D4LCN: Learning Depth-Guided Convolutions for Monocular 3D Object Detection (CVPR 2022)

Requirements

- **Cuda & Cudnn & Python & Pytorch**

  This project is tested with CUDA 8.0/9.0, Python 3, Pytorch 0.4/1.0/1.1, NVIDIA Tesla V100/TITANX GPU. And almost all the packages we use are covered by Anaconda.

  Please install proper CUDA and CUDNN version, and then install Anaconda3 and Pytorch.

- **My settings**

```
source ~/anaconda3/bin/activate (python 3.6.5)
(base)  pip list
torch                              1.1.0
torchfile                          0.1.0
torchvision                        0.3.0
numpy                              1.14.3
numpydoc                           0.8.0
numba                              0.38.0
visdom                             0.1.8.9
opencv-python                      4.1.0.25
easydict                           1.9
Shapely                            1.6.4.post2
```

Data preparation

Download and unzip the full KITTI detection dataset to the folder **/path/to/kitti/**. Then place a softlink (or the actual data) in **data/kitti/**. There are two widely used training/validation set splits for the KITTI dataset. Here we only show the setting of **split1**, you can set **split2** accordingly.

```
cd D4LCN
ln -s /path/to/kitti data/kitti
ln -s /path/to/kitti/testing data/kitti_split1/testing
```

Our method uses DORN (or other monocular depth models) to extract depth maps for all images. You can download and unzip the depth maps extracted by DORN here and put them (or softlink) to the folder **data/kitti/depth_2/**. (You can also change the path in the scripts **setup_depth.py**)

Then use the following scripts to extract the data splits, which use softlinks to the above directory for efficient storage.

```
python data/kitti_split1/setup_split.py
python data/kitti_split1/setup_depth.py
```

Next, build the KITTI devkit eval for split1.

```
sh data/kitti_split1/devkit/cpp/build.sh
```

Lastly, build the nms modules

```
cd lib/nms
make
```

Training

We use visdom for visualization and graphs. Optionally, start the server by command line

```
sh visdom.sh
```

The port can be customized in config files. The training monitor can be viewed at http://localhost:9891.

You can change the batch_size according to the number of GPUs, default: 4 GPUs with batch_size = 8.

If you want to utilize the resnet backbone pre-trained on the COCO dataset, it can be downloaded from git or Google Drive, default: ImageNet pretrained pytorch model. You can also set use_corner and corner_in_3d to False for quick training.

See the configurations in **scripts/config/depth_guided_config** and **scripts/train.py** for details.

```
sh train.sh
```

Testing

We provide the weights, model and config file on the val1 data split available to download.

Testing requires paths to the configuration file and model weights, exposed variables near the top **scripts/test.py**. To test a configuration and model, simply update the variables and run the test file as below.

```
sh test.sh
```