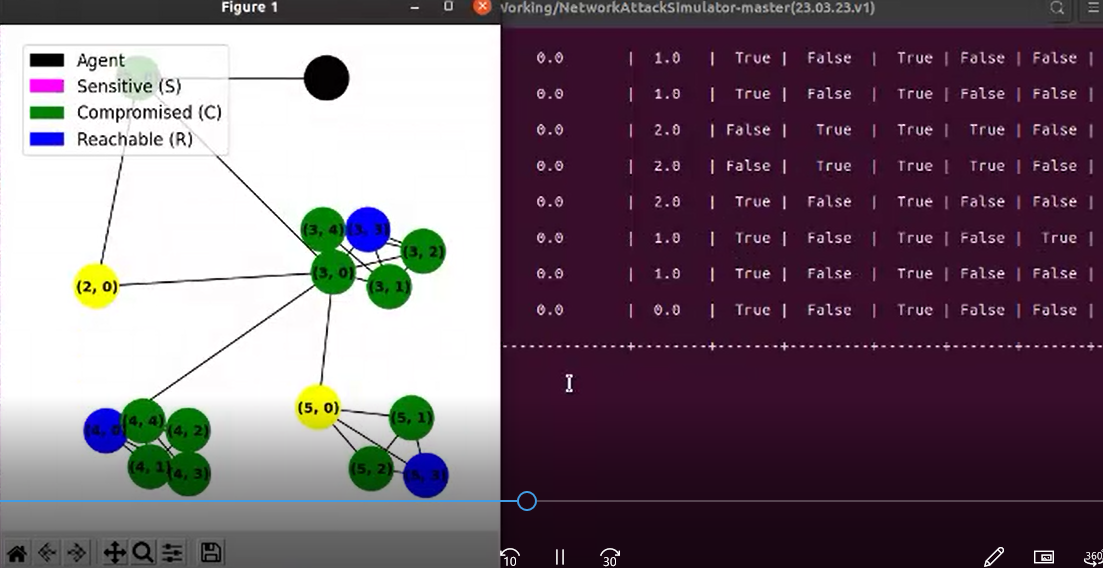
To continue test, you should press ‘Enter’.

Then you can see that agent finds attack path and gets the goal.

Once goal is achieved, you can see the figure which shows the final state of network.

You can also see the transition of network state in the terminal.

If the nodes which are sensitive are converted to red from pink, it shows that the goal is achieved.



1. Ql agent

Command:

$python3 –m nasim.ql [--training\_steps] <scenario>

For example

$python3 –m nasim.ql –training\_steps=20000 tiny.yaml

1. Ql agent with replay

Command:

$python3 –m nasim.ql\_replay [--training\_steps] <scenario>

For example

$python3 –m nasim.ql\_replay --training\_steps=20000 tiny.yaml

For Ql agents, you should using version 0.

1. Appendix (commands used in the test)

🡪medium multi

$python3 -m nasim.dqn\_train --training\_steps=100000 --hidden\_sizes=256 medium-multi-site.yaml dqn\_medium\_muti.pt

$python -m nasim.dqn\_test --hidden\_sizes=256 medium-multi-site.yaml dqn\_medium\_muti.pt

🡪 medium

$python3 -m nasim.dqn\_train --training\_steps=100000 --hidden\_sizes=256 medium.yaml dqn\_medium.pt

$python -m nasim.dqn\_test --hidden\_sizes=256 medium dqn\_medium.pt

🡪small

$ python3 -m nasim.dqn\_train --training\_steps=100000 --hidden\_sizes=128 small.yaml dqn\_small.pt

$ python -m nasim.dqn\_test --hidden\_sizes=128 small dqn\_small.pt

🡪tiny

$ python3 -m nasim.dqn\_train --training\_steps=100000 --hidden\_sizes=64 tiny.yaml dqn\_tiny.pt

$python3 –m nasim.dqn\_test –hidden\_sizes=64 tiny.yaml dqn\_tiny.pt