3D YOLO with Uncertainty

Yuanchu Dang
yd2466@columbia.edu

Wei Luo
wl2671@columbia.edu

Neil Menghani
nlm2138@columbia.edu

Abstract

In this project, we implement and improve Complex YOLO, a state-of-the-art real-time 3D object detection network for point clouds Lidar data. Complex YOLO is a new framework that expands YOLOv2, a fast 2D standard object detector for RGB images, with a specific complex regression strategy to estimate multi-class 3D boxes in the Cartesian space. More specifically, Complex YOLO augments traditional CNNs with Euler-Region Proposal Network (E-RPN) to estimate the pose of the object by adding an imaginary and a real fraction to the regression network. This innovative E-RPN approach supports to generalize well during training and also leads to greater efficiency. In this project, we implement and test various architectures and parameters to slightly improve the accuracy of the Complex YOLO algorithm. Furthermore, we add additional dropout layers and generate uncertainty estimation for our own predictions using Bernoulli masks when running the model on the test set, a technique that the original Complex YOLO study did not leverage. Last but not least, we also attempt to improve the model in cases when it wrongly predicts overlapping bounding boxes, albeit our proposal turns out to have limited effect.

1. Introduction

Object detection and classification has been a field of great interest for many years. From as early as 1963, researchers have been studying how machines could perceive objects [13]. In deep learning, object detection and classification is well-known for 2D bounding box regression using images [14]. Because deeper models could give accurate results but must be trained and tested for much longer time, there is a trade-off between accuracy and efficiency. With the field of autonomous driving gaining momentum, constructing models that run efficiently while not losing too much accuracy has emerged as an important topic. Networks such as [12] are sufficiently efficient that they are capable of running on embedded devices, making true autonomous driving closer to reality.

Further, given the complicated ethics surrounding autonomous vehicles, the room for error in detecting objects is extremely slight. Having the ability to quantify the uncertainty of particular classified objects would prove quite useful in a practical context, as this information could influence the decisions of such autonomous machines. Thus, we look to build an accurate and efficient model that not only detects and classify objects but also provides the uncertainty of these classifications.

Our main contributions are as follows:

1. We effectively incorporated uncertainty into 3D object detection while preserving average precision.

2. We came up with our own way of projecting predictions back to 3D using homography and view-point geometry.

3. We attempted to improve models on overlapping bounding box predictions by constructing our customized loss function.

2. Related Work

Throughout the years, researchers have experimented with a variety of approaches such as geometric primitives and sliding-windows [9] to approach the problem of object detection. Recent approaches such as R-CNN perform object detection by generating bounding boxes and running classifiers, then proceeding on to post-processing steps to refine the boxes and predictions based on other objects in the scene [11]. Such methods, however, can be slow - humans can glance at an image and instantly identify the objects in that image. To build a fast system that detects objects in real time, Redmon et al. [11] use a single convolutional network called "YOLO" ("You only learn once") to predict both bounding boxes and class probabilities simultaneously. This work is still once removed from what we expect from object detection in reality because in our daily lives, we recognize objects on the 3D scale.

In order to mirror these daily experiences, the Complex YOLO [14] model takes YOLO to the 3D scale. In particular, it is able to make predictions about 3D bounding boxes and object classifications at the same time, using Lidar-generated point cloud data. The model is comprised

of a Convolutional Neural Network (CNN) [8] and an Euler-Region Proposal Network (E-RPN) [12], where the CNN's predictions about bounding boxes and classifications are input to the E-RPN which adds an imaginary and a real fraction to the network to estimate the pose and orientation of the object. Complex YOLO is shown to surpass the leading methods for 3D object detection on the KITTI benchmark suite [6] in terms of both accuracy and efficiency and thus is selected to form the basis for our work.

On a separate note, there has been significant previous work related to measuring uncertainties in models that use deep networks. [4] and [3] explore the use of Bernoulli dropout layers with sampling at prediction time in order to model uncertainty. [10] builds on this work by introducing other techniques such as Gaussian dropout to improve test set accuracy. These studies all take a generalized approach to exploring these methods rather than focusing on one specific application, such as object detection. [1] uses the techniques explored in this previous work to measure prediction uncertainty in an object detection model that looks to identify people's trajectories. In our project, we look to apply uncertainty modeling techniques used in these previous studies to our improved Complex YOLO model. In this way, our model contains novel elements that build on previous work.

3. Problem Formulation and Data

3.1. Basic Problem Formulation

We formulate our core problem as the following: Given an image, our goal is to draw 3D boxes around detected objects and classify them. This problem then breaks down into two parts: drawing bounding boxes around objects, and generating classifications for the identified boxes. These two tasks can be simultaneously achieved by the Complex YOLO algorithm 4 in an accurate and efficient manner.

3.2. Innovation

In addition to the basic box and class prediction tasks, we also intend to gauge the uncertainty for our own predictions about the locations and sizes of the bounding boxes. To accomplish this goal, we borrow techniques from [5] and make use of active dropout layers and Bernoulli masks in the evaluation phase to repeatedly sample the prediction distribution. We visualize the sampling distribution in the form of heat maps.

Moreover, we also attempt to improve the original model in cases where it wrongly predicts overlapping bounding boxes, by adding another penalty term to the loss function that explicitly penalizes the IoU between the predicted boxes. This attempt turns out to have limited effects.

3.3. Data

We used the KITTI benchmark suite [6] provided by the courtesy of Karlsruhe Institute of Technology and Toyata Technological Institute at Chicago. This suite contains computer vision data gathered using a Velodyne laser scanner and a GPS localization system. The images in the suite include driving scenes from the city of Karlsruhe, and contains ground-truth labels for Car, Van, Truck, Pedestrian, Person sitting, Cyclist, and Tram. The specific 3D data set that we use from this suite contains four core pieces:

- Velodyne point clouds (29 GB): Information about the surrounding for a single frame gathered by Velodyne HDL64 laser scanner. This is the primary input data we use.
- Left color images of object data set (12 GB): The cameras were one color camera stereo pairs. We use left Images corresponding to the Velodyne point clouds for each frame.
- Camera calibration matrices of object data set (16 MB): Used for calibrating and rectifying the data captured by the camera and sensor.
- Training labels of object data set (5 MB).

Put together, these four pieces constitute a complete set of Lidar point clouds data in RGB channels with ground-truth bounding boxes and object labels. Specifically, using the techniques described in [14], each frame of the 3D point cloud can be converted into one birds-eye-view RGB map.

4. Methods

We follow the design as put forth by [14]. The network takes a birds-eye-view RGB-map as input, and the entire pipeline consists of two distinct parts. First, it uses a simplified YOLOv2 CNN architecture to condense and extract contextual features from the input. Next, it extends the output layer of the CNN with a complex angle regression and E-RPN to detect accurate multi-class oriented 3D objects. Both parts are described in greater details below.

4.1. CNN

The CNN part has 18 convolutional and 5 maxpool layers, as well as 3 intermediate layers for feature reorganization respectively. Detailed parameters including the number and size of filters at each layer can be found in the original paper [14]. To incorporate uncertainty, we added an extra Bernoulli dropout layer after the last convolutional layer.

4.2. E-RPN

The E-RPN step synthesizes the 3D positions, dimensions, base probability, class scores and orientation from the

incoming feature map output by CNN. Specific transformations are defined below:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_x) + c_y \\ b_w &= p_w e^{t_w} \\ b_l &= p_l e^{t_l} \\ b_\phi &= arg(|z|e^{ib_\phi}) = arctan_2(t_{Im}, t_{Re}) \end{aligned} \tag{1}$$

According to the authors [14], E-RPN can help make more accurate estimations about object orientations by using an imaginary and real fraction directly embedded into the network.

4.3. Anchor Boxes

Similar to how we give initial weights to networks to work off with, we can set anchor boxes that the bounding box regressor can use. Then, instead of predicting bounding boxes from scratch, the model can adjust the anchor boxes. Our anchor boxes are defined based on the distribution of boxes in the KITTI data set for vehicle (heading up) size, vehicle (heading down) size, cyclist (heading up) size, cyclist (heading down) size, and pedestrian (heading left) size.

4.4. Loss Function

Similar to the architectural design, the loss function consists two parts as well. The first part of the loss function is simply a sum of squared errors from YOLO 2D, while the second part is built on the Euler regression logic and is defined to be the difference between the complex numbers of prediction and ground truth.

More specifically,

$$L_{Total} = L_{YOLO} + L_{Euler} \tag{2}$$

Here, the YOLO 2D loss is:

$$\begin{aligned} L_{YOLO} &= \lambda_{coord} \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ &+ \lambda_{coord} \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 \\ &+ (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\ &+ \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 \\ &+ \lambda_{noobj} \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\ &+ \Sigma_{i=0}^{S^2} 1_{i}^{obj} \Sigma_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \tag{3}$$

And the Euler loss is:

$$L_{Euler} = \lambda_{coord} \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} |e^{ib_\phi} - e^{i\hat{b}_\phi}| \tag{4}$$

where λ_{coord} is a scaling factor to ensure stable convergence.

4.5. Uncertainty

Typical feed-forward neural networks linearly transform input, apply activation functions on each dimension, and then repeat the cycle again, providing point estimates without information of uncertainty [2]. Even in cases when probabilities themselves become subjects of interest, naive statistics such as the softmax function fail to yield sufficient information about the uncertainty of a model. [2].

Dropout can be thought of as the random vector (scaled by $\frac{1}{1-rate}$) from the Bernoulli distribution with dropout rate as the probability of success [15]. The length of the random vector equals the number of hidden units on the hidden layer prior to the dropout layer [15]. It turns out that, as [4] discovered, implementing a dropout layer approximates the Gaussian process quite well. Given a model trained with dropout having an input $\boldsymbol{x}^*$, we make some prediction $\hat{\boldsymbol{y}}^*$ [2]. Given this prediction from our network, we can compute an expected model mean output $\mathbb{E}(\boldsymbol{y}^*)$ and a predictive variance $Var(\boldsymbol{y}^*)$ which signals how confident the model is about its output [2]. Since the dropout network is simply an approximation of the Gaussian process, we are able to determine predictive mean and predictive variance according to the following equations:

$$\mathbb{E}(\boldsymbol{y}^*) \approx \frac{1}{T} \sum_{t=1}^{T} \hat{\boldsymbol{y}}_{\boldsymbol{t}}^*(\boldsymbol{x}^*) \tag{5}$$

where T represents the number of repetitions and

$$\begin{aligned} Var(\boldsymbol{y}^*) &\approx \tau^{-1} \boldsymbol{I}_D \\ &+ \frac{1}{T} \sum_{t=1}^{T} \hat{\boldsymbol{y}}_{\boldsymbol{t}}^*(\boldsymbol{x}^*)^T \hat{\boldsymbol{y}}_{\boldsymbol{t}}^*(\boldsymbol{x}^*) \\ &- \mathbb{E}(\boldsymbol{y}^*)^T \mathbb{E}(\boldsymbol{y}^*) \end{aligned} \tag{6}$$

where $\tau = \frac{l^2(1-p)}{2N\lambda}$ [2].

As previous studies [3] and [4] do, we model uncertainty using Bernoulli dropout as specified above. Adding a dropout layer with dropout rate $x = 0.2$ and predicting our test output 1000 times, we are able to sample the distribution of the predictions. We primarily apply the dropout and sampling process to predictions regarding the locations and sizes of the bounding boxes.

4.6. Evaluation

Evaluating bounding box predictions is more nuanced than regular classification, because there is no strict "right" or "wrong". Instead, our predicted boxes and the ground-truth boxes might overlap. We take advantage of this fact, and define

Intersection over Union (IoU) of box C and D as

$$\frac{Area(C \cap D)}{Area(C \cup D)}.$$

Then we can define a prediction $(box_p, class_p)$ as "correct" with respect to ground-truth $(box_g, class_g)$ if $class_p = class_g$ and $IoU(box_p, box_g) > threshold$, where the threshold is usually 0.5.

Precision measures how accurate predictions are — that is, the number of correct positive predictions our of total positive predictions — as shown in the following equation.

$$Precision = \frac{TP}{TP + FP}$$

Recall measures how well positives are found — that is, the number of correct positive predictions out of total positives — as shown in the following equation.

$$Recall = \frac{TP}{TP + FN}$$

To evaluate our model, we use mean average precision (mAP), often called average precision (AP), which is the primary metric to evaluate object detection architectures such as CNNs [7]. AP is measured by taking the average of the maximum precision value at the different recall levels [7]. The precise definition is as follows:

$$AP = \frac{1}{n} \sum_{r} p_{interp} r$$

where r represents the set of recall values being examined, n represents the cardinality of this set, and [7]

$$p_{interp} r = \max_{\widetilde{r} \geq r} p(\widetilde{r})$$

.

5. Results

5.1. Training and Testing Details

We trained the model on a Google Cloud VM instance with 4 CPUs, 1 NVIDIA Tesla P100 Virtual Workstation and 16 GB of memory. We split the data set into a training set of 6000 images and a test set of 1480 images. The learning rate η is set to be gradually decreasing with respect to the epoch iteration. Specifically, we set:

$$\eta = \begin{cases} 1e{-}5 & \text{epoch} < 4 \\ 1e{-}4 & 4 \leq \text{epoch} < 80 \\ 1e{-}5 & 80 \leq \text{epoch} < 160 \\ 1e{-}6 & 160 \leq \text{epoch} \end{cases}$$

We set momentum to 0.9, weight decay to 0.0005, and dropout probability to 0.2. During training, we use a batch size of 12 images and run 400 epochs. When making predictions, we generate bounding boxes whenever the confidence for those predictions exceed 0.5. Moreover, in determining if a bounding box is correct, we apply an IOU ratio threshold of 0.5.

5.2. Average Precision

Below we report our average precision. The first table shows values for both training and test sets using our YOLO with uncertainty, and the second table compares our AP for cars with that achieved by Complex YOLO.

Average Precision (AP)		
Training		
Car	Van	Truck
97.14%	92.08%	76.98%
Testing		
Car	Van	Truck
80.01%	27.58%	3.17%

Model	Class	AP
Complex YOLO	Car	85.89%
Uncertainty YOLO	Car	80.01%

In the first table, we show AP values for selected classes of objects. There are 8 classes of objects in total, encompassing Car, Van, Truck, Pedestrian, Person, Cyclist, Tram, and Misc, but the data set is heavily biased towards Car objects admittedly. In fact, more than 80% of our data are Car objects, so we observe the highest AP values occur for that particular class. Some other categories suffer from limited number of training samples so have modest AP values or sharp drop from training to testing.

In the second table, we see that our AP for cars is comparable to that obtained by Complex YOLO without uncertainty. Such AP allows the model to generate uncertainty without having to suffer in precision. In applied terms for autonomous vehicles, this means that the model can make accurate predictions, and inform the human if it finds that it is uncertain about some predictions. We do not compare our AP for the other classes with those from Complex YOLO, because our dataset has different distribution than the one used for regular Complex YOLO.

5.3. Projection back to 3D

This section steps through the process through which we can project our predictions back to the original 3D image.

Suppose our image for prediction is the Lidar birds-eye-view point-cloud corresponding to Figure 1.

Figure 1. Original image that the Lidar point-cloud corresponds to.

After letting our model predict on the point-cloud image, we obtain prediction boxes, as shown in Figure 2.

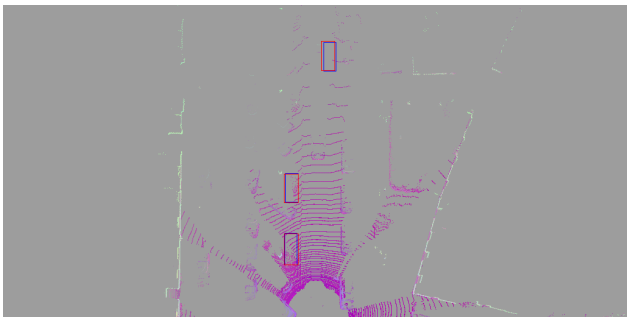

Figure 2. Predictions made on the point-cloud. Blue boxes indicate ground-truth, and red boxes indicate predictions.

To map this back to 3D, we need to change the view from birds-eye-view to the canonical front view. We select four view anchor points from each of the two views, and determine a homography between them. Specifically, let (x, y) be a view anchor point, then our homography is a 3×3 matrix H that satisfies:

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

We can then apply this matrix to the entire image, thereby "warping" from the birds-eye-view point-cloud to a front-view point-cloud (Figure 3).

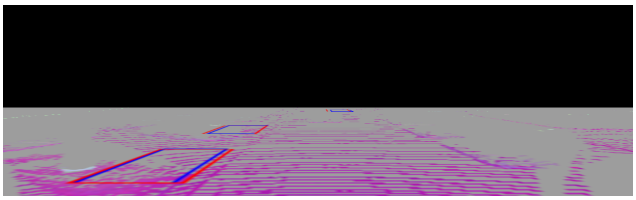

Figure 3. Birds-eye-view point-cloud Warped to front-view.

To turn 2D boxes into 3D ones, we use predefined heights for different classes. Then we can shift the 2D box by an amount proportional to the height h to obtain the top rectangle of the bounding box. Because objects further from view appear smaller, this shift is not parallel to the y-axis (where y-value is 0 at top of image). Rather, we penalize points that are already further up, indicating that they are further from view:

$$\Delta y = h(\frac{y}{512})^2.$$

Using this equation to draw the top parts of the bounding boxes and adding vertical connecting edges, we obtain the full 3D result (Figure 4).

Figure 4. Predictions warped to 3D and displayed on the original image.

5.4. Uncertainty

Using our model with Bernoulli mask, We ran the prediction step 1000 times to model the uncertainty of our classifications on the test set. To visualize uncertainty, we created a frequency map, where each integer in the map is the number of predicted boxes a corresponding pixel in the point-cloud is in. For example, a pixel contained in ten predicted boxes would have frequency 10. We then used a color map to turn these frequencies to colors, thereby obtaining a heat map. Table 5.4 shows point-cloud images containing prediction boxes and also the corresponding heat maps. We do not project back confidence heat maps, because the notion of turning 2D confidence back into 3D is not well-defined; one patch of confidence in 2D would have the same value for all heights in 3D.

5.5. Loss Innovation

Last but not least, we add a customized loss term to the existing objective function targeting the IoU summation of overlapping predictions within a single image across the entire batch. Mathematically, the loss term can be written as:

$$L_{\text{IoU}} = \sum_{\text{image} \in \text{batch}} \sum_{i \neq j} IoU(\text{Box}_i, \text{Box}_j)$$

This term is intended to help optimize the locations and sizes of predicted bounding boxes such that overlapping occurrences could be reduced. Intuitive as it is, this innovative loss term does not perform well in our training experiments, presumably due to the PyTorch computation graph being confused by the its sophisticated gradients with respect to the CNN model parameters.

5.6. Github Repository

All our code are checked to the following repository, where the commit and edit history can be found:

`https://github.com/Yuanchu/YOLO3D`

Acknowledgements

We would like to thank Professor Iddo Drori and Chenqin for their guidance and constructive feedbacks throughout this project as well as for providing us with the opportunity to conduct and present this research.

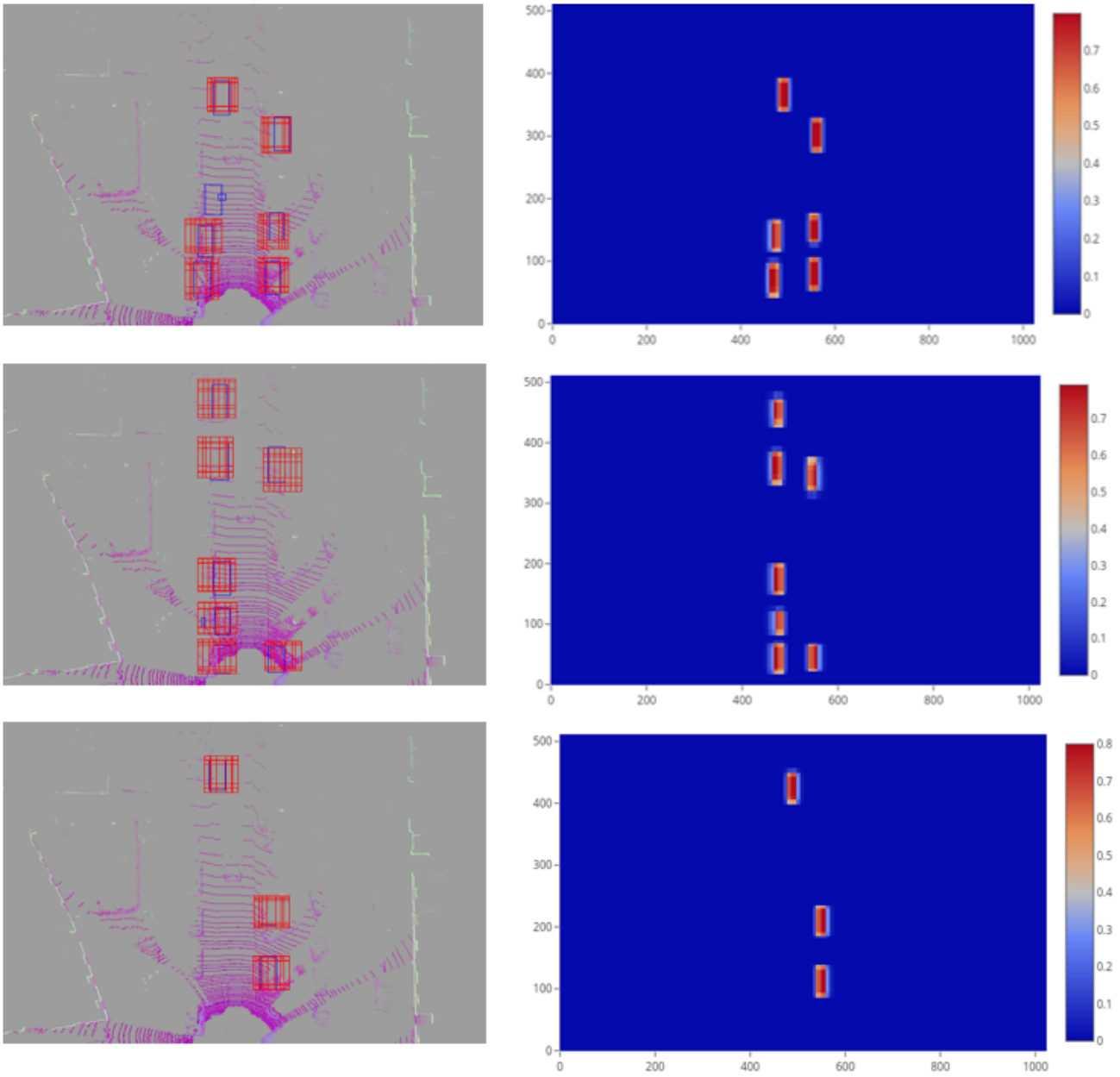

Table 1. Confidence heat maps using our model. The left column shows point-cloud images with the model's predictions, and the right column shows the corresponding heat maps.

References

[1] A. Bhattacharyya, M. Fritz, and B. Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. 2018.

[2] Gal. What my deep model doesn't know... 2015.

[3] Y. Gal. Uncertainty in deep learning. 2016.

[4] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Insights and applications. 2015.

[5] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*, 2015.

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

[7] J. Hui. map (mean average precision) for object detection. 2018.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] S. Lazebnik. Object recognition: History and overview. Lecture Notes for COMP 776: Computer Vision. The University of North Carolina at Chapel Hill.

[10] P. McClure and N. Kriegeskorte. Robustly representing uncertainty through sampling in deep neural networks. 2018.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[12] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[13] L. G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[14] M. Simon, S. Milz, K. Amende, and H.-M. Gross. Complex-yolo: Real-time 3d object detection on point clouds. *arXiv preprint arXiv:1803.06199*, 2018.

[15] Yumi. Measure the uncertainty in deep learning models using dropout. 2018.