실행 방법

conda activate rapids-env

cd ai\_rtc

# CPU(sklearn)로 빠르게 드라이런

python main\_patched.py \

--device cpu \

--episodes 20 \

--n\_boot 400 \

--R 50 \

--cheap\_mode \

--rf\_train\_estimators 80 \

--rf\_full\_estimators 500 \

--rf\_feature\_frac 0.6

# GPU(cuML) 경로 (RAPIDS 설치 환경)

python main.py --device cuda --rf\_backend cuml\_cv --episodes 100 --n\_boot 2000 --R 500 --action\_set 5,10,15 --guess\_arl1 15

python main.py \   --device cuda \   --rf\_backend cuml\_cv \   --episodes 100 \   --n\_boot 2000 \   --R 500 \   --action\_set 5,10,15 \   --guess\_arl1 15

짧은 검증 용 실행

python main.py --device cuda --rf\_backend cuml\_cv --episodes 5 --n\_boot 400 --R 50

**출력물**

* outputs/cl\_table\_actions-\*.csv
* outputs/rl\_training\_log.csv
* outputs/policy\_snapshot.pt
* outputs/arl1\_scenI\_\*.csv, outputs/arl1\_scenII\_\*.csv
* outputs/summary.json (경로와 하이퍼파라미터, ARL1 요약 포함)

**💡 PC를 꺼도 스크립트를 계속 실행하는 방법**

**2. 권장하는 방법: tmux 사용하기**

tmux는 "가상 터미널"을 만들고 그 안에 접속하는 방식입니다. PC를 껐다가 나중에 다시 접속해도 그 "가상 터미널"은 그대로 살아있습니다.

1. **tmux 설치 (서버에 한 번만):**

sudo apt install tmux

(또는 yum install tmux 등)

1. **tmux 세션 시작:**

tmux new -s rtc\_session

(rtc\_session은 원하는 이름)

conda activate rapids-env

1. **스크립트 실행 (새로 열린 tmux 창 안에서):**

python main.py --device cuda --rf\_backend cuml\_cv ...

(여기서는 &나 nohup이 필요 없습니다. 그냥 실행하세요.)

1. **tmux 세션에서 나오기 (세션은 유지됨):**
   * Ctrl + b 를 눌렀다 뗀 후, d 키를 누릅니다.

이제 SSH를 종료하거나 PC를 꺼도 rtc\_session은 서버에서 계속 실행됩니다.

1. **나중에 다시 접속하기:**

tmux attach -t rtc\_session

(실행 중인 화면을 그대로 다시 볼 수 있습니다.)

python main.py --device cuda --rf\_backend cuml\_cv --episodes 5 --n\_boot 400 --R 50

**빠른 생성(권장: 이미 policy 파일이 있을 때)**

# RAPIDS 사용 시

python main\_251027\_v2.py \

--device cuda --rf\_backend cuml\_cv \

--episodes 0 --n\_boot 2000 --R 0 \

--policy\_out outputs/policy\_5-10-15.pt

**0) 전체 파이프라인(Phase I + RL학습 + 평가)**

python main\_251027\_v2.py \

--device cuda --rf\_backend cuml\_cv \

--episodes 100 --n\_boot 2000 --R 500 \

--n\_estimators\_eval 150 \

--policy\_out outputs/policy\_5-10-15.pt

**1) Phase I만 (CL 보정만 수행하고 종료)**

* 목적: S0\_ref.npy, calib\_map.pkl 생성
* 방법: RL 학습/평가 스킵 → --episodes 0 --R 0

python main\_251027\_v2.py \

--device cuda --rf\_backend cuml\_cv \

--episodes 0 --n\_boot 2000 --R 0 \

--policy\_out outputs/policy\_5-10-15.pt

**2) Phase II — RL 학습만 (평가 스킵)**

* 목적: 정책만 다시 학습(.pt 저장), 평가(ARL1)는 건너뜀
* 주의: 현재 스크립트는 Phase I을 항상 먼저 실행합니다. 시간을 줄이려면 **CL 부트스트랩을 최소화**(--n\_boot 1).

python main\_251027\_v2.py \

--device cuda --rf\_backend cuml\_cv \

--episodes 100 --n\_boot 1 --R 0 \

--n\_estimators\_eval 150 \

--policy\_out outputs/policy\_5-10-15.pt

**3) Phase II — 평가만 (이미 학습된 policy로 ARL1만 산출)**

* 목적: 기존 정책으로 ARL1 테이블만 다시 계산
* 방법: RL 학습 스킵(--episodes 0), **정책 로드**(--policy\_in ...pt),  
  Phase I은 어차피 돌아가므로 빠르게 하려면 --n\_boot 1.

python main\_251027\_v2.py \

--device cuda --rf\_backend cuml\_cv \

--episodes 0 --n\_boot 1 \

--R 500 --n\_estimators\_eval 150 \

--policy\_in outputs/policy\_5-10-15.pt

**4) 빠른 smoke-test(최소 시간으로 파이프라인 확인)**

python main\_251027\_v2.py \

--device cuda --rf\_backend cuml\_cv \

--episodes 1 --n\_boot 1 --R 1 \

--n\_estimators\_eval 50 \

--policy\_out outputs/policy\_5-10-15.pt

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)

conda: command not found 오류는 1단계(deactivate)를 수행한 후에도 계속 발생할 수 있습니다.

이는 (.venv) 환경을 나온 것은 맞지만, **시스템에 Anaconda 또는 Miniconda가 설치되어 있지 않거나**, 설치되었더라도 현재 셸(terminal)에 conda 명령어의 경로가 등록되지 않았다는 의미입니다.

venv와 conda는 서로 다른 별개의 가상환경 시스템입니다.

**해결 방법**

**1순위: 터미널(셸) 재시작 (가장 가능성 높은 해결책)**

**만약 방금 Anaconda/Miniconda를 설치했다면,** 설치 프로그램이 셸의 설정 파일(예: .bashrc)을 변경했을 것입니다. 이 변경 사항을 적용하려면 **터미널을 껐다가 다시 켜야 합니다.**

1. 현재 터미널 창을 닫습니다.
2. 새 터미널 창을 엽니다.
3. conda activate rapids-env (또는 conda create ...) 명령어를 다시 시도합니다.

**2순위: Anaconda/Miniconda 설치**

터미널을 재시작해도 conda: command not found가 계속 발생한다면, **시스템에 Conda가 아예 설치되지 않은 것입니다.**

Ubuntu 환경이므로, 다음 단계를 따라 Miniconda(가벼운 버전의 Anaconda)를 설치하세요.

1. **Miniconda 설치 스크립트 다운로드:**

Bash

wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh

1. **설치 스크립트 실행:**

Bash

bash Miniconda3-latest-Linux-x86\_64.sh

1. **설치 진행:**
   * 라이선스 동의 (Enter 키를 눌러 읽고, yes 입력)
   * 설치 경로 확인 (대부분 Enter)
   * [가장 중요] conda init 실행 여부:

설치 마지막에 Do you wish the installer to initialize Miniconda3 by running conda init?라고 묻는 메시지가 나옵니다.

반드시 yes를 입력하고 Enter를 누르세요.

1. 터미널 재시작:

설치가 완료되면, .bashrc 파일이 수정되었으므로 터미널을 껐다가 다시 켜야 합니다.

1. Conda 명령어 재시도:

새 터미널에서 conda create -n rapids-env python=3.10 명령어를 다시 실행해 보세요.

아니요, 그 순서는 **잘못되었습니다.** Conda 환경과 .venv 가상 환경을 **동시에 활성화하면 안 됩니다.** 💥 두 환경은 서로 충돌하게 되어, 어떤 라이브러리를 사용해야 할지 시스템이 혼란스러워하고 결국 RAPIDS (cuml, cupy) 라이브러리를 찾지 못해 오류가 발생할 가능성이 매우 높습니다.

**올바른 방법: Conda 환경 하나만 사용하기**

RAPIDS를 Conda 환경(rapids-env라고 가정)에 설치하셨으므로, **모든 필요한 라이브러리를 그 Conda 환경 안에 설치**하고 해당 환경만 활성화해야 합니다.

1. **Conda 환경 활성화:**

Bash

conda activate rapids-env

1. **(필수) Conda 환경에 나머지 라이브러리 설치:**
   * **중요:** .venv에서 사용했던 requirements.txt를 Conda 환경에서도 사용하여 **torch, sklearn, numpy, tqdm 등을 rapids-env 안에 설치**해야 합니다.
   * ai\_rtc 폴더로 이동한 후, rapids-env가 활성화된 상태에서 실행하세요.

Bash

cd ~/ai\_rtc

pip install -r requirements.txt

*(Conda 환경 내에서도 pip 사용 가능)*

1. **스크립트 실행:**
   * rapids-env가 활성화된 상태에서, ai\_rtc 폴더 안에서 스크립트를 실행합니다. **.venv는 절대 활성화하지 마세요.**
   * tmux 사용을 권장합니다.

Bash

# tmux 세션 시작 (이미 있다면 tmux attach -t <세션이름>)

tmux new -s rapids\_run

# --- (새로운 터미널 창) ---

# Conda 환경 활성화 (pip install 후라면 이미 활성화 상태일 수 있음)

**conda activate rapids-env**

# GPU 백엔드로 스크립트 실행

**python3 main.py --action\_set "5,10,15" \**

**--episodes 300 \**

**--n\_boot 5000 \**

**--R 500 \**

**--rf\_backend cuml\_cv \**

**--device cuda \**

**| tee run\_gpu\_5\_10\_15.log**

# tmux 세션에서 나오기: Ctrl+b 누른 후, d 누르기

**요약**

* Conda 환경(rapids-env)에 RAPIDS **및** torch, sklearn, numpy, tqdm을 모두 설치합니다.
* 스크립트 실행 전에는 **오직 Conda 환경(conda activate rapids-env)만 활성화**합니다.
* .venv 환경은 사용하지 않습니다 (source .venv/bin/activate 실행 X).