# SAMSUNG SDS
# 강화학습 실습

# 실습환경소개

# 실습환경소개

## 2. jupyter notebook 환경

```
(base) sds05@ec9986975b81:~$ cd Day5/expl_rnd/
(base) sds05@ec9986975b81:~/Day5/expl_rnd$ conda activate day02
```

→ cd Day5/expl_rnd

→ cd conda activate day02

**전반부 실습인 day02 conda 환경을 사용합니다!**

# 실습환경소개

## 3. Jupyter Notebook 실행

```
(day02) sds05@ec9986975b81:~/Day5/expl_rnd$ jupyter notebook --port 8590
[W 2021-08-27 02:42:56.961 LabApp] 'port' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be
 sure to update your config before our next release.
[W 2021-08-27 02:42:56.961 LabApp] 'port' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be
 sure to update your config before our next release.
[I 2021-08-27 02:42:56.972 LabApp] JupyterLab extension loaded from /home/sdssudo/anaconda3/envs/day02/lib/python3.7/site-pack
ages/jupyterlab
[I 2021-08-27 02:42:56.973 LabApp] JupyterLab application directory is /home/sdssudo/anaconda3/envs/day02/share/jupyter/lab
[I 02:42:56.980 NotebookApp] Serving notebooks from local directory: /home/sds05/Day5/expl_rnd
[I 02:42:56.980 NotebookApp] Jupyter Notebook 6.4.3 is running at:
[I 02:42:56.980 NotebookApp] http://localhost:8590/?token=df2fc204373fcbb0db4d6378a72f5ca82e6259b0ea764e2f
[I 02:42:56.980 NotebookApp]  or http://127.0.0.1:8590/?token=df2fc204373fcbb0db4d6378a72f5ca82e6259b0ea764e2f
[I 02:42:56.980 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 02:42:56.985 NotebookApp] No web browser found: could not locate runnable browser.
[C 02:42:56.985 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/sds05/.local/share/jupyter/runtime/nbserver-25195-open.html
    Or copy and paste one of these URLs:
        http://localhost:8590/?token=df2fc204373fcbb0db4d6378a72f5ca82e6259b0ea764e2f
```

→ jupyter notebook --port 85{아이디 번호}

ex) jupyter notebook --port 8501

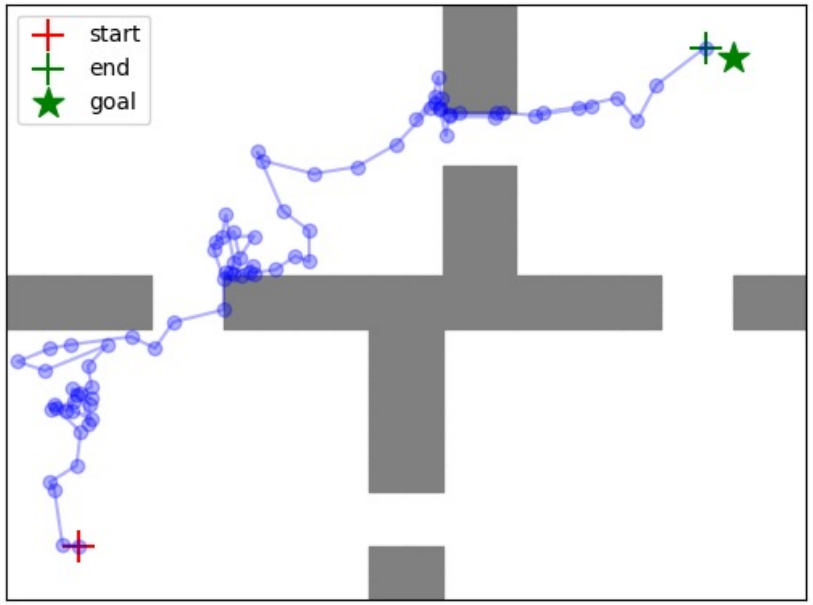
# 실습환경소개

## 6. 접속 완료!

- expl_rnd
- SAC_HER
  - answer
  - core
  - data
  - envs
  - run.ipynb
  - dqn.py
  - dqn_agent.py
  - rl_trainer.py
  - rnd_model.py
  - run_dqn.py

# 실습과제소개

# 실습과제소개




pointmass env

## Exploration with Random Network Distillation

과제 소개:

- RND를 사용해 Pointmass 환경 해결
- Hard 환경에서 40000 step 기준
  -80이상 best mean reward 획득

제출 : kim95175@gmail.com

# 실습과제소개

```
Timestep 40001
Num Episodes 581
Success rate(%) = 70.05
mean reward (100 episodes) -48.190000
best mean reward -48.190000
running time 735.225359
        Train_EnvstepsSoFar : 40001
        Train_EpisodeSoFar : 581
        Num_Episode_reach_the_goal : 407
        Train_AverageReturn : -48.19
        Train_BestReturn : -48.19
        TimeSinceStart : 735.2253589630127
        Training Loss : 0.12992346286773682
        Exploration Bonus : 0.0
        Exploration Model Loss : 0.03259282931685448
        Exploitation weight : 1.0
        Exploration weight : 0.0
```

**위와 같이 Log를 캡처하여 보내주시면 됩니다.**

## Exploration with Random Network Distillation

과제 소개:

- RND를 사용해 Pointmass 환경 해결
- Hard 환경에서 40000 step 기준
  -80이상 best mean reward 획득

제출 : kim95175@gmail.com

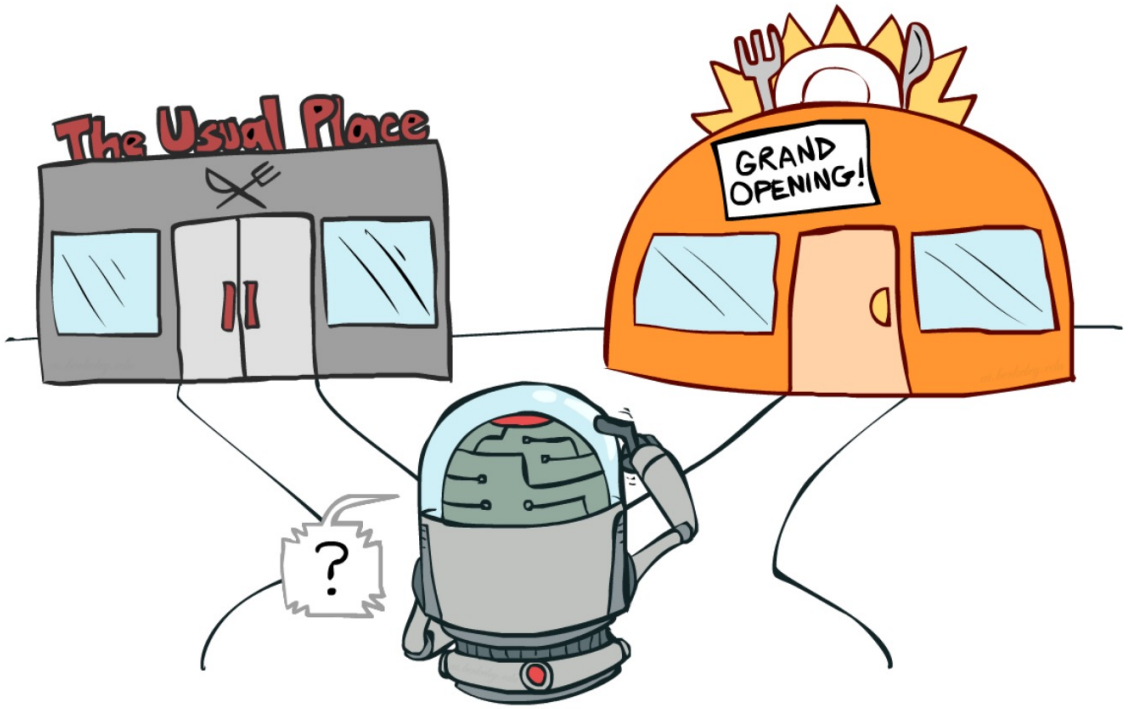

Exploration
with
RND

# Exploration

## Exploration with Random Network Distillation

실습 목표 :

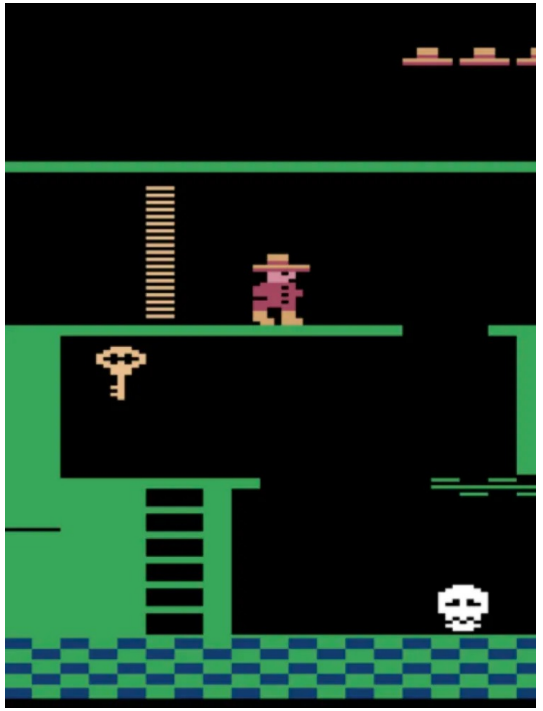
- RND를 사용해 Pointmass 환경 해결
- Exploration에 대한 이해



# Hard-Exploration Problem


Pacman

**Dense Reward**



Montezuma's revenge

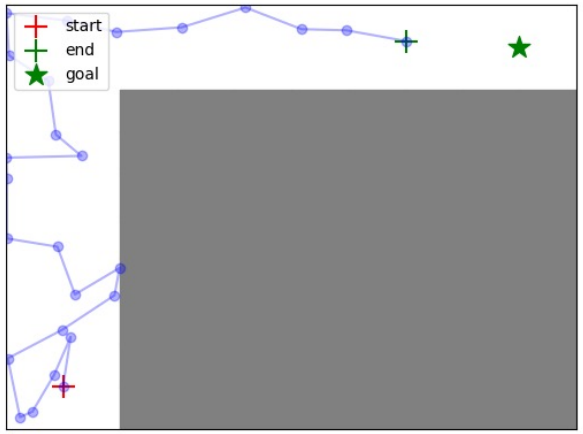
**Sparse Reward**

# 환경 소개 – Pointmass

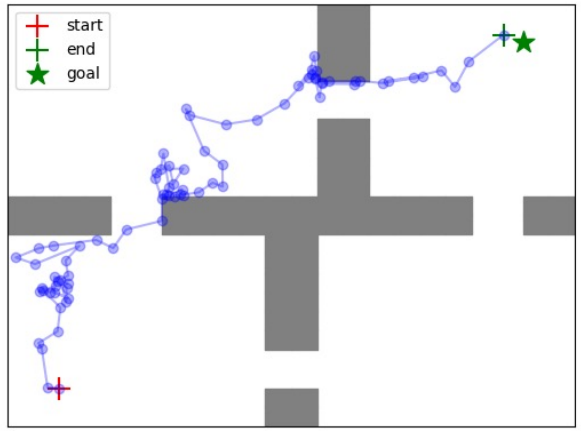
- 목표 : 움직여서 **GOAL**도달

- 매 **step reward -1**
- **Goal**에 도달하면 **reward 0 -> done**
- **100 step** 도달 시 **-> done**

- 시작점과 **Goal**은 항상 고정.

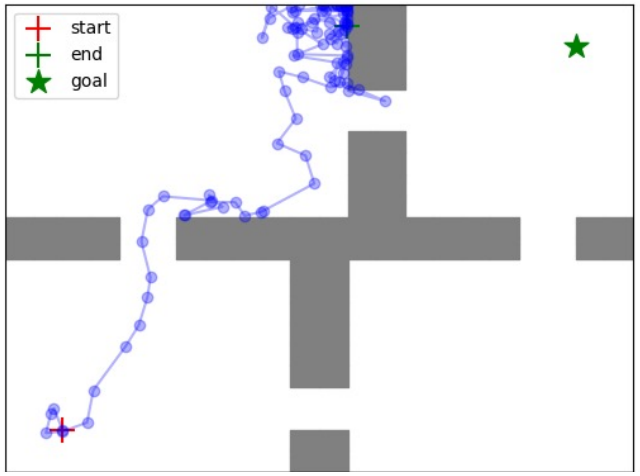



Easy

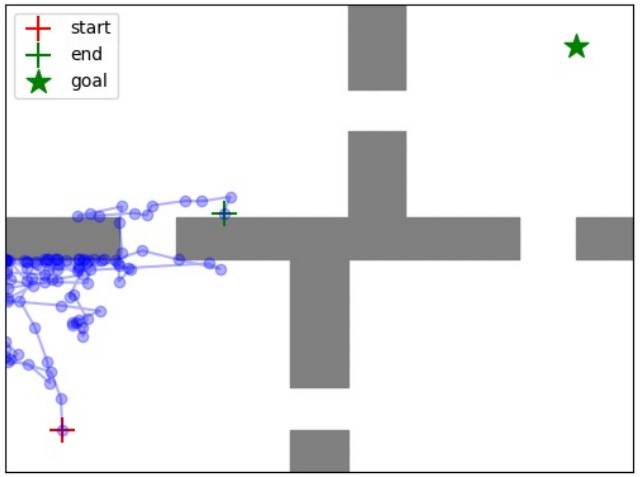



Hard

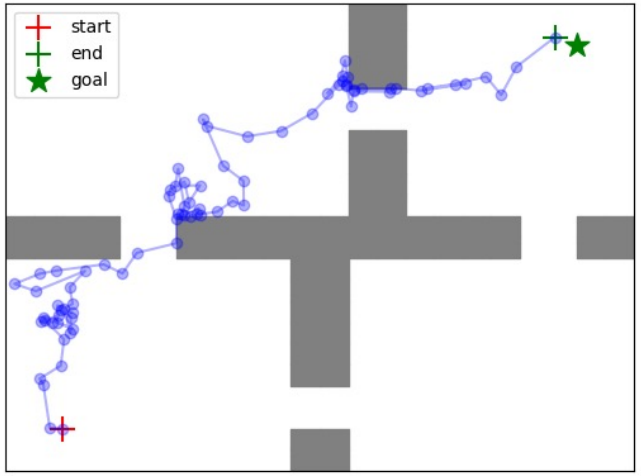
# 환경 소개 – Pointmass




Fail




Fail

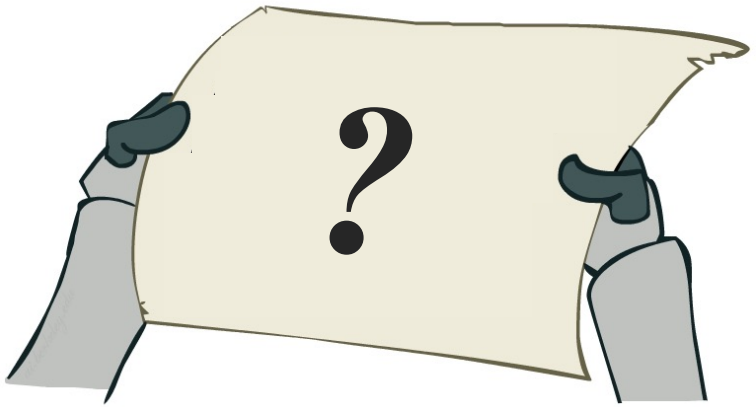



Success

## State
= Box(2,)
(현재 x 좌표, 현재 y좌표)

## Action
= Discrete(5)
(·↓↑←→)

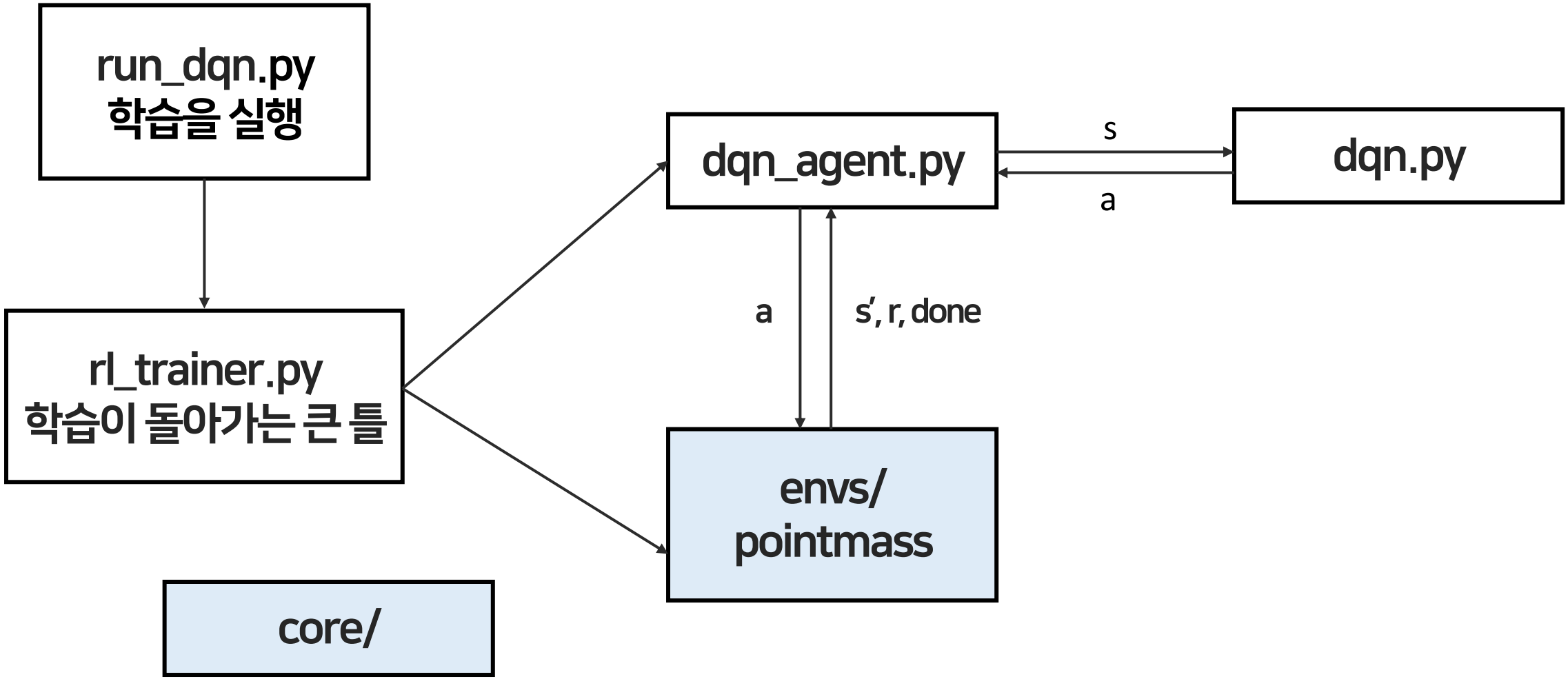
# 환경 소개 – Pointmass




**No Reward**
**Until the goal**

**Stochastic**
**Action**

# 코드 구성


run_dqn.py
학습을 실행
rl_trainer.py
학습이 돌아가는 큰 틀
core/
dqn_agent.py
s
a
dqn.py
a
s', r, done
envs/
pointmass

# 실습 : Only DQN


**Day5/Templete/expl_rnd/run.ipynb**

# Epsilon greedy

$\epsilon - greedy$

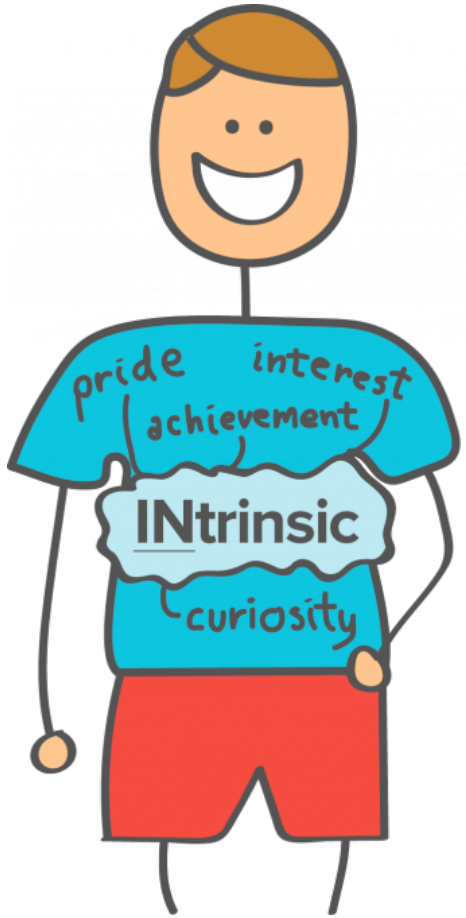
**= Random Action**

왜 epsilon greedy 로 환경을 해결할 수 없을까?

```
p = random()

if p < ε:
   pull random action
else:
   pull current-best action
```

# Intrinsic Reward

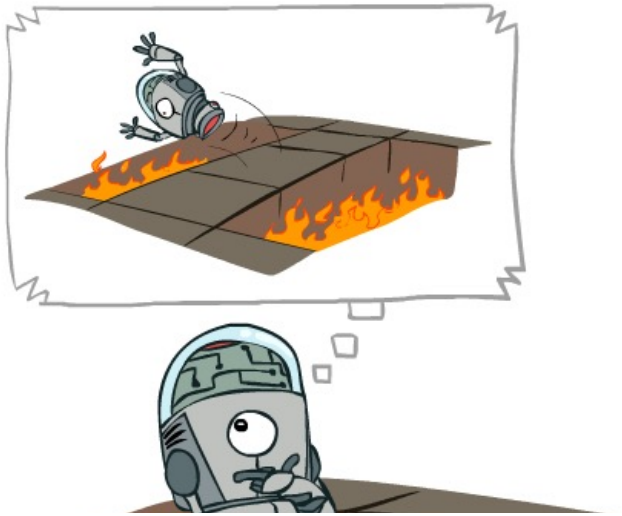




환경에서 주어지는 **reward** 가 없기 때문에, **Intrinsic reward**를 사용해야함.

**Intrinsic reward**를 '어떻게' 설정해야 할까?

# Intrinsic reward

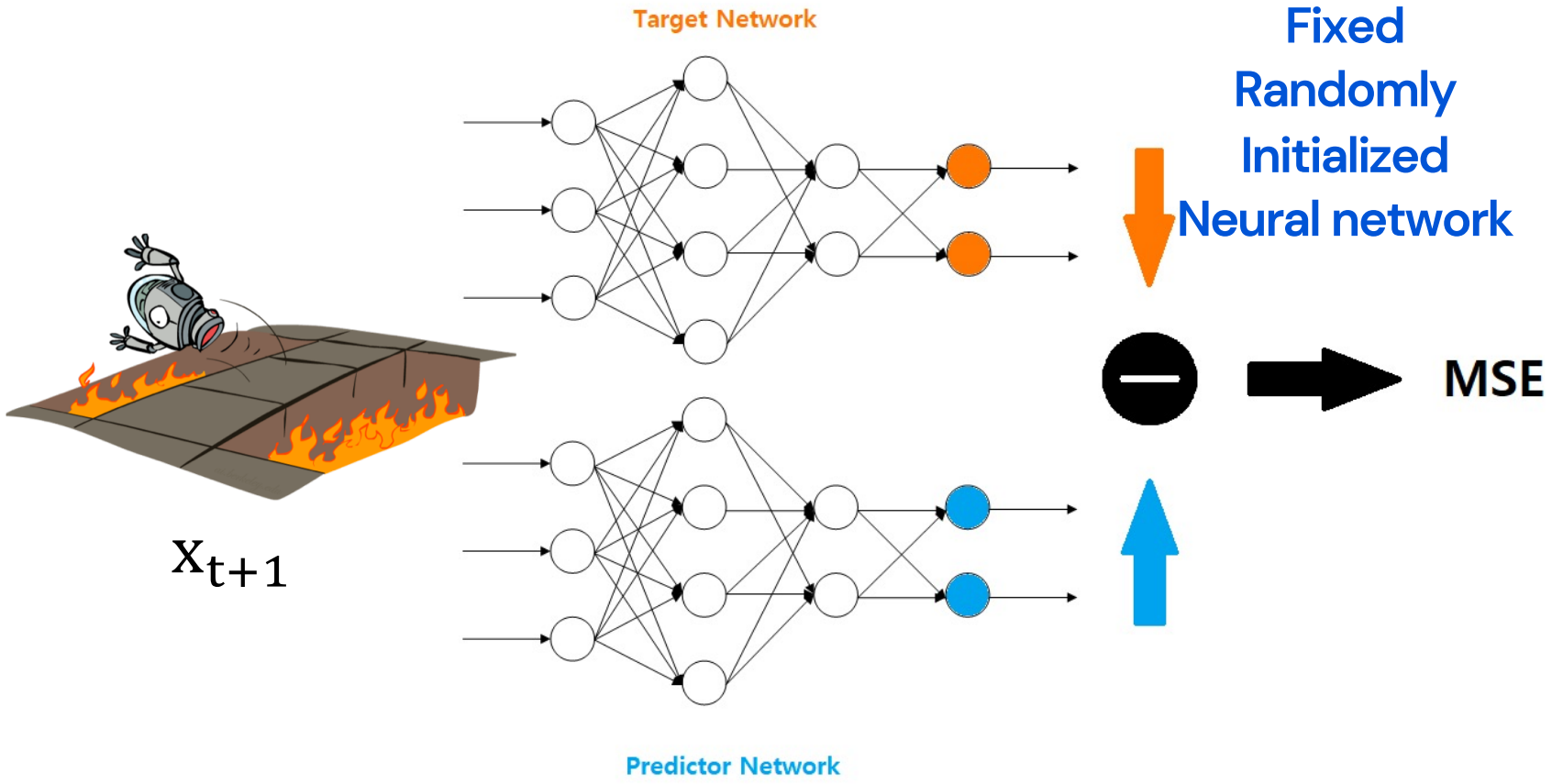


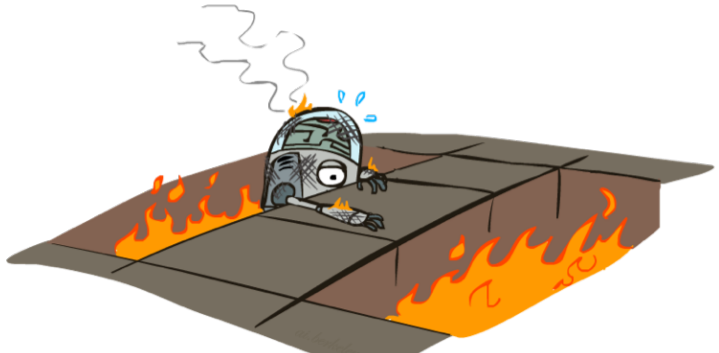
$x_{t+1}$ ( **prediction** )

==

$x_{t+1}$ ( **real** )

# Intrinsic reward



# Exploration

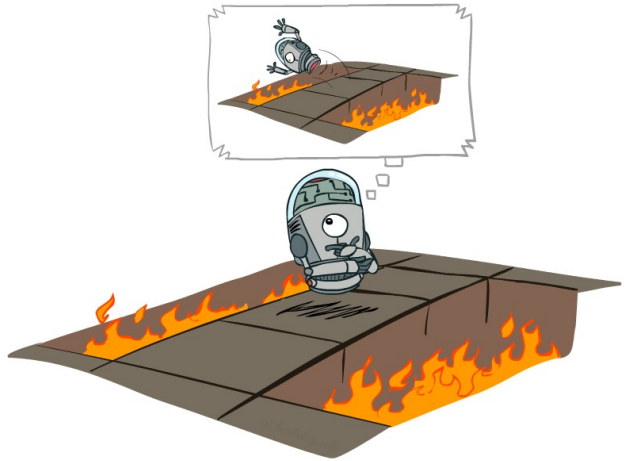
**가보지 않은 곳(state)으로 가보자**

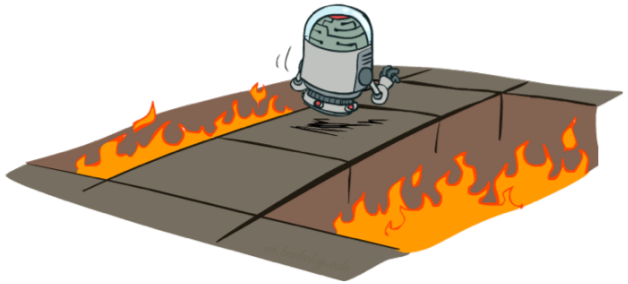
→ 잘 아는 곳은 적당히 지나가자.

→ 현재 있는 곳에 대해 얼마나 잘 아는가?

( 얼마나 많이 와봤는가?)

→ 다음 **state**를 예측할 수 있는가?





# Exploration

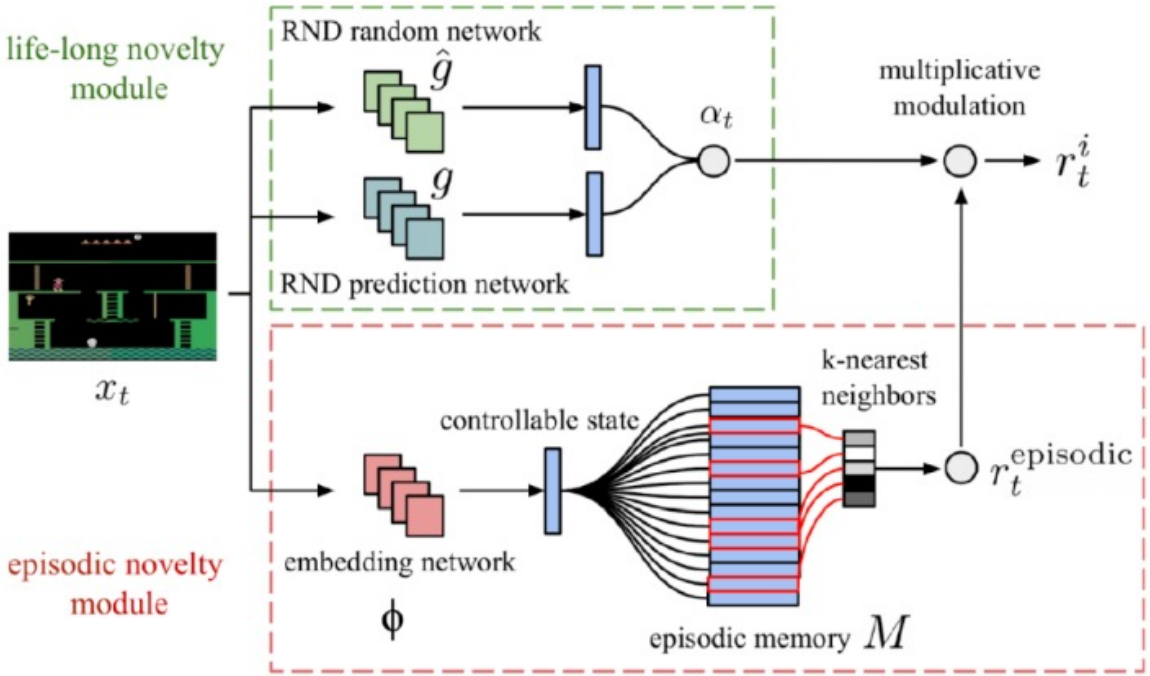
**가보지 않은 곳(state)으로 가보자**

→ 잘 아는 곳은 적당히 지나가자.

→ 현재 있는 곳에 대해 얼마나 잘 아는가?

( 얼마나 많이 와봤는가?)

→ 왔던 **state**를 **memory**에 기록하자

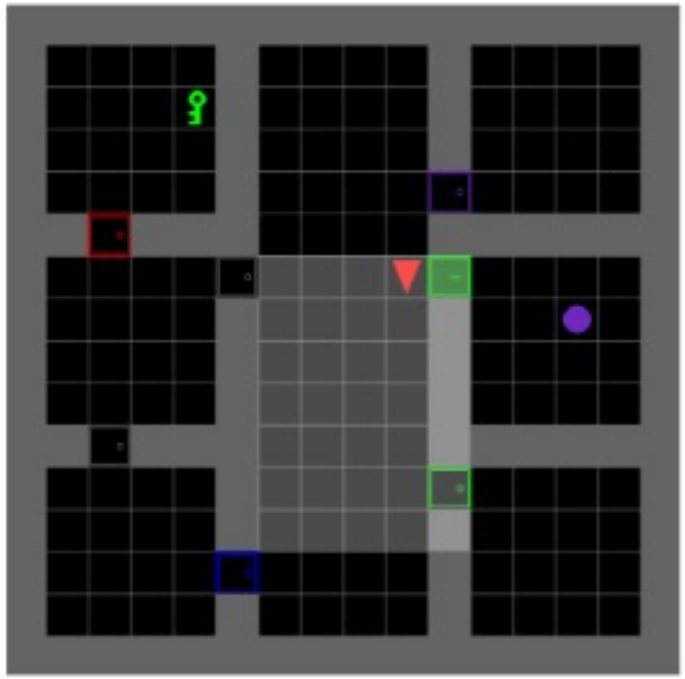


# Exploration : AGAC

**가보지 않은 곳(state)으로 가보자**

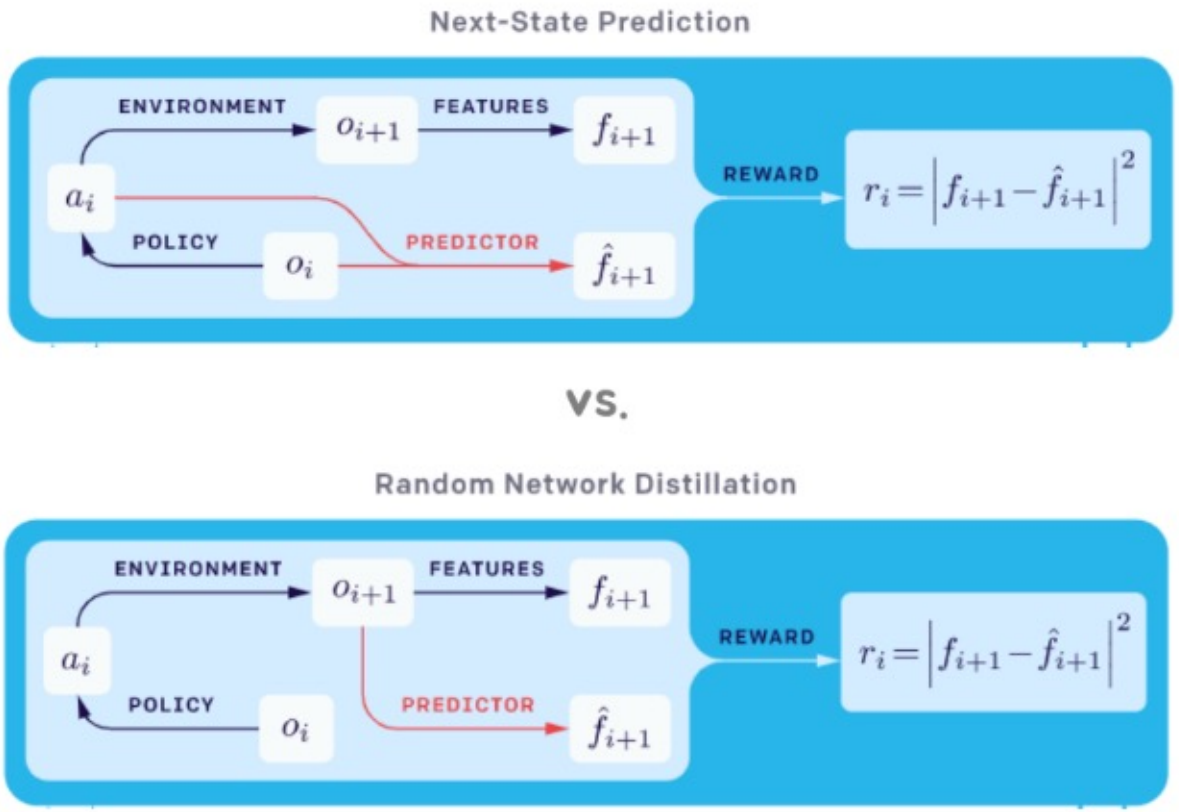
→ 하지 않았던 행동(action)을 해보자

→ 다음 행동을 예측하는 **adversary** 를 속이자.



MiniGrid-KeyCorridor

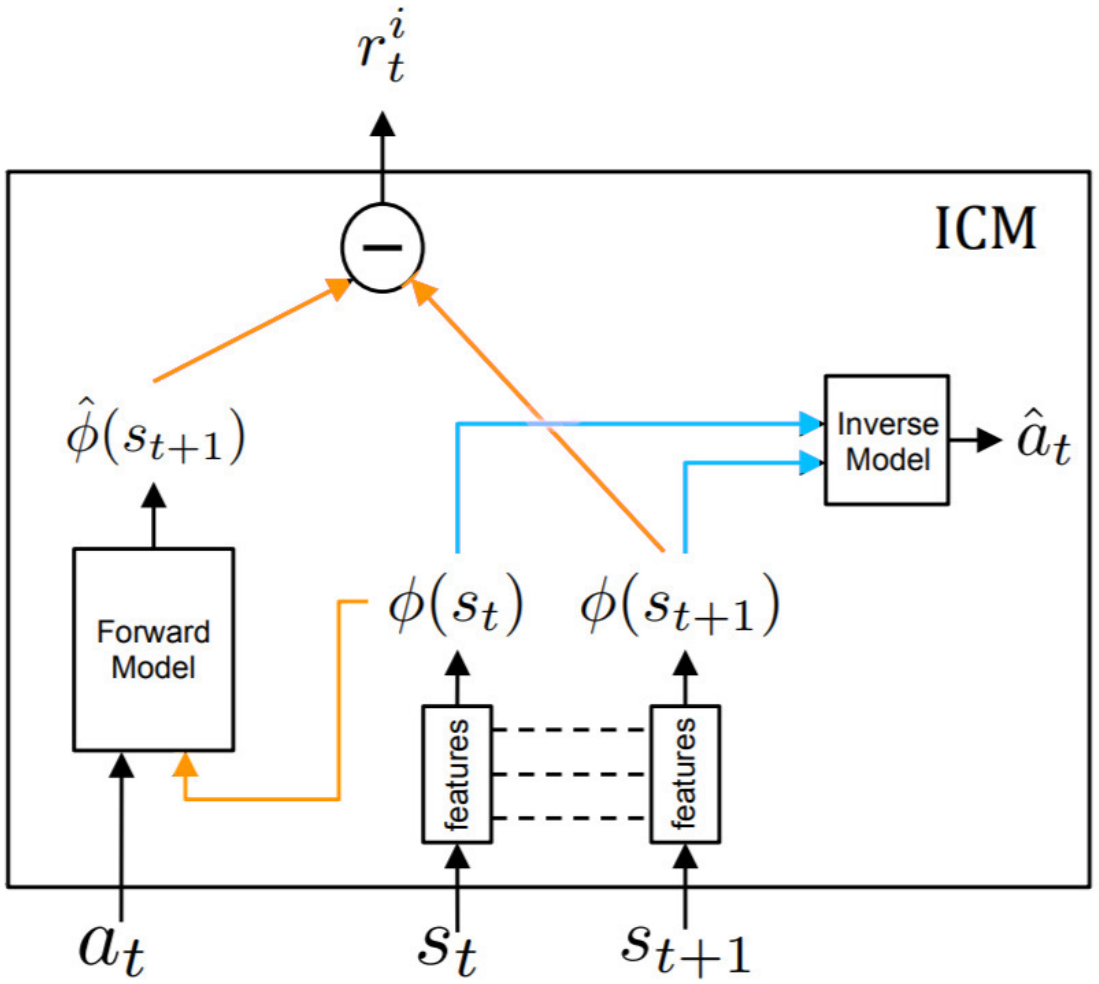
# state prediction vs random network




Predict Next State
which can be stochastic.

Predict 'Random Network'
which is deterministic.

# state prediction vs random network




# 코드 구성

run_dqn.py
학습을 실행

rl_trainer.py
학습이 돌아가는 큰 틀

core/

dqn_agent.py

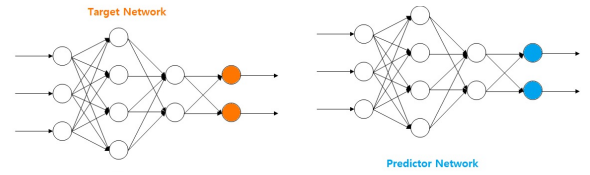
dqn.py

envs/
pointmass

s'

Intrinsic
Reward

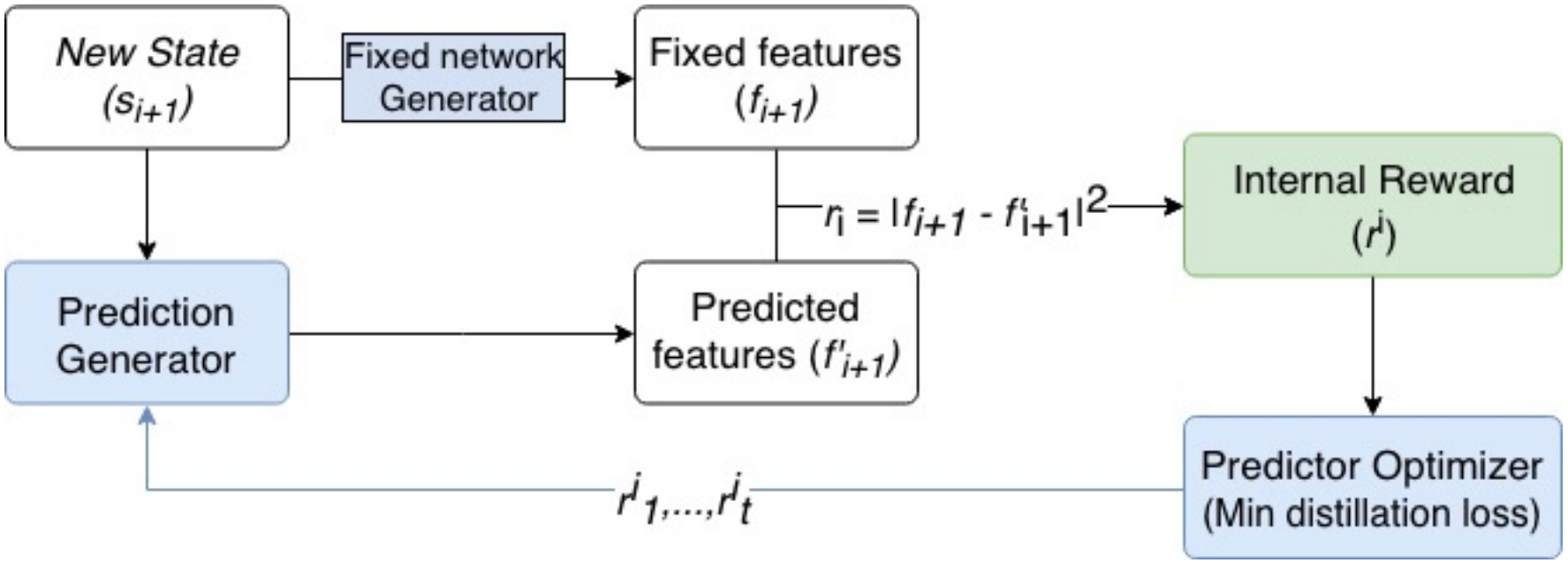
rnd_model.py



# RND to implement

**for** $i = 1$ **to** $N$ **do**
  **for** $j = 1$ **to** $K$ **do**
    sample $a_t \sim \pi(a_t|s_t)$
    sample $s_{t+1}, e_t \sim p(s_{t+1}, e_t|s_t, a_t)$
    calculate intrinsic reward $i_t = \|\hat{f}(s_{t+1}) - f(s_{t+1})\|^2$
    add $s_t, s_{t+1}, a_t, e_t, i_t$ to optimization batch $B_i$
    Update reward normalization parameters using $i_t$
    t += 1
  **end for**
  Normalize the intrinsic rewards contained in $B_i$
  Calculate returns $R_{I,i}$ and advantages $A_{I,i}$ for intrinsic reward
  Calculate returns $R_{E,i}$ and advantages $A_{E,i}$ for extrinsic reward
  Calculate combined advantages $A_i = A_{I,i} + A_{E,i}$
  Update observation normalization parameters using $B_i$
  **for** $j = 1$ **to** $N_{\text{opt}}$ **do**
    optimize $\theta_\pi$ wrt PPO loss on batch $B_i, R_i, A_i$ using Adam
    optimize $\theta_{\hat{f}}$ wrt distillation loss on $B_i$ using Adam
  **end for**
**end for**

# 실습 : RND




**Day5/Template/expl_rnd/**


Thank You