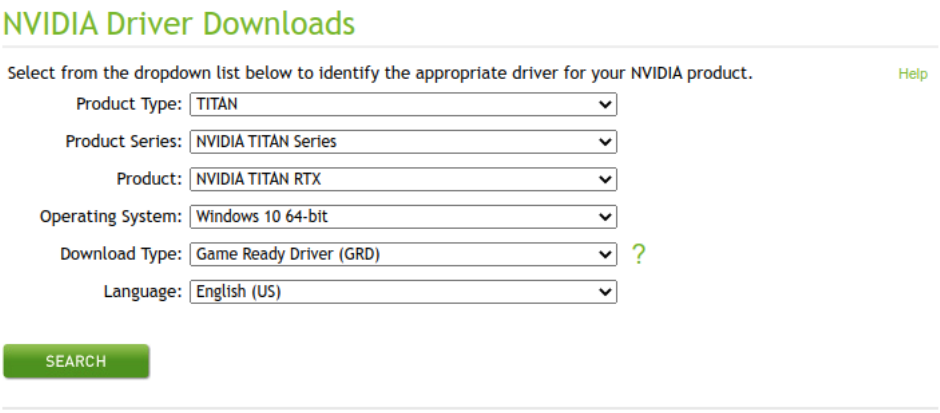
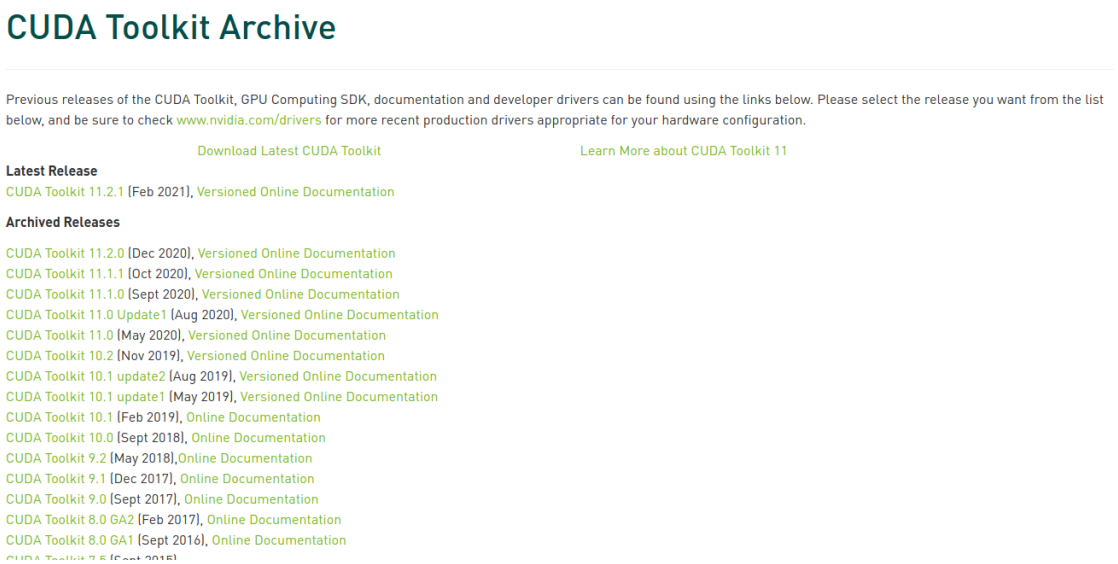
차선 인식 환경 설정

▼ 초기 세팅

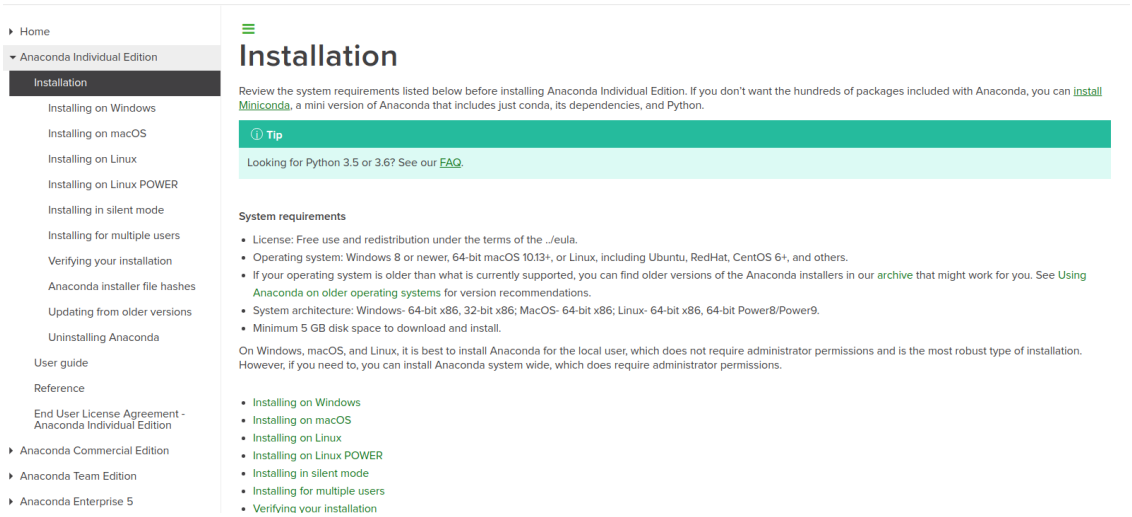
1. gpu driver 설치(https://www.nvidia.com/Download/index.aspx)



2. cuda toolkit 설치(https://developer.nvidia.com/cuda-toolkit-archive)



3. anaconda 설치(https://developer.nvidia.com/cuda-toolkit-archive)



4. 가상환경 만들기

   conda create -n env python=3.7

```
(base) sang@sang-H470-HD3:~$ conda create -n env python=3.7
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

5. 가상환경 활성화

   conda activate env

```
(base) sang@sang-H470-HD3:~$ conda activate env
(env) sang@sang-H470-HD3:~$
```

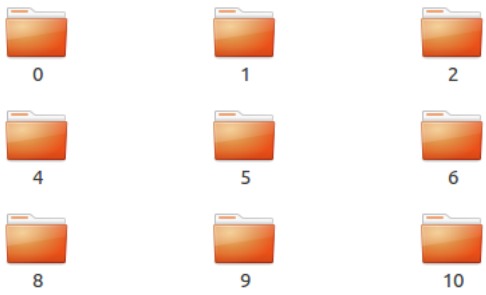
6. 필요한 패키지 설치
   1. conda install pytorch torchvision torchaudio cudatoolkit=11.0 -c pytorch -y
   2. conda install -c conda-forge pytorch-lightning -y
   3. conda install -c anaconda ujson -y
   4. conda install -c conda-forge opencv -y
   5. conda install -c conda-forge matplotlib -y

▼ trafficlight/NIA

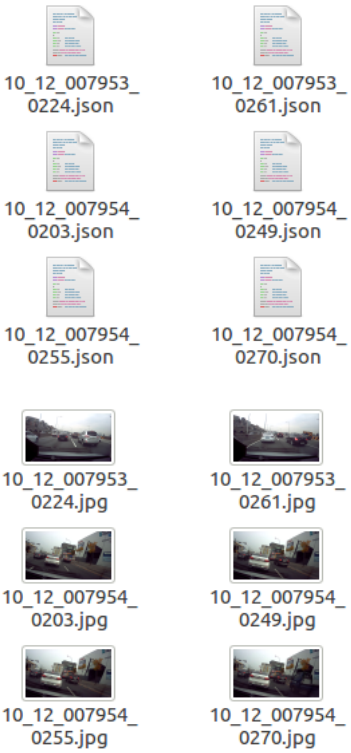
▼ 데이터셋 준비하는 방법

1. 다음과 같이 image와 json 폴더를 만듭니다.

2. Image와 json 각각에 동일한 이름으로 하위 폴더를 필요한만큼 만듭니다.



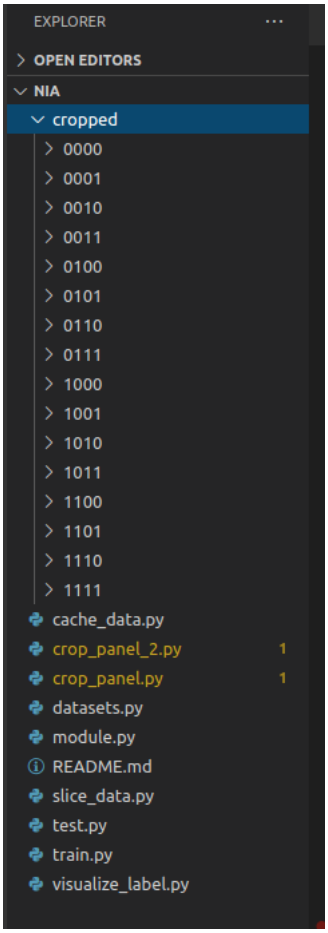
3. 각 하위 폴더에는 다음과 같이 동일한 이름으로 이미지와 json을 하위 폴더에 넣습니다.



4. NIA 폴더에서 imgroot와 resultroot를 다음과 같이 앞서 만들어주었던 image와 json의 폴더 경로를 지정해줍니다.

```
imgroot = "/media/sang/20/20만장 납품_2021.02.16/light_object/image"
resultroot = "/media/sang/20/20만장 납품_2021.02.16/light_object/json"
```

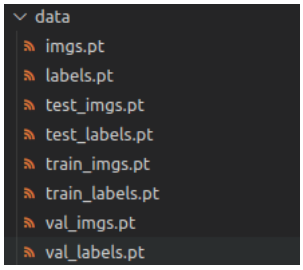
5. NIA폴더에 cropped 라는 이름으로 폴더를 생성합니다.
6. python crop_panel.py을 실행시키면 아래와 같이, cropped된 신호등 데이터가 준비됩니다.



7. cache_data.py 코드에서 img_dirs에 cropped 폴더의 경로를 넣고 img_folders에는 cropped의 하위 폴더의 이름을 넣습니다.

```
img_dirs = ["cropped/"]
img_folders = ["0000","0001","0011","1000","0010"]
```

8. NIA폴더에 data 라는 이름의 폴더를 생성해줍니다.
9. cache_data.py를 실행시키면, data 폴더에 imgs.pt 파일과 labels.pt 파일이 생성될 것입니다.
10. 다음으로 datasets.py를 실행시키면, train, validation, test를 위한 pt파일이 생성될 것이고 학습을 위한 데이터가 모두 준비 됐습니다.



▼ 학습하는 방법

1. train.py 코드에서 pl.Trainer에 사용 가능한 gpu 개수만큼 인수 값을 바꾸어 줍니다.

```
#gpu가 1개인 경우
trainer = pl.Trainer(gpus=0, distributed_backend="ddp", callbacks=[EarlyStopping(monitor='val_loss',patience=10)])
```

```
#gpu가 4개인 경우
trainer = pl.Trainer(gpus=[0,1,2,3], distributed_backend="ddp", callbacks=[EarlyStopping(monitor='val_loss',patience=10)]
```

2. python train.py 실행

▼ 테스트하는 방법

1. test용 데이터는 데이터셋을 준비하며 생성했으므로 따로 준비할 것은 없습니다.
2. train.py로 훈련된 가중치 파일의 경로를 test.py의 4번째 줄의 checkpoint_path = 에 입력합니다.

```
#예시
model = NIA_SEGNet_module.load_from_checkpoint(checkpoint_path="lightning_logs/version_9/checkpoints/epoch=32-step=32.ckp
```

3. test.py 를 실행합니다.

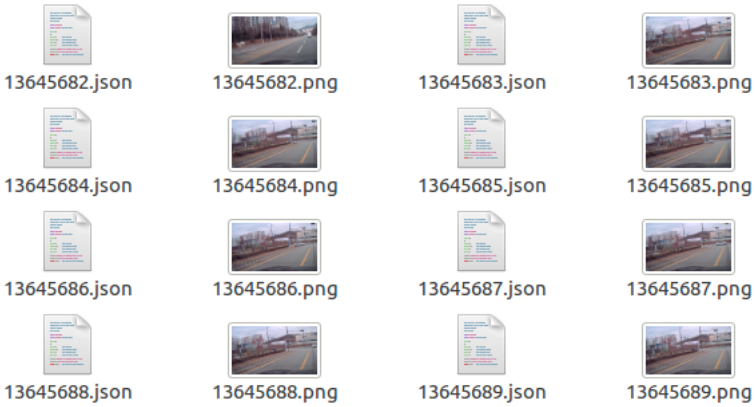
▼ lane_detection/NIA_SEGMENTATION

▼ 데이터셋 준비하는 방법

1. 다음과 같이 dataset 폴더를 생성합니다.

2. dataset폴더 아래에 하위 폴더를 다음과 같이 생성합니다.

3. 각 하위 폴더에는 아래와 같이 이미지 파일과 json 파일을 함께 넣어줍니다.

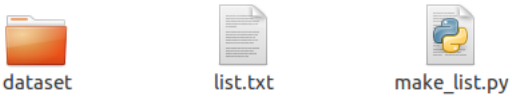


4. dataset의 하위 폴더의 경로를 저장할 수 있는 코드(make_list.py)를 준비합니다.

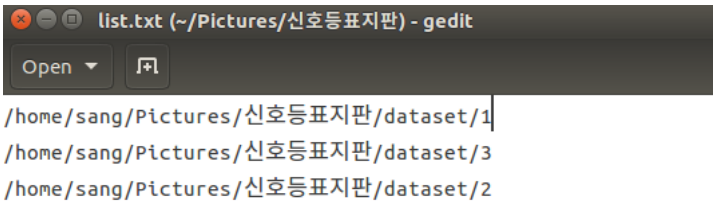
```
import os

#dataset의 경로
path = os.getcwd()+'/dataset'

#dataset의 하위폴더의 경로를 담은 list.txt 경로
txt = open(os.getcwd()+'/list.txt',"w+")

img_list = os.listdir(path)

for img in img_list:
    img_path = path+'/'+img
    if os.path.isdir(img_path):
```

```
        txt.writelines(img_path+"\n")

txt.close()
```

5. 아래와 같이 구성한 후, make_list.py 코드를 dataset을 담고 있는 폴더의 경로에서 실행하면 list.txt가 생성됩니다.



6. list.txt는 다음과 같이 생성됐음을 확인할 수 있습니다.

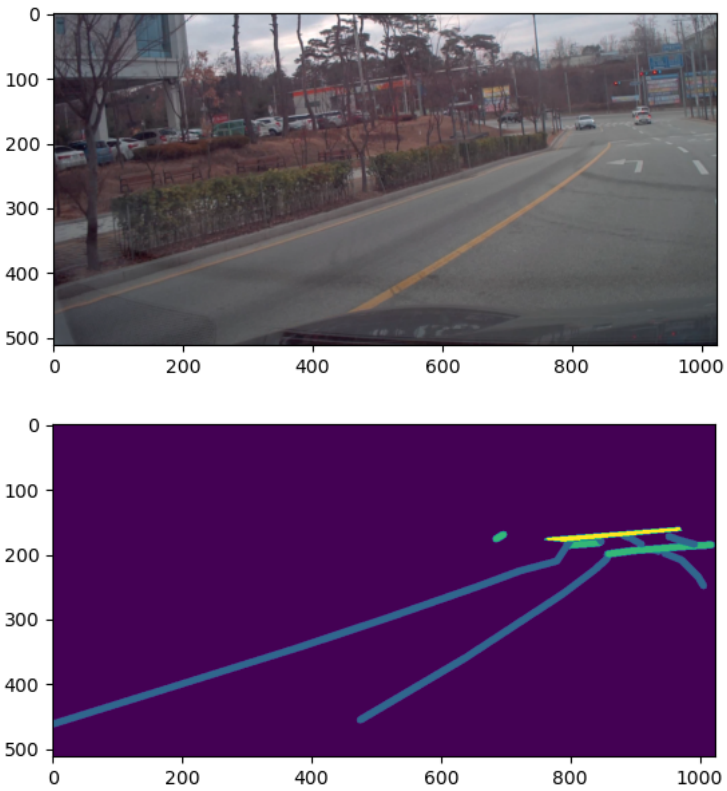


7. list.txt까지 다 만들어 졌다면, 이제 NIA_SEGMENTATION 폴더의 dataset.py에서 list.txt를 읽도록 해주어야 합니다.

```
if __name__ == "__main__":
    ld = LaneDataset(data_path='/home/sang/Pictures/신호등표지판/list.txt')
    # ss = torch.zeros(4)
    # cnt = 0
    for i,(img, target) in enumerate(ld):
        print(i)
        plt.imshow(img.permute((1,2,0)))
        plt.savefig("b.png")
        plt.figure()
        plt.imshow(target)
        plt.show()
        # plt.savefig("a.png")
        # input()
        # plt.close('all')
        # cnt += 0
        # tt = torch.bincount(target.reshape(-1))
        # print(tt)
        # if len(tt) <= len(ss):
        #     ss[:len(tt)] += tt/91357
        # else:
        #     tmp = ss
        #     tt = ss
        #     tt[:len(tmp)] += tmp/91357
        # print(ss)
```

위의 코드에서 LaneDataset에 들어가는 data_path 값에 list.txt의 경로를 넣고, 코드를 약간 변경해 줍니다.

8. dataset.py를 실행하면 다음과 같이 시각화된 데이터를 확인할 수 있습니다.



▼ 학습하는 방법

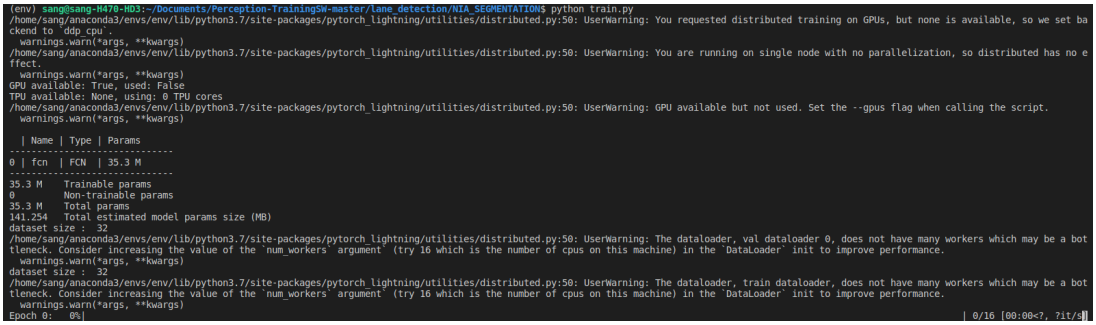
1. 준비된 데이터셋의 경로를 담는 list.txt의 경로를 각각 module.py의 dataloader의 data_path에 넣어줍니다.

```
def val_dataloader(self):
        dataset = LaneDataset(data_path='/home/sang/Pictures/신호등표지판/list_train.txt')
        train_loader = DataLoader(dataset, batch_size = self.batch_size, num_workers=0)
        return train_loader

  def test_dataloader(self):
      dataset = LaneDataset(data_path='/home/sang/Pictures/신호등표지판/list_test.txt')
      train_loader = DataLoader(dataset, batch_size = self.batch_size, num_workers=0,shuffle=True)
      return train_loader

  def train_dataloader(self):
      dataset = LaneDataset(data_path='/home/sang/Pictures/신호등표지판/list_val.txt')
      train_loader = DataLoader(dataset, batch_size = self.batch_size, num_workers=0,shuffle=True)
      return train_loader
```

2. train.py에 pl.Trainer에 사용 가능한 gpu 만큼 값을 바꾸어 줍니다.

```
#gpu가 1개인 경우
trainer = pl.Trainer(gpus=0, distributed_backend="ddp", callbacks=[EarlyStopping(monitor='val_loss',patience=2)])
#gpu가 4개인 경우
trainer = pl.Trainer(gpus=[0,1,2,3], distributed_backend="ddp", callbacks=[EarlyStopping(monitor='val_loss',patience=2)])
```

3. python train.py 를 실행해서 다음과 같이 학습이 진행됨을 확인할 수 있습니다.



▼ 테스트하는 방법

1. train.py로 훈련된 가중치 파일의 경로를 test.py의 4번째 줄의 checkpoint_path = 에 입력.
2. 테스트 할 데이터셋 세팅(train.py의 데이터셋 세팅과 동일)
3. python test.py 실행