HW2 Report

311554021 張祐閤

a. Code

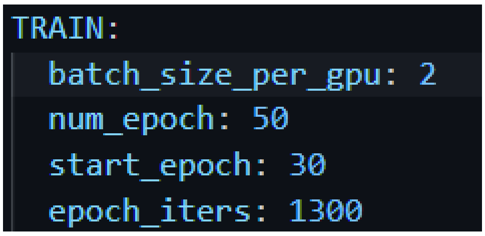
1. Training segmentation model:

Dataset:

The training dataset is split into two parts, images from apartment_0 and images from others. Each set contains 1300 images, the number 1300 comes from 13 rooms in apartment_0 and each room collects 100 images. For a fair experiment setting, I also collect a total of 1300 images from other scenes.

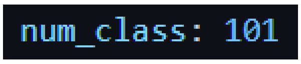
Training Settings:

I use hrnetv2 semantic segmentation model which is trained on ADE20K dataset for 30 epoch, and fine tune it for 20 epoch.



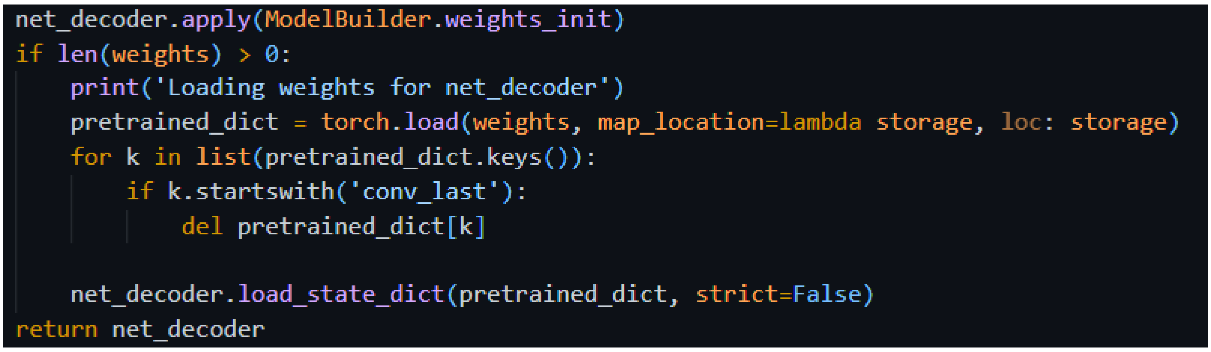
```
TRAIN:
  batch_size_per_gpu: 2
  num_epoch: 50
  start_epoch: 30
  epoch_iters: 1300
```

The other modification from the original config file is that the ADE20K dataset has 150 different classes while the Replica dataset only has 101 classes. So I have to change num_class in the config file.



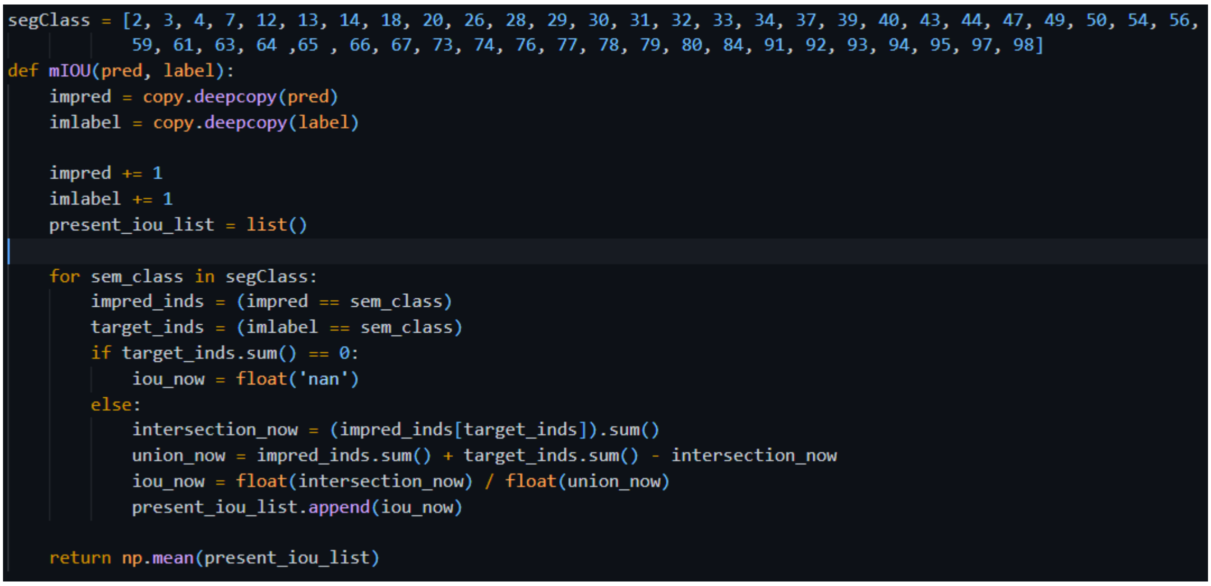
```
num_class: 101
```

As a result of changing the number of predicted classes, the last layer of the decoder will have a different shape from the original model. Thus, when loading the pre-trained weight of the decoder, I have to remove the final layers of the pre-trained model as shown in below:



```
net_decoder.apply(ModelBuilder.weights_init)
if len(weights) > 0:
    print('Loading weights for net_decoder')
    pretrained_dict = torch.load(weights, map_location=lambda storage, loc: storage)
    for k in list(pretrained_dict.keys()):
        if k.startswith('conv_last'):
            del pretrained_dict[k]

    net_decoder.load_state_dict(pretrained_dict, strict=False)
return net_decoder
```

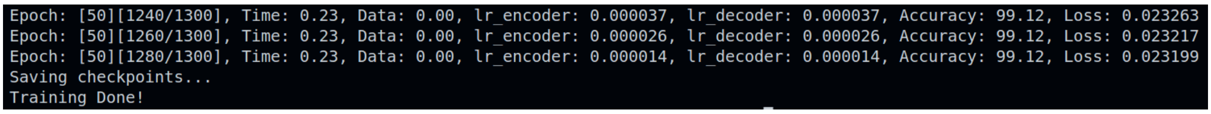
Also when calculate mean Iou at validation, only the 49 classes appear in apartment_0 need to be considered, so I write a new function to compute new mIoU:



```
segClass = [2, 3, 4, 7, 12, 13, 14, 18, 20, 26, 28, 29, 30, 31, 32, 33, 34, 37, 39, 40, 43, 44, 47, 49, 50, 54, 56,
            59, 61, 63, 64 ,65 , 66, 67, 73, 74, 76, 77, 78, 79, 80, 84, 91, 92, 93, 94, 95, 97, 98]
def mIOU(pred, label):
    impred = copy.deepcopy(pred)
    imlabel = copy.deepcopy(label)

    impred += 1
    imlabel += 1
    present_iou_list = list()

    for sem_class in segClass:
        impred_inds = (impred == sem_class)
        target_inds = (imlabel == sem_class)
        if target_inds.sum() == 0:
            iou_now = float('nan')
        else:
            intersection_now = (impred_inds[target_inds]).sum()
            union_now = impred_inds.sum() + target_inds.sum() - intersection_now
            iou_now = float(intersection_now) / float(union_now)
            present_iou_list.append(iou_now)

    return np.mean(present_iou_list)
```

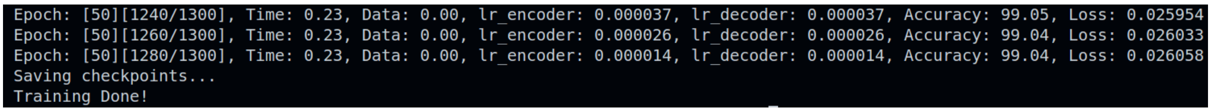
Training Result:

apartment_0 training log



```
Epoch: [50][1240/1300], Time: 0.23, Data: 0.00, lr_encoder: 0.000037, lr_decoder: 0.000037, Accuracy: 99.12, Loss: 0.023263
Epoch: [50][1260/1300], Time: 0.23, Data: 0.00, lr_encoder: 0.000026, lr_decoder: 0.000026, Accuracy: 99.12, Loss: 0.023217
Epoch: [50][1280/1300], Time: 0.23, Data: 0.00, lr_encoder: 0.000014, lr_decoder: 0.000014, Accuracy: 99.12, Loss: 0.023199
Saving checkpoints...
Training Done!
```

others training log



```
Epoch: [50][1240/1300], Time: 0.23, Data: 0.00, lr_encoder: 0.000037, lr_decoder: 0.000037, Accuracy: 99.05, Loss: 0.025954
Epoch: [50][1260/1300], Time: 0.23, Data: 0.00, lr_encoder: 0.000026, lr_decoder: 0.000026, Accuracy: 99.04, Loss: 0.026033
Epoch: [50][1280/1300], Time: 0.23, Data: 0.00, lr_encoder: 0.000014, lr_decoder: 0.000014, Accuracy: 99.04, Loss: 0.026058
Saving checkpoints...
Training Done!
```

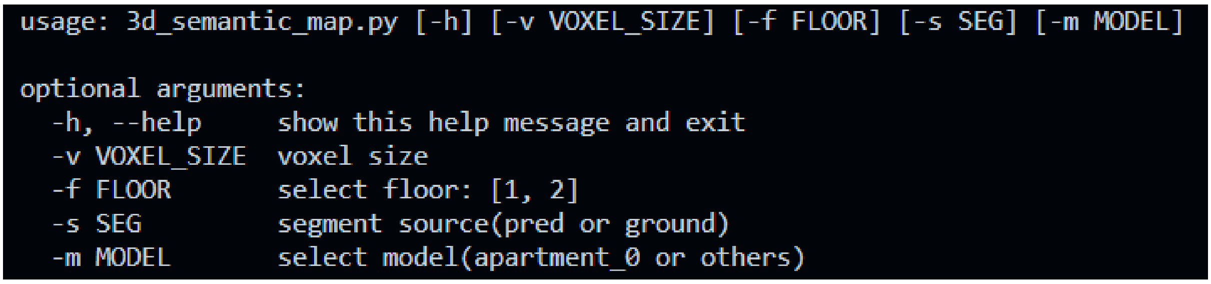
Validation Result:

	Mean IoU	Accuracy
apartment_0 floor1	**0.5254**	**89.18%**
others floor1	0.2354	65.37%
apartment_0 floo2	**0.6682**	**92.09%**
other floor2	0.3462	77.24%

2. 3D semantic map reconstruction:

How to Run:

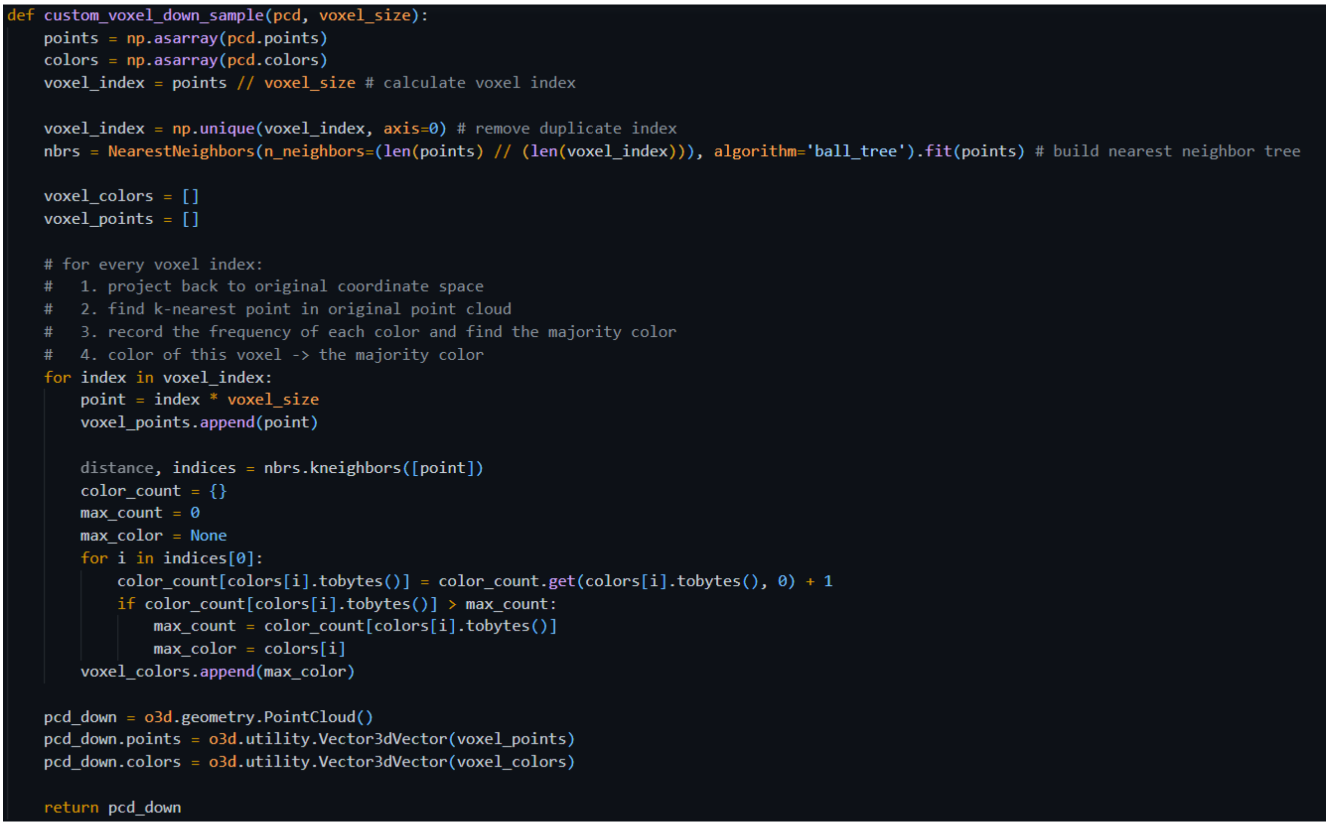
python 3d_sematic_map.py [args]



```
usage: 3d_semantic_map.py [-h] [-v VOXEL_SIZE] [-f FLOOR] [-s SEG] [-m MODEL]

optional arguments:
  -h, --help     show this help message and exit
  -v VOXEL_SIZE  voxel size
  -f FLOOR       select floor: [1, 2]
  -s SEG         segment source(pred or ground)
  -m MODEL       select model(apartment_0 or others)
```

Custom Voxel Down:

To do voxel down, I first divide the coordinate of each point in the original point cloud by the voxel size to get their voxel index, then use np.unique to find out every unique voxel index. Second, for every voxel index, I project it back to the original coordinate space, and find the majority color of its k-nearest neighbor, k here is set to the average original points a voxel represents.

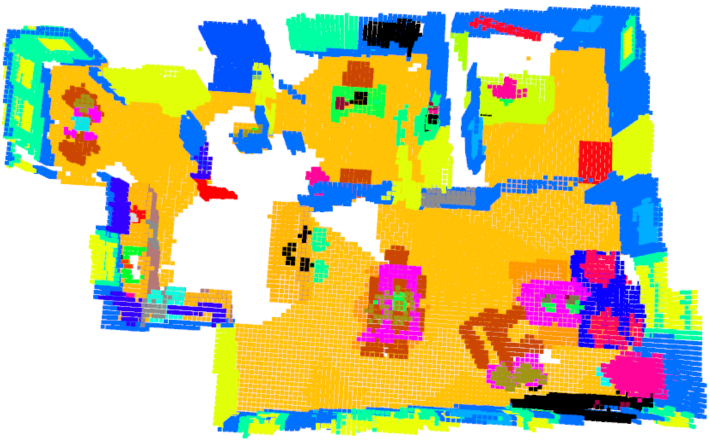


```python
def custom_voxel_down_sample(pcd, voxel_size):
    points = np.asarray(pcd.points)
    colors = np.asarray(pcd.colors)
    voxel_index = points // voxel_size # calculate voxel index

    voxel_index = np.unique(voxel_index, axis=0) # remove duplicate index
    nbrs = NearestNeighbors(n_neighbors=(len(points) // (len(voxel_index))), algorithm='ball_tree').fit(points) # build nearest neighbor tree

    voxel_colors = []
    voxel_points = []

    # for every voxel index:
    #   1. project back to original coordinate space
    #   2. find k-nearest point in original point cloud
    #   3. record the frequency of each color and find the majority color
    #   4. color of this voxel -> the majority color
    for index in voxel_index:
        point = index * voxel_size
        voxel_points.append(point)

        distance, indices = nbrs.kneighbors([point])
        color_count = {}
        max_count = 0
        max_color = None
        for i in indices[0]:
            color_count[colors[i].tobytes()] = color_count.get(colors[i].tobytes(), 0) + 1
            if color_count[colors[i].tobytes()] > max_count:
                max_count = color_count[colors[i].tobytes()]
                max_color = colors[i]
        voxel_colors.append(max_color)

    pcd_down = o3d.geometry.PointCloud()
    pcd_down.points = o3d.utility.Vector3dVector(voxel_points)
    pcd_down.colors = o3d.utility.Vector3dVector(voxel_colors)

    return pcd_down
```

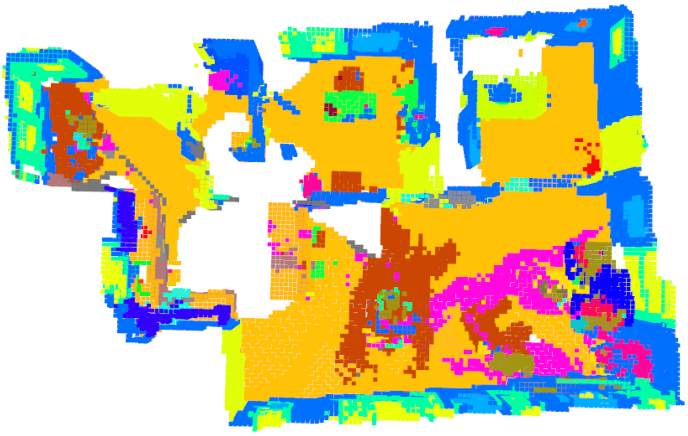
b. Result and Discussion

Floor1 Reconstruction Result

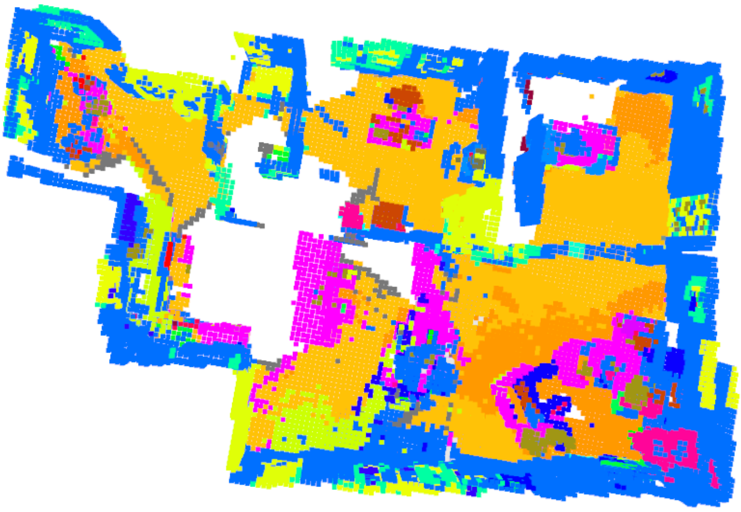
Ground Truth



apartment_0

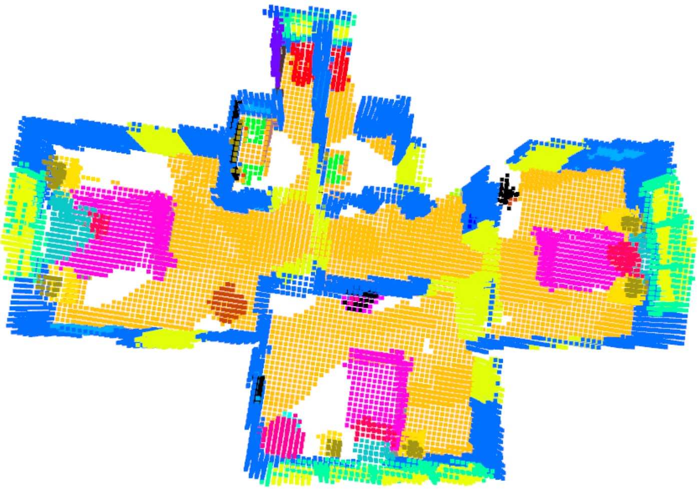


others

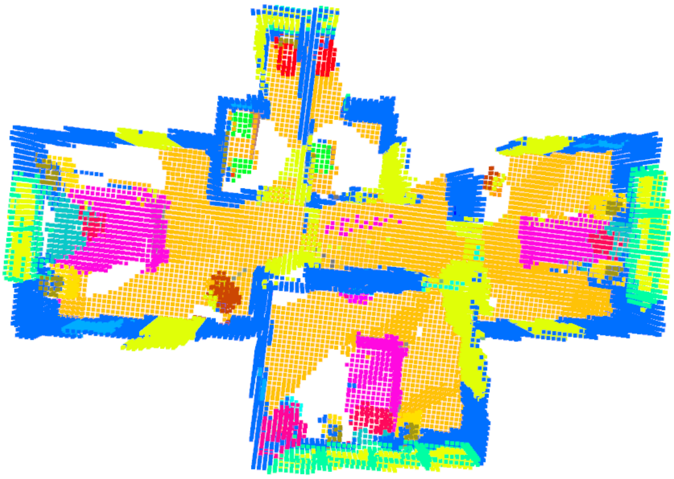


Floor2 Reconstruction Result

Ground Truth



apartment_0



others