CS230

An Exploration of Neural Net Architectures for 3D Region-Proposal Networks and Object Detection

Shawn Hu
Department of Computer Science
Stanford University
shawnghu@stanford.edu

1 Motivation and Problem Statement

Classically, in autonomous driving systems' perception modules, 2D object detection algorithms such as Faster R-CNN are used to detect obstacles (typically cars and humans) using camera input. However, avoiding obstacles requires an internal representation of their 3D location, and 2D object detection is really only a bound on the projection of the obstacle into the image plane. Thus, stereo vision techniques, radar or LiDAR systems or other bird's eye view (BEV)-plane sensors are frequently used to determine the distance to these obstacles, and either classical computer vision techniques or other task-specific, proprietary algorithms may be used to combine both sets of input data to obtain the internal 3D representation of obstacle locations. These algorithms may be highly sensitive to various parameters, which may sometimes require a repeated, painstaking hand-calibration process, which may sometimes have to be redone every time the setup of a car changes.

However, instead of using techniques to handle the specific task of fusing the 2D object detections with LiDAR input, it is possible to learn the task of inferring a 3D representation of obstacles directly, using both the LiDAR input and camera input as inputs to a deep neural network which directly outputs 3D bounding boxes. More concretely, we will be tackling the task of locating an car, establishing a rectangular-prism boundary on its 3D location, and determining its orientation. This project builds on the results of AVOD [6], an algorithm which in the past achieved the top score on the KITTI benchmark by a wide margin (and to date, is a close second to the state of the art), by exploring modifications to AVOD's model architecture.

2 Related work

- Our architecture's (very common) scheme of regressing from a series of anchors, selecting from them using a region proposal network, and feeding the proposals to a final object detecting network originated in the "Faster R-CNN" [11] series of papers.
- [13] was the first paper to extend this approach to 3D prior anchors and RPN, instead of using a 2D RPN to perform 3D object detection.
- [2] and [8] perform the 3D object detection task with surprising performance using only 2D images as input, one by leveraging explicit priors about the way cars look and the other by observing that the projection of the 3D box into the image plane must be contained in the 2D box.
- [9] is the only other algorithm competitive with AVOD on the KITTI leaderboards. Notably, they start by detecting objects with 2D camera object detection, and use geometric techniques to extend the 2D bounding box to a small set of priors on the 3D location of that object. One major limitation of this approach is that it's fundamentally bounded in performance by the performance of the original 2D image object detector, which may be problematic under certain lighting conditions.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Dataset and Preprocessing

We work with the KITTI Vision Benchmark Suite [5], the canonical dataset and evaluation metric for 3d object detection in autonomous driving scenarios. The dataset consists of around 100 rural, urban, and highway scenes, gathered by driving a car with various sensors around a mid-size city. These scenes are discretized at a rate of 10 Hz into 7481 training datapoints, which we split evenly between training and validation. For our purposes, each datapoint consists of a left roof-mounted camera image (1224px * 370px resolution), LiDAR input data (Velodyne point clouds, stored in a standardized format as a binary float matrix), and calibration data used to align the inputs and crop the LiDAR input to the scope of the camera. The dataset includes scenes involving up to 30 cars and 15 pedestrians/cyclists, and the labels were generated by humans.

We augmented the data using both horizontal flipping and PCA-based jittering (both available in the AVOD source code). (To be specific, we performed PCA on the dataset, then added noise equal to the principal components multiplied by a random number sampled from a Gaussian with mean 0 and SD 0.1.) These were the only simple augmentations we found appropriate for the task- i.e, we reasoned that e.g rotations, vertical flips, or blurring would violate or destroy important features of the dataset.

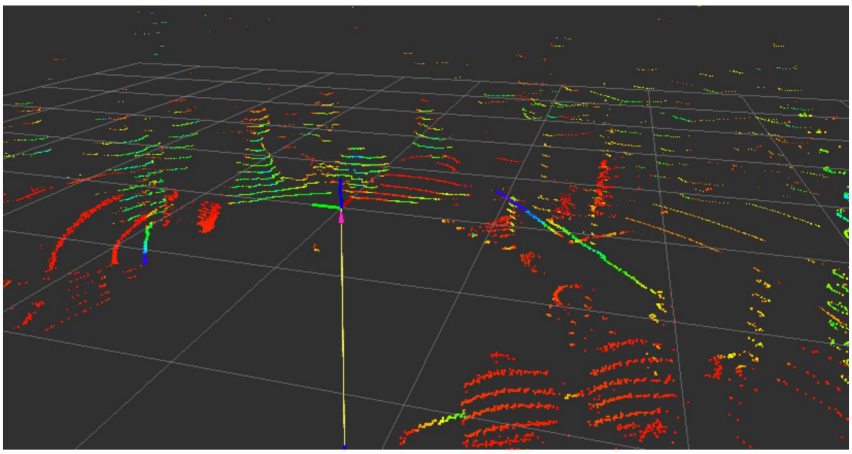

Figure 1: Visualization of LiDAR point cloud in ROS (point cloud not actually from our dataset, here to illustrate the nature of LiDAR data), from [1]

Figure 2: Sample input image from the dataset.

4 Evaluation

The performance of a model is evaluated using a pre-defined metric described in precise detail in a paper accompanying the KITTI dataset [5]. It is based on a variation of the PASCAL criteria [3] for 2D images. In broad strokes, we take the 3D IoU of the predicted and true bounding boxes, and mark our object detection correct if it exceeds 0.7. Then, a precision-recall curve is drawn for the model's correctness on the test set. At each of 11 different values for recall, we scale the precision by a measure of the orientation correctness (by taking the average cosine similarity between true

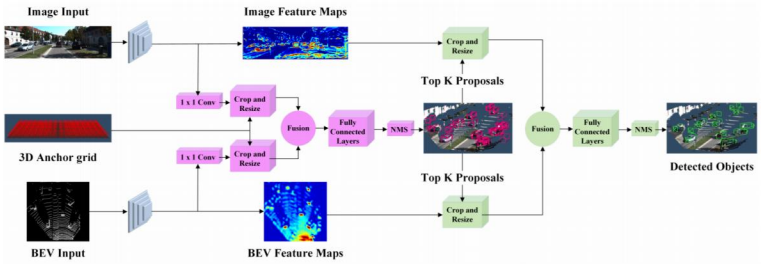

Figure 3: The model architecture. Architecture and figure inherited from AVOD [6]

and predicted orientation angles over all samples, and then normalizing it to be between 0 and 1; we penalize duplicate predictions by including them in the calculation with a similarity of 0). We then take the average of these scaled precision values over the 11 recall values. This metric is very complex, but was motivated by a desire to thoroughly capture the performance of models at a variety of levels of recall, as well as to penalize excess predictions. Since this metric is based on the Average Precision metric used in the PASCAL criteria, we will follow the convention of the AVOD authors and refer to it as the AP.

The metric is scored separately on the car, pedestrian, and cyclist tasks for easy, medium, and hard subsets of the test set (labeled according to the occlusion of the obstacle and the height of its bounding box, see [6]), on a few different tasks (3d detection, 2d detection, bird's eye view detection). For a simple single evaluation metric, we optimized for the mean of the scores for the three difficulties for the car class on the 3d object detection task.

5 Architecture and Training

Predictions are made using using an end-to-end network which both generates region proposals and uses those region proposals to make object location and classification predictions. The AVOD paper [6] describes and motivates the architecture in thorough detail; we first offer a high-level description of the AVOD architecture and describe modifications in following sections.

5.1 Feature Extraction

The LiDAR data and the image data are passed through two identical (but separate; i.e, no shared parameters) convolutional feature extractors to generate two 256-filter feature maps. The layer structure of these extractors is motivated by and shaped similarly to VGG [12], but truncated at the conv4 layer and with half the filters.

5.2 Region-Proposal Network

100,000 prior anchor boxes for each class are determined by performing clustering on the training set, and these anchors are projected onto the planes of the two inputs to generate 7x7 "feature crops". One of the major innovations of AVOD is to vastly downsample the feature crops at this point with 1x1, 1-filter convolutional layer to reduce the number of parameters for the following operations. For each anchor, these feature crops are element-wise averaged and then used as input to a fully connected network which outputs the region-proposal parameters (i.e, the object-credence on the anchors, as well as learned offsets). 2D NMS in the BEV plane is used to remove overlapping proposals, and the top 1024 are saved.

5.3 Object Detector

In a manner similar to the process with the anchors above, the learned proposals are projected onto the two input planes. But now, since there are far fewer proposals than anchors, the full, 256 filter feature maps from both images are used as input to an object detection network to detect and predict the offsets from the region propoals, as well as orientation vectors (specifically, we output the x-z coordinates of four corners, the top and bottom y-coordinates of the box, and the x and z lengths of the orientation vector. This choice of box overparameterization is used to make it easier for the

network to fit the corners, as opposed to needing to implicitly learn to transform an axis-aligned box by a predicted orientation angle).

For the outputs of both the RPN and the object detector, we include multiple losses:

- Cross-entropy loss on whether or not an object has been correctly detected and, for the object detector, classified;
- Smooth L1 loss on the offset of the 3D bounding boxes, if the IoU with a ground truth box exceeds 0.65 and the class prediction/objectness is correct;
- Smooth L1 loss on the offset of the orientation vector, if the same conditions as above are met.

We then minimize a [1, 5, 1] weighted sum of these losses using a standard Adam optimizer with an initial learning rate of 0.0001, exponentially decaying by a factor of 0.8 every 30k steps, for 120k steps in total, taking a total of about 16 hours. We regularize with layers with 0.5 probability of dropout in the FC layers of both the RPN and object detector, and L2 regularization of 0.0005.

6 Experiments

6.1 Changes to the Architecture

- **Concatenation for Fusion:** The element-wise mean of the two feature maps for fusion seemed to us to be poorly motivated. In particular, the BEV feature map comes from data in the xz plane, whereas the image feature map comes from data in the yz plane, so there is no reason to believe that there is a relationship between the corresponding entries in the two feature maps. Moreover, the mean isn't a parameterized operation, so the model can't learn anything here; taking the mean of the two feature crops just seems to irrecoverably destroy salient information. We simply replaced the mean with a concatenation of the two vectors, which at least has the effect of keeping all the extracted feature data intact.
- **Fusion Location:** We experimented with fusing the input maps both before and after the FC layers, thinking that perhaps concatenating spatial data and then putting them through FC layers would severely overparameterize the relationship between the two types of data, making it hard to learn.
- **Intermediate Downsampling:** The decision to compress the 256 filters of the feature map into 1 for the region-proposal network seemed somewhat extreme. We found it somewhat implausible that all the salient information associated with a location in a single feature map could be expressed with a single number, so we experimented with changing the number of filters in the 1x1 convolution to 4, 16, and 32. The nice thing about this approach is it intuitively occupies a better location on the tradeoff between memory and expression- 16 or even 4 numbers can capture substantially more complex info than just 1, while barely increasing the memory load.

6.2 Miscellaneous Parameter Tuning

- After looking at the qualitative results of some models, we hypothesized that the performance was bottlenecked by RPN performance (despite the above fixes). We tried changing the ratio of objectness loss weights to regression loss weights to 0.2, 0.4, 0.6, 1, and 3.
- To accommodate the fact that our above experiments generally increase the complexity of the model, we experimented with increasing the dropout odds to values between 0.6 and 0.7 and the weight penalty by factors of 2, 4, and 10.
- We tried changing the number of clusters for prior anchor generation.

7 Results and Analysis

With respect to the above experiments, our best model used concatenation for fusion, downsampled the RPN input to four filters, kept the original regularization values, and used two clusters for anchor generation. We optimized for the 3D object detection task, but provide our performance on a few other metrics for interest:

AVOD	easy	medium	hard	mean
2D Object Detection	89.2	79.9	79.7	83.1
BEV Object Detection	87.8	78.4	78.0	81.4
3D Object Detection	**80.0**	**66.0**	**65.7**	**70.6**
Our Model	easy	medium	hard	mean
2D Object Detection	89.6	87.0	79.7	85.4
BEV Object Detection	88.9	85.6	78.3	84.3
3D Object Detection	**81.1**	**67.1**	**65.8**	**71.3**

Table 1: Mean average precision of AVOD compared to our model on three tasks, as evaluated on our validation set.

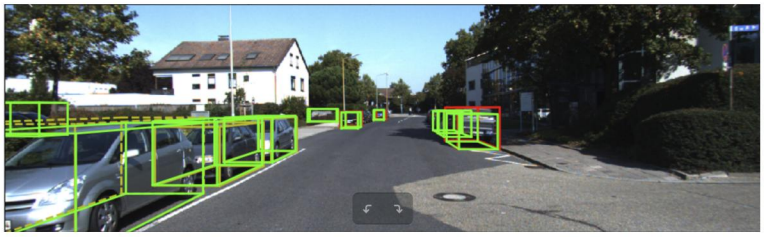

Figure 4: A sample detection from our trained model. Red/yellow represent ground truth where green presents our model's prediction.

We were surprised to find that the changes to our RPN's expressiveness didn't have much of an effect on the final performance. One simple hypothesis for this is that the performance is not truly bottled by the performance of the RPN, which can output 1024 proposals, but rather by the end network's ability to regress to good enough object locations. Our initial estimates of the RPN performance might have been colored by a bad estimate of what a 3D IOU of 0.7 looks like- after all, our visualizations count a 3D IOU of less than 0.7 as a complete failure, so we were exposed to bounding boxes which looked convincingly precise even if our model was struggling to accurately regress object locations.

Moreover, we seem to still have a substantial gap between validation and training performance that we can't seem to close using conventional regularization techniques. The most simple explanation for this is that our training set is far too small to generalize completely, at only about 3500 datapoints (on the contrary, thinking about the difficulty of the task, we think the performance of both our model and the baseline is remarkable).

8 Conclusion/Future Work/Extensions

We were able to improve incrementally on the original architecture by making simple observations about some of its limitations, but not as much as we imagined might be possible. Many of our efforts to encourage the improved performance of the RPN were not very effective, and may have been due to a slightly misguided view of the model's limitations. Ultimately, we hypothesize that the model and algorithm are sufficiently powerful to work much better simply given a larger dataset. There are still many branches of interest down this avenue- some reasonable or interesting experiments to perform include:

- Further improve the existing scheme, especially model regularization or dataset augmentation (one basic thing we didn't try is increasing the magnitude of our PCA whitening);
- Experiment with using a distinct feature extractor for BEV input, since as a different type of input data it might be amenable to different heuristic architectures;
- Explore the effects of foreign datasets (such as those gathered by other self-driving cars) on the performance of the model, both as training and as test data;
- Explore the ability of the model to predict car locations coming from data generated from different sensor setups (e.g, hood-mounted cameras);
- Use a similar scheme to train models using stereo vision;
- Train a single model which can jointly detect both cars and people at high performance.

References

[1] from the public page of the rochester institute of tech senior design program. http://edge.rit.edu/edge/C15504/public/Pictures.

[2] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep MANTA: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. *CoRR*, abs/1703.07570, 2017.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[4] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. *arXiv preprint arXiv:1712.02294*, 2017.

[7] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[8] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. *CoRR*, abs/1612.00496, 2016.

[9] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CoRR*, abs/1711.08488, 2017.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[11] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[13] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 808–816, June 2016.

[0]