RandLA-Net #custom

INTRO

- 해당 페이지는 custom dataset을 RandLA-Net에 적용하기 위해 거친 과정을 정리하기 위해 작성되었음
- custom dataset은 labeling된 상태여야 하며, point labeler로 labeling되었음을 가정
- custom dataset은 아래와 같이 semantickitti와 동일한 파일 구성을 가지고 있음을 가정
  - labels → .label
  - velodyne → .bin
  - calib.txt, instances.txt, poses.txt

- 해당 custom dataset은 unlabeled와 kickboard - 2개의 class만 존재하는 point cloud dataset임

- 수정해야 하는 파일 목록
  - main_SemanticKITTI.py
  - helper_tool.py
  - utils/semantic_kitti.yaml
  - utils/data_prepare_semantickitti.py
  - jobs_test_semantickitti.sh

- 본 guide에서의 데이터셋명은 SST임
  - main_SST.py
  - utils/semantic_sst.yaml
  - utils/data_prepare_sst.py
  - jobs_test_sst.sh

주의사항

- **custom dataset의 폴더명 구성은 반드시 여기에 기재된 대로 따를 것**
  - 다르게 할 경우 파일 수정이 ***상당히*** 어려워진다
- SST에서의 구성
  - training
    - 01 ~ 06
  - validation
    - 08

    - **test**
        - **11**
- **test는 반드시 11부터 이루어져야 함**
    - test 폴더 number가 training 사이에 껴 있을 경우 int(seq_id) 수정하기 매우 힘듦
- 가급적 training 00 ~ 10, validation 08, test 11 ~ 로 할 것
    - 최대한 semantickitti와 유사하게 할 것

- **반드시!! 파일 경로 수정할 것**

순서

1. point labeler로 labeling된 파일의 label number 확인
    - point labeler로 labeling을 완료할 경우 아래와 같은 파일 구성을 가지게 된다.
        - labels → .label
        - velodyne → .bin
        - calib.txt, instances.txt, poses.txt
    - 여기서 labels, velodyne을 참고하여 현재 labeling된 파일의 label number를 확인해야 함
    - 따라서 다음의 code를 RandLA-Net 디렉토리 내에 추가하여 확인

```
python verify.py
```

        - verify.py

```
import os
import numpy as np

# 라벨 파일이 저장된 디렉토리 경로
label_dir_path = "/home/j2/sequence/01/labels"

# 디렉토리 내 모든 라벨 파일 읽어들이기
label_files = [os.path.join(label_dir_path, f) for f in os.listdir(label_dir_path) if f.endswith(".label")]

# 모든 라벨 파일에 대해 클래스 번호 확인
unique_labels = set()
for label_file in label_files:
    labels = np.fromfile(label_file, dtype=np.uint32)
    unique_labels = unique_labels.union(set(labels))

print("클래스 번호:", sorted(list(unique_labels)))
```

        - 출력 결과


```
클래스 번호: [0, 80]
```

- label number는 0(unlabeled)과 80(kickboard)임을 알 수 있음

2. 총 point 개수, labeling된 point 개수, class 0과 80의 point 개수 출력
   - 현재 .label, .bin 파일에서 상기 목록을 알기 위해 다음의 code를 RandLA-Net 디렉토리 내에 추가하여 확인

```
python count.py
```

     - count.py

```
import os
import numpy as np

# 포인트 클라우드 파일 경로 설정
velodyne_path = "/home/j2/sequence/01/velodyne/"
# 라벨 파일 경로 설정
label_path = "/home/j2/sequence/01/labels/"

# 포인트 클라우드 총 개수, 라벨링된 포인트 개수, 클래스 0과 80의 개수를 초기화
total_points = 0
total_labeled_points = 0
class_0_points = 0
class_80_points = 0

# velodyne_path 경로에 있는 모든 파일을 하나씩 읽어들이며 반복
for file_name in os.listdir(velodyne_path):
    # 파일의 확장자가 .bin일 경우에만 실행
    if file_name.endswith('.bin'):

  # 포인트 클라우드와 해당 클라우드의 라벨 파일 경로 설정
        bin_path = os.path.join(velodyne_path, file_name)
        label_path_file = os.path.join(label_path, file_name[:-4] + ".label")

        # numpy 배열로 포인트 클라우드와 라벨 파일 읽어들임
        points = np.fromfile(bin_path, dtype=np.float32)
        labels = np.fromfile(label_path_file, dtype=np.uint32)

        # 포인트 클라우드와 라벨링된 포인트 개수 계산하여 총 개수에 누적
        num_points = len(points) // 4
        total_points += num_points
        labeled_points = np.count_nonzero(labels)
        total_labeled_points += labeled_points

        # 클래스 0과 80의 포인트 개수를 계산하여 누적
        class_0_points += np.count_nonzero(labels == 0)
        class_80_points += np.count_nonzero(labels == 80)

# 총 포인트 개수 출력
print("Total number of points:", total_points)

# 라벨링된 포인트 개수 출력
print("Total number of labeled points:", total_labeled_points)
```

```
# 클래스 0과 80의 포인트 개수 출력
print("Number of points belonging to class 0:", class_0_points)
print("Number of points belonging to class 80:", class_80_points)

# 80 클래스의 비율 계산
class_80_percentage = class_80_points / total_points
print("Percentage of points belonging to class 80:", class_80_percentage)
```

- 출력 결과
  - 참고만 할 것

```
(randlanet) j2@j2:~/repository/RandLA-Net$ python count.py
Total number of points: 122191872
Total number of labeled points: 209866
Number of points belonging to class 0: 121982006
Number of points belonging to class 80: 209847
Percentage of points belonging to class 80: 0.001717356453954646
(randlanet) j2@j2:~/repository/RandLA-Net$ python count.py
Total number of points: 67829760
Total number of labeled points: 0
Number of points belonging to class 0: 67829760
Number of points belonging to class 80: 0
Percentage of points belonging to class 80: 0.0
(randlanet) j2@j2:~/repository/RandLA-Net$ python count.py
Total number of points: 79265792
Total number of labeled points: 50103
Number of points belonging to class 0: 79215689
Number of points belonging to class 80: 50103
Percentage of points belonging to class 80: 0.0006320885559309115
(randlanet) j2@j2:~/repository/RandLA-Net$ python count.py
Total number of points: 119046144
Total number of labeled points: 165140
Number of points belonging to class 0: 118881004
Number of points belonging to class 80: 165140
Percentage of points belonging to class 80: 0.001387193187878475
(randlanet) j2@j2:~/repository/RandLA-Net$ python count.py
Total number of points: 48889856
Total number of labeled points: 286652
Number of points belonging to class 0: 48603204
Number of points belonging to class 80: 286652
Percentage of points belonging to class 80: 0.005863220378476876
(randlanet) j2@j2:~/repository/RandLA-Net$ python count.py
Total number of points: 115081216
Total number of labeled points: 219596
Number of points belonging to class 0: 114861620
Number of points belonging to class 80: 219596
Percentage of points belonging to class 80: 0.0019081828262919988
(randlanet) j2@j2:~/repository/RandLA-Net$ python count.py
Total number of points: 307134464
Total number of labeled points: 388336
Number of points belonging to class 0: 306746128
Number of points belonging to class 80: 387752
Percentage of points belonging to class 80: 0.001262482871345887
```

3. 0 class number, 80 class number 비율 계산

4. data_prepare_sst.py 수정
   - 12번 줄 semantic-kitti.yaml → sst.yaml 수정

- 20번 21번 경로 수정
- 51번 숫자를 validation 폴더명으로 수정

5. sst.yaml **1차** 수정
   - yaml 파일 수정 매우 중요하므로 유의
   - class 2개인 SST에 맞게 수정한 것임
     - labels
       - 1번에서 .label, .bin 파일에서 얻은 label number가 0, 80임을 확인하였으므로 아래와 같이 기재

```
0 : "unlabeled"
80: "kickboard"
```

     - color_map
       - point_labeler의 assets/labels.xml 파일에서 해당 label의 bgr 색상 number를 확인할 것
       - 80은 [255, 240, 150] 색상임

```
0 : [0, 0, 0]
80: [255, 240, 150]
```

     - content
       - 2번, 3번에서 얻은 각 class의 ratio를 기재
     - learning_map
       - 각 class 뒤에 붙는 숫자는 각기 달라야 함을 유의

```
0 : 0     # "unlabeled"
80: 1     # "kickboard" mapped to "unlabeled" --------------------------mapped
```

     - learning_map_inv
       - learning_map의 inverse를 기재함

```
0 : 0      # "unlabeled", and others ignored
1 : 80     # "kickboard"
```

     - **learning_ignore**
       - True는 학습 시 무시하겠다는 뜻임
       - 따라서 무시하고자 하는 class 뒤에 true를 작성해 주면 됨
       - 본 SST 데이터셋에서는 class가 2개 - unlabeled, kickboard 이므로 두 class 모두 무시하지 않기로 함
         - 만약 여기서 0: True 로 작성하였다면 0 class를 무시하는 것이 되어 class가 1개가 됨
           - 해당 경우 나머지 파일들에서 class number를 수정해주어야 함

```
0 : False      # "unlabeled", and others ignored
80: False      # "kickboard"
```

     - split

```
train:
    - 1
    - 3
    - 4
    - 5
    - 6
  valid:
    - 8
  test:
    - 11
```

6. 수정한 sst.yaml대로 sequence(원본) 파일 변환

```
python utils/data_prepare_sst.py
```

- 변환된 파일의 폴더명은 sequence_0.06이 됨
  - 만약 grid size를 0.06에서 변경하게 된다면 폴더명도 변경할 수 있도록 data_prepare.py 수정

7. 변환된 폴더의 validation, test 폴더 확인
   - training
     - KDTree, labels, velodyne으로 구성
   - validation
     - KDTree, labels, proj, velodyne으로 구성
   - test
     - KDTree, proj, velodyne으로 구성
   - 만약 val, test가 위와 같이 구성되지 않고 training과 같이 구성되어 있다면 잘못된 것임
     - 해당 경우 data_prepare_sst.py의 35번줄, 51번줄을 재확인하기 바람

8. 변환된 파일의 class number 확인
   - 현재 .npy 파일에서 상기 목록을 알기 위해 다음의 code를 RandLA-Net 디렉토리 내에 추가하여 확인

```
python npyverify.py
```

     - npyverify.py

```
import numpy as np
import os

folder_path = "/home/j2/sequence_0.06/01/labels/"
file_list = os.listdir(folder_path)

npy_files = [f for f in file_list if f.endswith('.npy')]
for file_name in npy_files:
    file_path = os.path.join(folder_path, file_name)
    data = np.load(file_path)

label_numbers = np.unique(data[:, -1])
print(f"파일 {file_name}에 포함된 라벨 번호: {label_numbers}")
```

- 출력 결과

```
포함된 라벨 번호: [0 1]
```

- **.label, .bin 파일에서 확인한 label 번호가 80에서 data_prepare_sst.py로 변환 이후 1로 변경된 것을 알 수 있음**
  - 따라서 yaml파일을 수정해야 함

9. 변환된 파일의 point 개수 count
   - 현재 .npy 파일에서 상기 목록을 알기 위해 다음의 code를 RandLA-Net 디렉토리 내에 추가하여 확인

```
python npycount.py
```

   - npycount.py

```
import os
import numpy as np

# 포인트 클라우드 파일 경로 설정
velodyne_path = "/home/j2/sequence_0.06/01/velodyne/"
# 라벨 파일 경로 설정
label_path = "/home/j2/sequence_0.06/01/labels/"

# 포인트 클라우드 총 개수, 라벨링된 포인트 개수, 클래스 0과 80의 개수를 초기화
total_points = 0
total_labeled_points = 0
class_0_points = 0
class_80_points = 0

# velodyne_path 경로에 있는 모든 파일을 하나씩 읽어들이며 반복
for file_name in os.listdir(velodyne_path):
    # 파일의 확장자가 .npy일 경우에만 실행
    if file_name.endswith('.npy'):
        # 포인트 클라우드와 해당 클라우드의 라벨 파일 경로 설정
        bin_path = os.path.join(velodyne_path, file_name)
        label_path_file = os.path.join(label_path, file_name[:-4] + ".npy")

        # numpy 배열로 포인트 클라우드와 라벨 파일 읽어들임
        points = np.load(bin_path)
        labels = np.load(label_path_file)

        # 포인트 클라우드와 라벨링된 포인트 개수 계산하여 총 개수에 누적
        num_points = len(points)
        total_points += num_points
        labeled_points = np.count_nonzero(labels)
        total_labeled_points += labeled_points

        # 클래스 0과 80의 포인트 개수를 계산하여 누적
        class_0_points += np.count_nonzero(labels == 0)
        class_80_points += np.count_nonzero(labels == 1)

# 총 포인트 개수 출력
print("Total number of points:", total_points)

# 라벨링된 포인트 개수 출력
print("Total number of labeled points:", total_labeled_points)

# 클래스 0과 80의 포인트 개수 출력
print("Number of points belonging to class 0:", class_0_points)
print("Number of points belonging to class 80:", class_80_points)

# 80 클래스의 비율 계산
class_80_percentage = class_80_points / total_points
print("Percentage of points belonging to class 80:", class_80_percentage)
```

- 여기서 계산해야 하는 것
  - **class별 point 개수 (폴더 합산)**
    - **계산해서 helper_tool.py의 get_class_weight에 기재**
  - class 비율
    - 계산해서 sst.yaml의 content에 기재

10. sst.yaml **2차** 수정
    - 1차 수정에서 80이라서 했던 것을 모두 1로 변경

```
# This file is covered by the LICENSE file in the root of this project.
labels:
  0 : "unlabeled"
  1 : "kickboard"
color_map: # bgr
  0: [0, 0, 0]
  1: [255, 240, 150]
content: # as a ratio with the total number of points
  0 : 0.997518044
  1 : 0.00248195637
# classes that are indistinguishable from single scan or inconsistent in
# ground truth are mapped to their closest equivalent
learning_map:
  0 : 0     # "unlabeled"
  1 : 1     # "kickboard" mapped to "unlabeled" --------------------------mapped
learning_map_inv: # inverse of previous map
  0 : 0      # "unlabeled", and others ignored
  1 : 1     # "kickboard"
learning_ignore: # Ignore classes
  0 : False      # "unlabeled", and others ignored
  1 : False     # "kickboard"
split: # sequence numbers
  train:
    - 1
    - 3
    - 4
    - 5
    - 6
  valid:
    - 8
  test:
    - 11
```

11. helper_tool.py 수정
    - class 추가
      - 유의
        - num_points는 4096 * n 형태로 n은 **직접 하나씩 넣어서** run이 가능한 number를 찾아야 함
        - num_classes는 유의미한 class의 개수 기입
          - 만약 unlabeled를 무시한다면 class의 개수에서 1개를 빼야 함
        - 기타 입맛에 맞게 하이퍼파라미터 batch size, epoch 수정
        - 나머지는 semantickitti 데이터셋과 유사한 구조라면 가급적 건들지 말 것

```
class ConfigSST:
    k_n = 16  # KNN
    num_layers = 4  # Number of layers
    num_points = 4096 * 2  # Number of input points
    num_classes = 2  # Number of valid classes
```

```
    sub_grid_size = 0.06  # preprocess_parameter

    batch_size = 6  # batch_size during training
    val_batch_size = 20  # batch_size during validation and test
    train_steps = 500  # Number of steps per epochs
    val_steps = 100  # Number of validation steps per epoch

    sub_sampling_ratio = [4, 4, 4, 4]  # sampling ratio of random sampling at each layer
    d_out = [16, 64, 128, 256]  # feature dimension
    num_sub_points = [num_points // 4, num_points // 16, num_points // 64, num_points // 256]

    noise_init = 3.5  # noise initial parameter
    max_epoch = 100  # maximum epoch during training
    learning_rate = 1e-2  # initial learning rate
    lr_decays = {i: 0.95 for i in range(0, 500)}  # decay rate of learning rate

    train_sum_dir = 'train_log'
    saving = True
    saving_path = None
```

- get_file_list 수정
  - if seq_id == '08'은 validation 폴더명을 작성
  - elif int(seq_id) ≥ 11 은 test 폴더명을 작성하는 곳이나 가급적 test 폴더를 11번 이상으로 설정하고 이 부분은 건들지 말 것
  - elif seq_id in ['01', '03', '04', '05', '06'] 부분은 training 폴더명을 작성
- get_class_weights 수정
  - dataset_name을 추가해서 np.array에 class별 point 개수를 작성
    - **만약 unlabeld를 무시하거나 기타 무시하는 class가 있다면 그 class의 point 개수는 작성하지 않음**

```
elif dataset_name is 'SST':
    num_per_class = np.array([490220781, 1002977])
```

12. main_SST.py 수정
    - 12번줄 class명 변경
    - 14번 dataset name 변경
    - 15번 dataset_path 데이터셋 경로 변경
    - 16번 줄에 있는 class 모두 작성

```
self.label_to_names = {0: 'unlabeled',
                       1: 'kickboard'}
```

    - **21번 np.array([0]) 수정**
      - 만약 0(unlabeled) class를 무시하면 그대로 가져감
      - 0(unlabeled) class를 무시하지 않는다면 np.array([]) 이렇게 0을 뺌
    - 23번 validation 폴더명 작성
    - 96번 int(seq_id) ≥ 11 확인 (여태 강조한 데이터셋 구조를 가져간다면 수정할 필요 없음)
    - 177번 test_area의 default를 test의 첫 폴더명으로 변경

13. RUN

```
python main_SST.py --mode train --gpu 0 --test_area 11
```

14. jobs_test_sst. 수정
    - test_area 뒤에 test 폴더명 모두 작성 (한 줄씩)

```
python -B main_SST.py --gpu 0 --mode test --test_area 11
```

15. evaluation

```
sh jobs_test_semantickitti.sh
```

error list

- helper_tool.py의 num_points 배수 수정
- ZeroDivisionError

- 어쨌든 수정해야 하는 거 다 수정하면 돌아가긴 한다