

도커 이미지를 이용한 신호 최적화 실행

2021.09. 16.

1. 이미지 저장소에서 이미지를 다운로드해 온다.

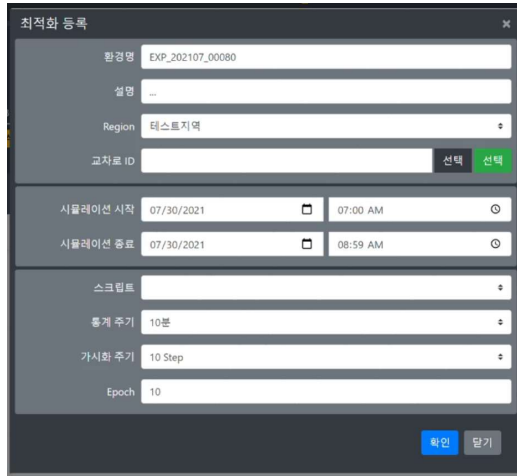
저장소 : <https://hub.docker.com/>

이미지 이름 : images4uniq/optimizer:v0.1a.20210927

\$] sudo docker pull images4uniq/optimizer:v0.1a.20210927

```
root@SALT2:~/z.delete# sudo docker pull hunsooni/optimizer:v0.1a.20210916
v0.1a.20210916: Pulling from hunsooni/optimizer
345e3491a907: Already exists
57671312ef6f: Already exists
5e9250ddb7d0: Already exists
d55423affe62: Pull complete
14b440b54b19: Pull complete
7c6759b9d009: Pull complete
afd1bccdd69a: Pull complete
0a00fa2d8d13: Pull complete
b15151d3278a: Pull complete
d29f0d8c0a5b: Pull complete
6726dade2a64: Pull complete
ba03d26fc6e8: Pull complete
a7c73a38977f: Pull complete
c990f78c08ba: Pull complete
da341e46f61a: Pull complete
189b745d30fe: Pull complete
b294f6e36641: Pull complete
bc5337d42f3a: Pull complete
974f4e099e83: Pull complete
e85a4bb93259: Pull complete
235784ce189b: Pull complete
060a4fb61fbe: Pull complete
ba15ff595892: Pull complete
398101546e75: Pull complete
01f2dbd80605: Pull complete
1b340bb3fa28: Pull complete
eb7d14a4501e: Pull complete
```

2. 데이터(시나리오) 준비



최적화 등록 버튼 누르면

시나리오id/scenario/doan 경로 밑에 시나리오, 데이터 파일 생성
시나리오 파일은 **4p** 참고해서 doan_2021, _ft, _test 3개 생성

아래 4개 파일은 고정으로 필요함

dj_doan_kaist_2h.rou.xml -> 카이스트 7-9시 수요

doan_20210401.edg.xml -> 신호 최적화에서 정보 가져오는 데 필요

doan_20210421.tss.xml -> 4p 참고

doan(without dan).tss.xml -> 신호 최적화에서 정보 가져오는 데 필요

```
root@SALT2:~/z.delete# ls
sample/  sample.tar.gz  volume.org.tar.gz
root@SALT2:~/z.delete# tar xvfz volume.org.tar.gz
volume.org/
volume.org/scenario/
volume.org/scenario/doan/
volume.org/scenario/doan/doan_2021_test.scenario.json
volume.org/scenario/doan/doan.edg.xml
volume.org/scenario/doan/doan.con.xml
volume.org/scenario/doan/doan_2021.scenario.json
volume.org/scenario/doan/doan.nod.xml
volume.org/scenario/doan/doan_2021_ft.scenario.json
volume.org/scenario/doan/doan.tss.xml
volume.org/scenario/doan/dj_doan_kaist_2h.rou.xml
volume.org/scenario/doan/doan_20210401.edg.xml
volume.org/scenario/doan/doan(without dan).tss.xml
volume.org/scenario/doan/doan_20210421.tss.xml
root@SALT2:~/z.delete#
root@SALT2:~/z.delete# cp -r volume.org volume
root@SALT2:~/z.delete#
root@SALT2:~/z.delete# tree volume
volume
├── scenario
└── doan
    ├── dj_doan_kaist_2h.rou.xml
    ├── doan_20210401.edg.xml
    ├── doan_20210421.tss.xml
    ├── doan_2021_ft.scenario.json
    ├── doan_2021.scenario.json
    ├── doan_2021_test.scenario.json
    ├── doan.con.xml
    ├── doan.edg.xml
    ├── doan.nod.xml
    ├── doan.tss.xml
    └── doan(without dan).tss.xml

2 directories, 11 files
root@SALT2:~/z.delete#
```

doan 폴더는 아래 github link에서 받을 수 있음

<https://github.com/etri-city-traffic-brain-jr/traffic-signal-optimization/tree/master/salt-rl/data/envs/salt/doan>

Scenario file 생성시 output fileDir 다르게 해서 3개 생성 필요

1. doan_2021.scenario.json -> mode=train 일 때
2. doan_2021_ft.scenario.json -> mode=simulate 일 때
3. doan_2021_test.scenario.json -> mode=test 일 때

trafficLightSystem은 일단 doan_20210421.tss.xml 고정

```
doan_2021.scenario.json
{
  "scenario": {
    "time": {
      "begin": 0,
      "end": 7200
    },
    "input": {
      "fileType": "SALT",
      "node": "doan.nod.xml",
      "link": "doan.edg.xml",
      "connection": "doan.con.xml",
      "trafficLightSystem": "doan_20210421.tss.xml",
      "route": "dj_doan_kaist_2h.rou.xml"
    },
    "parameter": {
      "minCellLength": 30,
      "vehLength": 5
    },
    "output": {
      "fileDir": "output/r1/",
      "period": 10,
      "level": "link",
      "save": 1
    }
  }
}
```

```
doan_2021_ft.scenario.json
{
  "scenario": {
    "time": {
      "begin": 0,
      "end": 7200
    },
    "input": {
      "fileType": "SALT",
      "node": "doan.nod.xml",
      "link": "doan.edg.xml",
      "connection": "doan.con.xml",
      "trafficLightSystem": "doan_20210421.tss.xml",
      "route": "dj_doan_kaist_2h.rou.xml"
    },
    "parameter": {
      "minCellLength": 30,
      "vehLength": 5
    },
    "output": {
      "fileDir": "output/ft/",
      "period": 10,
      "level": "link",
      "save": 1
    }
  }
}
```

```
doan_2021_test.scenario.json
{
  "scenario": {
    "time": {
      "begin": 0,
      "end": 7200
    },
    "input": {
      "fileType": "SALT",
      "node": "doan.nod.xml",
      "link": "doan.edg.xml",
      "connection": "doan.con.xml",
      "trafficLightSystem": "doan_20210421.tss.xml",
      "route": "dj_doan_kaist_2h.rou.xml"
    },
    "parameter": {
      "minCellLength": 30,
      "vehLength": 5
    },
    "output": {
      "fileDir": "output/test/",
      "period": 10,
      "level": "link",
      "save": 1
    }
  }
}
```

3. 이미지 파일 실행

공유할 디렉토리를
(반드시) /uniq/optimizer의 하부 디렉토리로 마운트

3p에서 생성한 시나리오id 폴더

실행할 이미지

```
> sudo docker run -v /root/z.delete/volume:/uniq/optimizer/io images4uniq/optimizer:v0.1a.20210916
python run.py --mode train --epoch 1 --io-home io --scenario-file-path io/scenario/doan/doan_2021.scenario.json
```

(신호 최적화) 실행

mode == train일 때, doan_2021.scenario.json
mode == test일 때, doan_2021_test.scenario.json
mode == simulate일 때, doan_2021_ft.scenario.json

인자명	설 명
mode	훈련(train)과 추론(test) 여부를 나타내는 'train', 'test', 고정시간 시나리오인 'simulate' 중 하나를 가지며, default 는 'train'
model-num	추론시 사용할 저장된 모델 번호로 default는 0
start-time	시뮬레이션 시작 시간(스텝)으로 default는 0
end-time	시뮬레이션 종료 시간으로 default는 7200
epoch	훈련(학습) 반복 회수로 default는 3000
model-save-period	훈련 중 모델 저장 주기로, default는 20
target-TL	신호 최적화 대상 신호등으로 default는 "SA 101,SA 104,SA 107,SA 111"임. 신호 그룹을 콤마로 연결(예, --target-TL SA 101,SA 104)
reward-func	강화학습의 보상 함수로 ['pn', 'wt', 'wt_max', 'wq', 'wt_SBV', 'wt_SBV_max', 'wt_ABV'] 중 하나, default는 'wt_SBV' pn은 통과 차량 수, wt는 대기시간, wq는 대기 큐 길이, SBV 는 sum에 기반한 보상값을, ABV는 avg에 기반한 보상값을 의미
io-home	최적화 중 생성되는 파일들이 저장되는 곳의 최상위 디렉토리로 최적화 기본 디렉토리인 /uniq/optimizer 로부터 상대 경로로 표시
scenario-file-path	최적화에 사용할 (교통 환경) 시뮬레이션 시나리오 파일로, 최적화 기본 디렉토리인 /uniq/optimizer 로부터 상대 경로로 표시 (예, io/scenario/doan.sample.json)

4. 실행 후 결과 파일

```
root@SALT2:~/z.delete# tree volume
```

```
volume
├── logs
│   └── DDQN_SALT_doan_discrete_PSA__rf_wt_SBV_yp0_actionT_
│       └── 09-27_12-22-14
│           └── events.out.tfevents.1632712934.fbfd221e1542.1.5.v2
├── model
│   └── ddqn
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent0-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent10-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent11-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent12-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent1-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent2-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent3-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent4-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent5-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent6-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent7-trial-0.model.h5
│       ├── PSA__rf_wt_SBV_yp0_actionT_10-agent8-trial-0.model.h5
│       └── PSA__rf_wt_SBV_yp0_actionT_10-agent9-trial-0.model.h5
├── output
│   ├── ft
│   │   └── ft_phase_reward_output.txt
│   ├── rl
│   │   ├── -PeriodicOutput.csv
│   │   └── progress.txt
│   ├── test
│   │   └── rl_phase_reward_output.txt
│   └── train
│       ├── train_epoch_tl_reward.txt
│       └── train_epoch_total_reward.txt
└── scenario
    ├── data
    │   ├── 2b0aaac7-a5c9-495e-9cde-de4897dc37f0
    │   ├── 31497c60-0f97-41d2-b3a3-ad4ad8d3e767
    │   ├── 330675d1-ad79-4719-88c7-e1738feaf567
    │   └── 40d5f8e4-9899-4e39-a689-c2dc32d6cf42
    └── doan
        ├── dj_doan_kaist_2h.rou.xml
        ├── doan_20210401.edg.xml
        ├── doan_20210421.tss.xml
        ├── doan_2021_ft.scenario.json
        ├── doan_2021.scenario.json
        ├── doan_2021_test.scenario.json
        ├── doan.con.xml
        ├── doan.edg.xml
        ├── doan.nod.xml
        ├── doan.tss.xml
        └── doan(without dan).tss.xml
```

Tensorboard용 log
→ 가시화 서버에서는 필요 x

학습된 모델이 저장되는 경로
→ test mode에서 model-num을 이용해서 불러오는 데 활용

output/ft → simulate mode에서 기존 신호에 대한 스텝별 phase와 SALT output 저장

output/rl -> train mode에서 SALT output 저장하는 경로, 가시화 서버에서 사용 x

output/test -> test mode에서 교차로별 phase와 reward, SALT output 저장

Output/train -> train mode에서 epoch별 reward 저장[전체 교차로(total), 교차로별(tl)]

실행마다 시나리오(doan) 폴더가 복사되는 tmp 폴더

입력 시나리오, 데이터 파일 저장 경로