HW2 Report

311554043 周俊毅

Code

Training segmentation model

1. Model: HRNetV2
2. Training dataset of **all_scene**: 1300 picture
3. Training dataset of **apartment0**: 1300 picture
4. **Training setting**, the setting of two models are totally same except the num_class and the some directory.

```
DATASET:
  root_dataset: "../"
  list_train: "../dataset_apartment0/training.odgt"
  list_val: "../dataset_apartment0/validation.odgt"
  num_class: 101
  imgSizes: (512, )
  imgMaxSize: 512
  padding_constant: 32
  segm_downsampling_rate: 4
  random_flip: True

MODEL:
  arch_encoder: "hrnetv2"
  arch_decoder: "c1"
  fc_dim: 720

TRAIN:
  batch_size_per_gpu: 16
  num_epoch: 50
  start_epoch: 30
  epoch_iters: 82 # training_size / batch_size
  optim: "SGD"
  lr_encoder: 0.02
  lr_decoder: 0.02
  lr_pow: 0.9
  beta1: 0.9
  weight_decay: 1e-4
  deep_sup_scale: 0.4
  fix_bn: False
  workers: 16
  disp_iter: 20
  seed: 304

VAL:
```

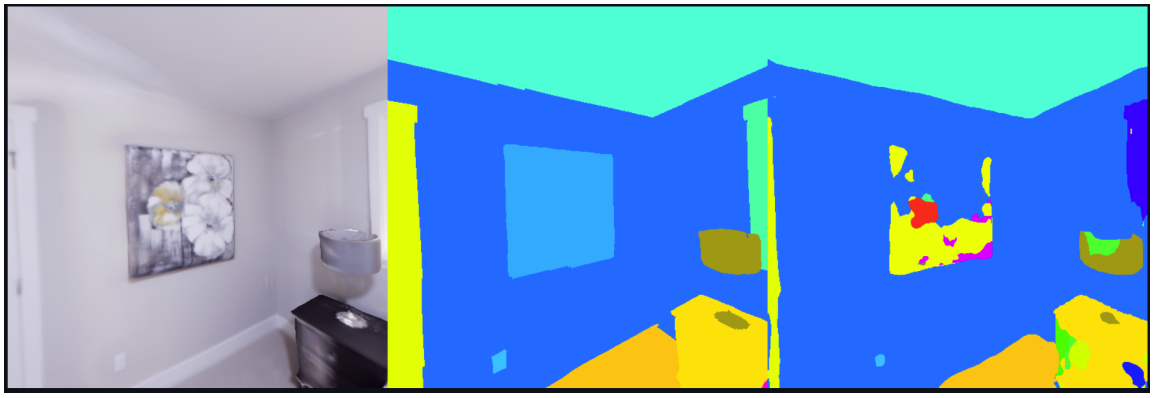
```
  visualize: True
  checkpoint: "epoch_50.pth"

TEST:
  checkpoint: "epoch_50.pth"
  result: "../HW1_Datacollect/first_floor/pred_semantic_apartment0"

DIR: "ckpt/model_apartment0"
```

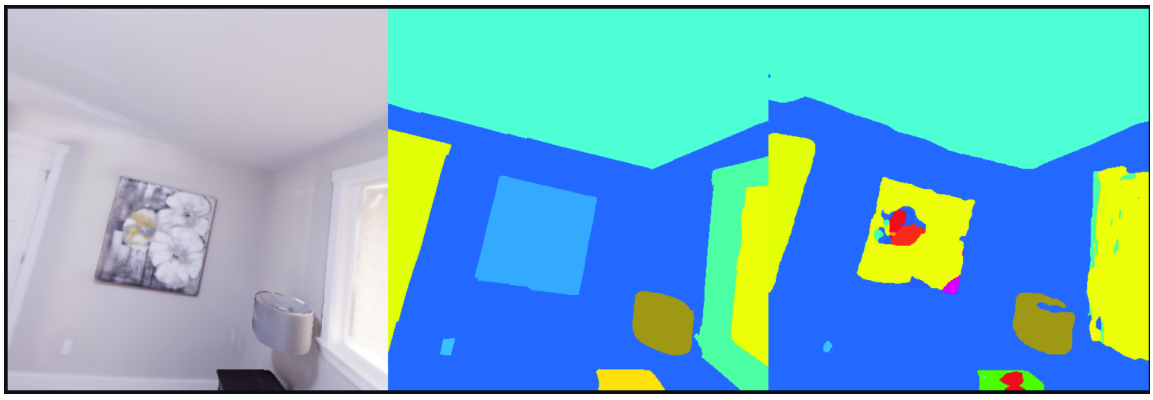
5. Training result:

   a. Model_all_scene

```
Epoch: [50][0/82], Time: 0.55, Data: 0.00, lr_encoder: 0.000592, lr_decoder: 0.000592, Accuracy: 97.97, Loss: 0.060211
Epoch: [50][20/82], Time: 0.70, Data: 0.00, lr_encoder: 0.000460, lr_decoder: 0.000460, Accuracy: 98.16, Loss: 0.054970
Epoch: [50][40/82], Time: 0.71, Data: 0.00, lr_encoder: 0.000324, lr_decoder: 0.000324, Accuracy: 98.15, Loss: 0.055296
Epoch: [50][60/82], Time: 0.80, Data: 0.00, lr_encoder: 0.000181, lr_decoder: 0.000181, Accuracy: 98.15, Loss: 0.055294
Epoch: [50][80/82], Time: 0.80, Data: 0.00, lr_encoder: 0.000021, lr_decoder: 0.000021, Accuracy: 98.15, Loss: 0.055423
```

```
[Eval Summary]:
Mean IoU: 0.3279, Accuracy: 71.60%
```

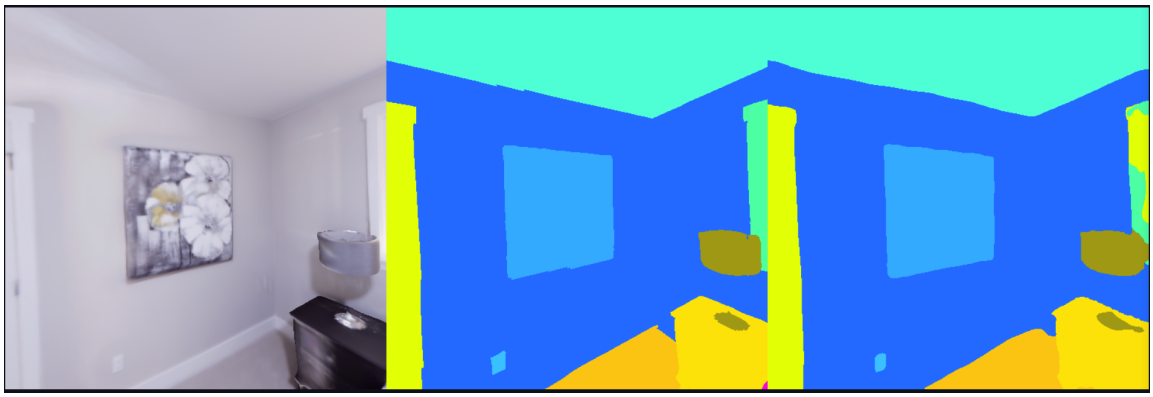


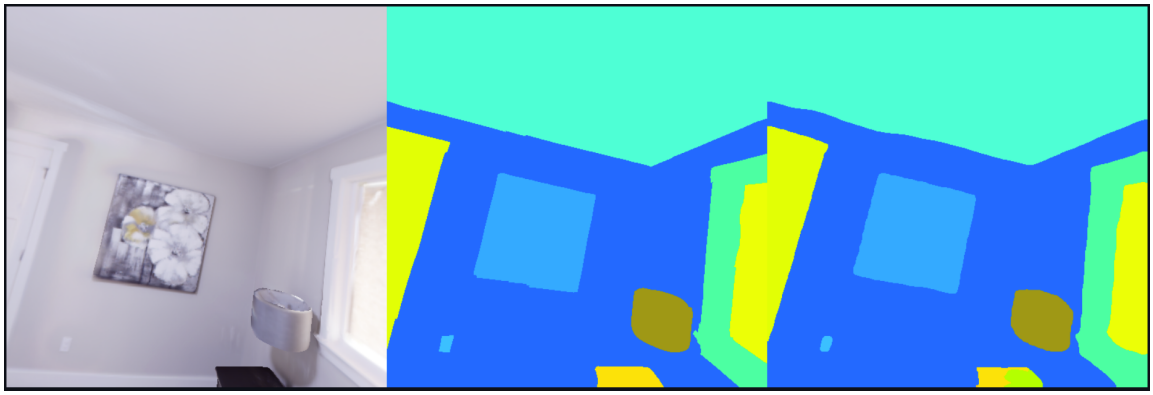


   b. Model_apartment0

```
Epoch: [50][0/82], Time: 0.57, Data: 0.00, lr_encoder: 0.000592, lr_decoder: 0.000592, Accuracy: 98.60, Loss: 0.039771
Epoch: [50][20/82], Time: 0.81, Data: 0.00, lr_encoder: 0.000460, lr_decoder: 0.000460, Accuracy: 98.41, Loss: 0.045903
Epoch: [50][40/82], Time: 0.82, Data: 0.00, lr_encoder: 0.000324, lr_decoder: 0.000324, Accuracy: 98.43, Loss: 0.045200
Epoch: [50][60/82], Time: 0.78, Data: 0.00, lr_encoder: 0.000181, lr_decoder: 0.000181, Accuracy: 98.41, Loss: 0.045636
Epoch: [50][80/82], Time: 0.75, Data: 0.00, lr_encoder: 0.000021, lr_decoder: 0.000021, Accuracy: 98.43, Loss: 0.045390
```

```
[Eval Summary]:
Mean IoU: 0.7305, Accuracy: 97.22%
```





3D semantic map reconstruction

1. How to run my program

```
# Data collection dataset_other
python data_generator.py --dataset_folder replica-dataset/ --output dataset_other/

# Data collection apartment0
python data_generator.py --dataset_folder replica-dataset/ --output dataset_apartment0/

# generate .odgt for dataset_other
python generate_odgt.py --train_folder dataset_other --val_folder dataset_apartment0

# generate .odgt for dataset_apartment0
python generate_odgt.py --train_folder dataset_apartment0 --val_folder dataset_apartment0

# Training for other
cd semantic-segmentation-pytorch
python train.py --gpus 1 --cfg config/other.yaml
```

```
cd ..
# Training for apartment0
cd semantic-segmentation-pytorch
python train.py --gpus 1 --cfg config/model_apartment0.yaml
cd ..

# evaluate for model_other
cd semantic-segmentation-pytorch
python eval_multipro.py --gpu 0 --cfg config/model_other.yaml
cd ..

# evaluate for model_apartment0
cd semantic-segmentation-pytorch
python eval_multipro.py --gpu 0 --cfg config/model_aparment0.yaml
cd ..

# output the HW1 rgb to semantic for model_other
cd semantic-segmentation-pytorch
python test.py  --imgs ../HW1_Datacollect/first_floor/rgb --cfg config/model_other.yaml --gpu 0
cd ..

# output the HW1 rgb to semantic for model_apartment0
cd semantic-segmentation-pytorch
python test.py  --imgs ../HW1_Datacollect/first_floor/rgb --cfg config/model_apartment0.yaml --gpu 0
cd ..

# reconstruct of model_other semantic map for first floor
python 3d_semantic_map.py 1 163 pred_semantic_other first_floor

# reconstruct of model_apartment0 semantic map first floor
python 3d_semantic_map.py 1 163 pred_semantic_apartment0 first_floor

# reconstruct of model_other semantic map for second floor
python 3d_semantic_map.py 1 163 pred_semantic_other second_floor

# reconstruct of model_apartment0 semantic map first floor
python 3d_semantic_map.py 1 163 pred_semantic_apartment0 second_floor
```

2. Implement of **custom voxel down**

   There are four steps in my voxel down function.

   a. Construct a cuboid to cover all the points in the point cloud

   b. Slice the big cuboid to the a small cube and the size of small cuboid is voxel size and filter the point which in out of the cube

   c. Compute the center of and the color majority all points in the cube then replace all the points to one points in the cube.

   d. Finally put all the down sampling points to the construct a new point cloud for replacement.

```
def myvoxeldown(pcd, voxel_size):
    """
    Downsample a point cloud with a voxel of given size
    """
```

```
    temp_pcd_down_points = []
    temp_pcd_down_colors = []
    point_np=np.asarray(pcd.points)
    color_np=np.asarray(pcd.colors)

    x_min=np.min(point_np[:,0])
    x_max=np.max(point_np[:,0])
    y_min=np.min(point_np[:,1])
    y_max=np.max(point_np[:,1])
    z_min=np.min(point_np[:,2])
    z_max=np.max(point_np[:,2])

    x_range=x_max-x_min
    y_range=y_max-y_min
    z_range=z_max-z_min

    # filter for x axis
    for i in range(int(x_range//voxel_size)):
        voxel_x_min=x_min+i*voxel_size
        voxel_x_max=x_min+(i+1)*voxel_size
        index = np.logical_and(point_np[:, 0]<=voxel_x_max, point_np[:, 0]>=voxel_x_min)
        if index.sum()==0:
            continue
        point_x = point_np[index]
        color_x = color_np[index]
        # filter for y axis
        for j in range(int(y_range//voxel_size)):
            voxel_y_min=y_min+j*voxel_size
            voxel_y_max=y_min+(j+1)*voxel_size
            index = np.logical_and(point_x[:, 1]<=voxel_y_max, point_x[:, 1]>=voxel_y_min)
            if index.sum()==0:
                continue
            point_xy = point_x[index]
            color_xy = color_x[index]
            # filter for z axis
            for k in range(int(z_range//voxel_size)):
                voxel_z_min=z_min+k*voxel_size
                voxel_z_max=z_min+(k+1)*voxel_size
                index = np.logical_and(point_xy[:, 2]<=voxel_z_max, point_xy[:, 2]>=voxel_z_min)
                if index.sum()==0:
                    continue
                point_xyz = point_xy[index]
                color_xyz = color_xy[index]

                color, counts = np.unique(color_xyz, return_counts=True, axis=0)
                max_count_index = np.argmax(counts)
                most_frequent_color = color[max_count_index]

                voxel_point=np.array(point_xyz).mean(axis=0)
                temp_pcd_down_points.append(voxel_point)
                temp_pcd_down_colors.append(most_frequent_color)
    pcd_down = o3d.geometry.PointCloud()
    pcd_down.points = o3d.utility.Vector3dVector(np.array(temp_pcd_down_points))
    pcd_down.colors = o3d.utility.Vector3dVector(np.array(temp_pcd_down_colors))

    return pcd_down
```

Result and Discussion

Semantic map

1. Model_all_scene
    a. 1F

b. 2F

2. Model_apartment0

a. 1F

b. 2F

Discuss

According to the result, the performance of the model training on the apartment0 is better, we can see the big improvement at the mIOU.

Reference

- Training

  https://hackmd.io/wNGlmMq2RC-lY3l8JhO4SA?view#Configuration

- Model

https://github.com/CSAILVision/semantic-segmentation-pytorch

- mIOU

  https://stackoverflow.com/questions/62461379/multiclass-semantic-segmentation-model-evaluation