좋습니다. AI\_RTC 폴더에서 VS Code를 여셨다면, 이제 파이썬 실행 환경을 설정하고 스크립트를 실행할 차례입니다.

다음 5단계를 순서대로 진행하시면 됩니다.

**1. 파이썬 스크립트 파일 생성**

VS Code의 파일 탐색기에서 AI\_RTC 폴더에 새 파일을 만듭니다. (예: main.py)

그리고 그 파일에 방금 완성한 파이썬 코드를 전부 붙여넣고 저장하세요.

**2. 가상 환경 생성 및 활성화**

터미널에서 직접 패키지를 설치하는 것보다, 프로젝트별로 독립된 **가상 환경**을 만드는 것이 좋습니다.

1. VS Code에서 터미널을 엽니다. (단축키: Ctrl + ```)
2. 터미널에 다음 명령어를 입력하여 .venv라는 이름의 가상 환경을 만듭니다.

Bash

python -m venv .venv

*(만약 python이 안되면 python3로 시도하세요)*

1. 가상 환경을 **활성화**합니다.
   * **Windows (PowerShell 또는 CMD):**

PowerShell

.\.venv\Scripts\activate

* + **macOS / Linux (bash/zsh):**

Bash

source .venv/bin/activate

1. (터미널 프롬프트 앞에 (.venv)가 보이면 성공입니다.)

**3. 필요 라이브러리 설치**

논문 재현에 필요한 핵심 라이브러리(torch, numpy, scikit-learn)를 설치해야 합니다.

1. AI\_RTC 폴더에 requirements.txt 파일을 만듭니다.
2. 해당 파일에 필요한 라이브러리 목록을 적고 저장합니다.

**requirements.txt**

Plaintext

numpy

scikit-learn

torch

1. 가상 환경이 활성화된 터미널에서 pip을 사용해 이 파일 목록을 한 번에 설치합니다.

Bash

pip install -r requirements.txt

**4. VS Code 인터프리터 설정**

VS Code가 방금 설치한 가상 환경을 인식하도록 설정해야 합니다.

1. Ctrl + Shift + P를 눌러 **명령 팔레트**를 엽니다.
2. Python: Select Interpreter (또는 파이썬: 인터프리터 선택)를 검색하고 선택합니다.
3. 목록에서 방금 생성한 가상 환경 (.venv 경로가 포함된 옵션)을 선택합니다. (예: ./.venv/bin/python)

이제 VS Code의 "실행" 버튼이나 디버거가 올바른 환경과 라이브러리를 사용하게 됩니다.

**5. 스크립트 실행 🚀**

모든 준비가 끝났습니다. 터미널에서 스크립트를 실행합니다.

코드가 argparse를 사용하므로, 터미널에서 직접 실행하며 인자를 조절하는 것이 좋습니다.

**A. 테스트 실행 (빠름)**

코드가 오류 없이 돌아가는지 빠르게 확인합니다.

(에피소드 10회, 부트스트랩 50회, ARL1 반복 20회)

Bash

python main.py --episodes 10 --n\_boot 50 --R 20

**B. 논문 재현 실행 (매우 느림)**

논문과 유사한 수준으로 실행합니다. (R=5001, --n\_boot와 --episodes는 클수록 좋음)

* **옵션 1 (Table 2/3):** 윈도우 {5, 10, 15}

Bash

python main.py --action\_set "5,10,15" --episodes 300 --n\_boot 5000 --R 500

* **옵션 2 (Table 4/5):** 윈도우 {3, 10, 17}

Bash

python main.py --action\_set "3,10,17" --episodes 300 --n\_boot 5000 --R 500

* **(GPU 사용 시):** 만약 PyTorch가 CUDA를 지원하는 GPU(NVIDIA)가 있다면 --device cuda를 추가하면 RL 학습 속도가 빨라질 수 있습니다.

Bash

python main.py --action\_set "5,10,15" --episodes 300 --n\_boot 5000 --R 500 --device cuda

**경고:** n\_boot와 R 값이 크면 **실행에 수 시간 또는 그 이상**이 걸릴 수 있습니다. 먼저 테스트 실행으로 확인한 후, 시간을 여유롭게 잡고 재현 실행을 시작하세요.

GitHub 푸시 완료 축하합니다.

이제 Ubuntu 서버에서 프로젝트 폴더를 생성(GitHub에서 복제)하고 환경을 설정하는 단계입니다.

SSH로 Ubuntu 서버에 접속한 후, 다음 6단계를 순서대로 실행하세요.

**1. (선택) Python 설치 확인 및 설치**

먼저, Ubuntu에 python3, pip, venv가 설치되어 있는지 확인하고, 없다면 설치합니다.

Bash

sudo apt update

sudo apt install python3-pip python3-venv

**2. 프로젝트 폴더 생성 (Git Clone)**

GitHub 저장소를 clone하면 ai\_rtc라는 이름의 프로젝트 폴더가 자동으로 생성됩니다.

Bash

# (홈 디렉토리 등 원하는 위치로 이동하세요)

cd ~

# GitHub 저장소를 복제 (이것이 "프로젝트 폴더 만들기"입니다)

git clone https://github.com/chapter5k/ai\_rtc.git

**3. 프로젝트 폴더로 이동**

방금 생성된 ai\_rtc 폴더로 이동합니다.

Bash

cd ai\_rtc

**4. 가상 환경 생성**

현재 폴더(ai\_rtc) 내부에 .venv라는 이름의 Python 가상 환경을 만듭니다.

Bash

python3 -m venv .venv

**5. 가상 환경 활성화**

생성한 가상 환경을 현재 터미널 세션에서 활성화합니다.

Bash

source .venv/bin/activate

(터미널 프롬프트 앞에 (.venv)가 나타나면 성공입니다.)

**6. 라이브러리 설치**

requirements.txt 파일을 이용해 가상 환경 안에 필요한 라이브러리를 모두 설치합니다.

Bash

pip install -r requirements.txt

이제 모든 환경 설정이 완료되었습니다. ai\_rtc 폴더는 GitHub과 동기화되었고, .venv 가상 환경에는 torch, sklearn 등이 모두 설치되었습니다.

다음 단계는 tmux를 사용하여 스크립트를 실행하는 것입니다.

1) 논문 “정확 재현” (CPU OOB, 기본값)

python rl\_rtc\_reproduction.py --device cuda --rf\_backend sklearn \

--episodes 100 --n\_boot 2000 --R 500 --action\_set 5,10,15

2) “가속 모드” (GPU RF + CV-OOB 근사)

python rl\_rtc\_reproduction.py --device cuda --rf\_backend cuml\_cv \

--episodes 100 --n\_boot 2000 --R 500 --action\_set 5,10,15

좋아요 👍 이제 v4 스크립트는 완성된 상태이니, 아래 단계대로 실행하면 됩니다.

**🧩 1. 환경 준비**

**✅ 필수 패키지 설치**

Python 3.9+ 기준으로 (CPU 모드)

pip install numpy scikit-learn torch tqdm

**⚙️ GPU 가속 (옵션)**

cuML을 쓰려면 **RAPIDS** 환경이 필요합니다. (A100 환경이라면 conda로 설치 권장)

# CUDA 12.0 예시 (A100 호환)

conda install -c rapidsai -c conda-forge -c nvidia \

cuml cupy python=3.10 rapids=24.04 cuda-version=12.0 -y

설치가 끝나면 python -c "import cuml; print('cuML OK')"로 정상 여부를 확인하세요.

**🚀 2. 실행 방법**

**기본 CPU 실행 (논문과 동일한 정확 재현 모드)**

python rl\_rtc\_reproduction\_v4.py \

--device cpu \

--rf\_backend sklearn \

--episodes 30 \

--n\_boot 800 \

--R 100 \

--action\_set 5,10,15

* **RandomForest는 CPU(scikit-learn)** OOB 방식으로 정확 재현
* CNN 정책 네트워크는 CPU 연산
* ETA는 벤치마크 기반 정적 예측값이 출력됩니다

**GPU 가속 실행 (RAPIDS + PyTorch CUDA)**

python rl\_rtc\_reproduction\_v4.py \

--device cuda \

--rf\_backend cuml\_cv \

--episodes 100 \

--n\_boot 2000 \

--R 500 \

--action\_set 5,10,15 \

--guess\_arl1 15

* **cuML RandomForest**를 GPU에서 수행 (교차검증 기반 OOB 근사)
* **CNN 정책 네트워크**는 PyTorch CUDA에서 동작
* **멀티-GPU(A100 ×2)** 환경이면 자동으로 nn.DataParallel 활성화됨
* ETA는 시작 시에 한 번만 계산되어 출력됩니다

**🧭 3. 실행 중 출력 예시**

실행하면 아래와 비슷한 로그가 표시됩니다:

============================================================

RL-RTC 논문 재현 시뮬레이션 시작 (v4)

============================================================

Device (CNN): cuda

RF Backend: cuml\_cv

Action Set: 5,10,15

RL Episodes: 100

CL n\_boot: 2000

ARL1 R: 500

Seed: 2025

Started at: 2025-10-20 18:32:00

[ETA] 전체 파이프라인 예상 소요 시간 벤치마크 수행 중...

Bench RF(n=300): 100%|██████████| 20/20 [00:23<00:00, 1.16s/it]

[ETA] 총 예상 소요 시간 : 2:17:35

[ETA] 예상 완료 시간 (ETA): 2025-10-20 20:49:35

============================================================

[작업 1/3] CL(ARL0=200) 보정 시작 (n\_boot=2000)...

CL Boot (w=5): 45%|████▌ | ...

**💾 4. 실행 후 결과 확인**

출력 폴더 outputs/에 결과 CSV가 자동 저장됩니다:

outputs/

├── arl1\_scenI\_5-10-15.csv

└── arl1\_scenII\_5-10-15.csv

각 파일에는 다음 열이 들어 있습니다:

lambda,arl1\_mean,arl1\_std

0.5, 212.5, 45.3

...

**⚡ 5. 추가 팁**

* **빠른 테스트용**: --n\_boot 50 --episodes 5 --R 10  
  → 약 1~2분이면 전체 파이프라인이 도는 미니 샘플 실행
* **로그 줄이기**: tqdm은 자동으로 ETA를 표시하므로, print 로그가 많을 경우 > logfile.txt 리다이렉트로 정리할 수 있습니다.
* **정확 재현 실험**: 논문 표(Table 2–5)에 맞추려면  
  --rf\_backend sklearn --episodes 300 --n\_boot 5000 --R 500 설정을 추천합니다.

원하신다면, \*\*conda 환경 파일(environment.yml)\*\*도 바로 만들어드릴까요?  
(torch, scikit-learn, cupy, cuml, tqdm 포함)