



三维点云处理第六次作业



主讲人 Lorenzo



第55页3DCNN输出张量形状

- 3DCNN的输入及输出皆是四维张量，大小用 $[C, D, H, W]$ 来表示
- D, H, W : 套用公式 $W_{out} = \lfloor (W_{in} + 2P - F) / S \rfloor + 1$
 - W_{in} 及 W_{out} 为输入及输出张量在该维度的长度
 - P : padding size。注意是左右两边都padding P 个单位
 - F : kernel size, 卷积核大小。
 - S : stride size, 即步长。
- C : 直接取output channel数

第55页3DCNN输出张量形状

- 第一层的输出张量的形状计算如下，其余两层可用相同的方式计算出来。

⬢ Input: $128 \times 10 \times 400 \times 352$

⬢ Conv3D

- Output channel # 64, kernel (3, 3, 3), stride (2, 1, 1), padding (1, 1, 1)
- Output channel # 64, kernel (3, 3, 3), stride (1, 1, 1), padding (0, 1, 1)
- Output channel # 64, kernel (3, 3, 3), stride (2, 1, 1), padding (1, 1, 1)

⬢ Output: $C' \times D' \times H' \times W'$ – Homework

- Answer is $64 \times 2 \times 400 \times 352$

⬢ Reshape into 2D feature map $2C' \times H' \times W'$

- This is image-like feature map!

$c : 64$

$d : \lfloor (10 + 2 * 1 - 3) / 2 \rfloor + 1 = 5$

$h : \lfloor (400 + 2 * 1 - 3) / 1 \rfloor + 1 = 400$

$w : \lfloor (352 + 2 * 1 - 3) / 1 \rfloor + 1 = 352$

KITTI 3d object detection数据集

● http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d

- [Download left color images of object data set \(12 GB\)](#)
 - [Download right color images, if you want to use stereo information \(12 GB\)](#)
 - [Download the 3 temporally preceding frames \(left color\) \(36 GB\)](#)
 - [Download the 3 temporally preceding frames \(right color\) \(36 GB\)](#)
- [Download **Velodyne** point clouds, if you want to use laser information \(29 GB\)](#)
- [Download camera calibration matrices of object data set \(16 MB\)](#)
- [Download training labels of object data set \(5 MB\)](#)
- [Download object development kit \(1 MB\)](#) (including 3D object detection and [bird's eye view](#) evaluation code)
 - [Download pre-trained LSVM baseline models \(5 MB\)](#) used in [Joint 3D Estimation of Objects and Scene Layout \(NIPS 2011\)](#). These models are referred to as LSVM-MDPM-sv (supervised version) and LSVM-MDPM-us (unsupervised version) in the tables below.
 - [Download reference detections \(L-SVM\) for training and test set \(800 MB\)](#)
 - Qianli Liao (NYU) has put together [code to convert from KITTI to PASCAL VOC file format](#) (documentation included, requires Emacs).
 - Karl Rosaen (U.Mich) has released [code to convert between KITTI, KITTI tracking, Pascal VOC, Udacity, CrowdAI and AUTTI](#) formats.
 - We thank [David Stutz](#) and [Bo Li](#) for developing the 3D object detection benchmark.

KITTI 3d object detection数据集

- 四个数据集：

- data_object_image_2: 点云所对应的二维图片
- data_object_velodyne: 点云数据
- data_object_calib: 点云是在lidar坐标系，而标签却是在rects坐标系，所以需要用它来做转换。
- data_object_label_2: 标签，格式如课件p. 84, 85。

KITTI 3d object detection数据集

● data_object_label_2格式



What does KITTI Label / Result look like?

#Values	Name	Description
1	type	Describes the type of object: 'Car', 'Van', 'Truck', 'Pedestrian', 'Person_sitting', 'Cyclist', 'Tram', 'Misc' or 'DontCare'
1	truncated	Float from 0 (non-truncated) to 1 (truncated), where truncated refers to the object leaving image boundaries
1	occluded	Integer (0,1,2,3) indicating occlusion state: 0 = fully visible, 1 = partly occluded 2 = largely occluded, 3 = unknown
1	alpha	Observation angle of object, ranging $[-\pi, \pi]$
4	bbox	2D bounding box of object in the image (0-based index): contains left, top, right, bottom pixel coordinates
3	dimensions	3D object dimensions: height, width, length (in meters)
3	location	3D object location x,y,z in camera coordinates (in meters)
1	rotation_y	Rotation ry around Y-axis in camera coordinates $[-\pi, \pi]$
1	score	Only for results: Float, indicating confidence in detection, needed for p/r curves, higher is better.

KITTI 3d object detection数据集

● data_object_label_2格式



What does KITTI Label / Result look like?



Label example:

- 000000.txt
 - Pedestrian 0.00 0 -0.20 712.40 143.00 810.73 307.92 1.89 0.48 1.20 1.84 1.47 8.41 0.01



Result example:

- data/000000.txt
 - Pedestrian 0.00 0 -0.20 712.40 143.00 810.73 307.92 1.89 0.48 1.20 1.84 1.47 8.41 0.01 10.0
 - Pedestrian 0.00 0 -0.20 712.40 143.00 810.73 307.92 1.89 0.48 1.20 1.84 1.47 8.41 0.01 8.0
 - Pedestrian 0.00 0 -0.20 712.40 143.00 810.73 307.92 1.89 0.48 1.20 1.84 1.47 8.41 0.01 6.0

kitti_eval评测工具

```
sudo apt update -y
sudo apt install build-essential gnuplot libboost-all-dev -y
git clone https://github.com/prclibo/kitti_eval.git
cd kitti_eval
g++ -O3 -DNDEBUG -o evaluate_3d_offline evaluate_object_3d_offline.cpp
./eval_detection_3d_offline gt_dir result_dir
```


PointRCNN开源项目

- 以下是几个实作了PointRCNN的开源项目：

Projects	Ubuntu	Python	PyTorch	TensorFlow
sshaoshuai/PointRCNN	14.04/16.04	3.6+	1.0+	
open-mmlab/OpenPCDet	14.04/16.04	3.6+	1.1~1.7	
intel-isl/Open3D-ML		3.6+	1.6	2.3

修改sshaoshuai/PointRCNN代码



深蓝学院
shenlanxueyuan.com

- 这个项目是基于PyTorch 1.0。为了使它能在PyTorch 1.7或Windows下运行，需要自行修改代码。需要修改的部分涉及sshaoshuai/PointRCNN本身及其子项目sshaoshuai/Pointnet2.PyTorch里的代码。

For PyTorch 1.2+:

- 将 `AT_CHECK` 替换成 `TORCH_CHECK`
- 将 `THCState_getCurrentStream(state);` 替换成 `at::cuda::getCurrentCUDASTream();`

For Windows:

- `os.system('cp aaa bbb')` 改为 `os.system('copy aaa bbb')`
- `long` 改为 `long long`
- `unsigned long long xxx[col_blocks];` 改为 `unsigned long long *xxx = new unsigned long long[col_blocks];`
- `lib/utils/iou3d/src/iou3d.cpp` 添加 `#include <torch/serialize/tensor.h>` 及 `#include <pybind11/pybind11.h>`
- `lib/utils/iou3d/src/iou3d_kernel.cu` 里 `const float EPS = 1e-8;` 改为 `#define EPS 1e-8`

PointRCNN测试

- 因为预训练的模型只接受三个通道，所以需要将intensity排除。
 - 修改lib/config.py: `__C.RPN.USE_INTENSITY = False`
- 进行测试：
 - `python eval_rcnn.py --cfg_file cfgs/default.yaml --ckpt ../PointRCNN.pth --batch_size 1 --eval_mode rcnn --set RPN.LOC_XZ_FINE False`

Final project preview - 建构分类数据集



- 要利用KITTI 3D object detection dataset中建构分类数据集，代表我们需要将物体点云从数据集中抽取出来，并为抽取出来的点云打上类别标签。
- “将物体点云从数据集中抽取出来”可以分为三个步骤，分别是读取点云，读取包围框，以及获取所关注物体的点云。
- 在实现的过程中，我们可以从<https://github.com/sshaoshuai/PointRCNN>项目里寻找用得上的函数，对它们进行改写。

建构分类数据集 - 读取点云

- kitti_rcnn_dataset.py的get_rpn_sample函数

- 读取点云
- 读取calibration file
- 调用calib.lidar_to_rect对进行点云做校正

```
calib = self.get_calib(sample_id)
# img = self.get_image(sample_id)
img_shape = self.get_image_shape(sample_id)
pts_lidar = self.get_lidar(sample_id)

# get valid point (projected points should be in image)
pts_rect = calib.lidar_to_rect(pts_lidar[:, 0:3])
pts_intensity = pts_lidar[:, 3]
```

建构分类数据集 - 读取包围框

- kitti_rcnn_dataset.py的get_proposal_from_file函数
 - 读取“长宽高+中心点”格式的包围框 (gt_boxes3d)
 - 将它转为“角落”格式的包围框 (gt_corners)

```
gt_obj_list = self.filtrate_objects(self.get_label(sample_id))
gt_boxes3d = kitti_utils.objs_to_boxes3d(gt_obj_list)

roi_corners = kitti_utils.bboxes3d_to_corners3d(roi_boxes3d)
gt_corners = kitti_utils.bboxes3d_to_corners3d(gt_boxes3d)
```

建构分类数据集 - 获取物体点云

- 已有点云及“角落”格式的包围框
- 自己写一个能判断“点云中的哪些点落在包围框内”的函数就能实际将所关注的物体从kitti点云中抽取出来





深蓝学院
shenlanxueyuan.com

感谢各位聆听 !
Thanks for Listening

