StarNet: Targeted Computation for Object Detection in Point Clouds

Jiquan Ngiam[*†], **Benjamin Caine**[*†], **Wei Han**[†], **Brandon Yang**[†],
Yuning Chai[‡], **Pei Sun**[‡], **Yin Zhou**[‡], **Xi Yi**, **Ouais Alsharif**[‡], **Patrick Nguyen**,
Zhifeng Chen[†], **Jonathon Shlens**[*†], **Vijay Vasudevan**[†]
[†]Google Brain, [‡]Waymo
{weihan,shlens}@google.com

Abstract

We present an object detection system designed for point cloud data that enables targeted and adaptive computation. We observe that objects in point clouds are quite distinct from traditional camera images: objects are sparse and vary widely in location, but do not exhibit scale distortions observed in single camera perspective. These two observations suggest that simple and cheap data-driven object proposals to maximize spatial coverage or match the observed densities of point cloud data may suffice. This recognition paired with a local point cloud-based network permits building an object detector that can adapt to different computational settings and target spatial regions. We demonstrate this flexibility and the targeted detection strategies it enables on the large-scale Waymo Open Dataset.

1 Introduction

Detecting and localizing objects forms a critical component of any autonomous driving platform [1, 2]. Self-driving cars (SDC) are equipped with a variety of sensors such as cameras, LiDARs, and radars [3, 4], where LiDAR is one of the most critical as it natively provides high resolution, accurate 3D data about the environment. However, object detection systems for LiDAR look remarkably similar to systems designed for generic camera imagery.

Despite large modality and task-specific differences, the best performing methods for 3D object detection re-purpose camera-based detection architectures. Several methods apply convolutions to discretized representations of point clouds in the form of a projected Birds Eye View (BEV) image [5, 6, 7, 8], or a 3D voxel grid [9, 10]. Alternatively, methods that operate directly on point clouds have re-purposed two stage object detector design [11, 12, 13].

We start by recognizing that 3D region proposals are fundamentally distinct. Every reflected point (above the road) must belong to an object or surface. We demonstrate that efficient sampling schemes that match the data distribution are sufficient for generating region proposals. Then, each proposed region is process independently. To avoid discretization, we featurize each local point cloud directly [14, 15] in order to classify objects and regress bounding box locations [16, 17].

The resulting detector is as accurate as the state-of-the-art at lower inference costs, and more accurate at similar inference costs. The model does not waste computation on empty regions because the proposal method naturally exploits the sparse distribution of the point cloud. One can also dynamically vary the number of proposals and the number of points per proposal at inference time since the model operates locally. This allows the cost of inference to scale linearly, and permits a single trained model to operate at different computational budgets. Finally, because each region is completely independent, one may select at run-time where to allocate region proposals based on context.

Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

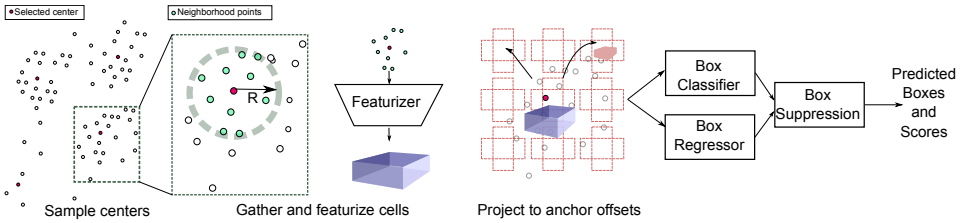


Figure 1: **StarNet overview**. After obtaining a proposal location, we featurize the local point cloud around the proposal. We randomly select K points within a radius of R meters of each proposal center. In our experiments, K is typically between 32 to 1024, and R is 2-3 meters. All local points are re-centered to an origin for each proposal.

For example, a deployed system could exploit priors (e.g. HD maps or temporal information) to target where in the scene to run the detector.

2 Related work

Object detection in point clouds has started with porting ideas from the image-based object detection literature. By voxelizing a point cloud (i.e. identifying a grid location for individual points $\vec{x}_i$) into a series of stacked image slices describing occupancy, one may employ CNN techniques for object detection on the resulting voxel grids [5, 9, 10, 6, 7].

Employing a grid representation for point clouds in object detection presents potential drawbacks. Even when ignoring the height dimension in 3D by using a Birds Eye View Representation (BEV), convolutions can be expensive and the computational demand grows as roughly as $O(hw)$ where h and w are the height and width of an image. In practice, this constraint requires that CNNs operate at no larger than ~ 1000 pixel input resolution [18]. Given the large spatial range of LiDAR, selecting a grid resolution to achieve this pixel resolution (e.g. $0.16 \sim 0.33$ meter/pixel [8]) discards detailed spatial information. This often results in systematically worse performance on smaller objects such as pedestrians [5, 9, 10, 7] where the latter may only occupy several pixels in a voxelized image.

For these reasons, many authors explored building detection systems that operate directly on representations of the point cloud data. For instance, VoxelNet partitions 3-D space and encodes LiDAR points within each partition with a point cloud featurization [9]. The result is a fixed-size feature map, on which a conventional CNN-based object detection architecture may be applied. Likewise, PointPillars [8] proposes an object detector that employs a point cloud featurization, providing input into a grid-based feature representation for use in a feature pyramid network [19]; the resulting per-pillar features are combined with anchors for every pillar to perform joint classification and regression. The resulting network achieves a high level of predictive performance with minimal computational cost on small scenes, but its fixed grid increases in cost notably on larger scenes and cannot adapt to each scene's unique data distribution.

In the vein of two stage detection systems, PointRCNN [12] employs a point cloud featurizer [15] to make proposals via a per-point segmentation network into foreground and background. Subsequently, a second stage operates on cropped featurizations to perform classification and localization.

Finally, other works propose bounding boxes through a computationally intensive, learned proposal system operating on paired camera images [11, 13], with the goal of improving predictive performance by leveraging a camera image to seed a proposal system to maximize recall.

3 Methods

Our goal is to construct a detector that takes advantage of the sparsity of LiDAR data and allows us to target where to spend computation. We propose a sparse targeted object detector, termed *StarNet*: From a sparse sampling of locations (centers) in the point cloud, the model extracts a local subset of neighboring points. The model featurizes the point cloud [14], classifies the region, and regresses bounding box parameters. Importantly, the object location is predicted *relative* to the selected location. Each spatial location may be processed by the detector completely independently. An overview of this method is depicted in Figure 1 and Appendix C.

The structure of the proposed system confers two advantages. First, inference on each cell proposal occurs completely independently, enabling computation of each center location to be parallelized.

Second, heuristics or side information [20, 7] may be used to rank the locations to process. This permits the system to focus it's computation budget on the most important locations.

3.1 Center location selection

We propose using an inexpensive, data-dependent algorithm to generate proposals from LiDAR with high recall. In contrast to prior work [5, 8, 10], we do not base proposals on fixed grid locations, but instead generate proposals to respect the observed data distribution in a scene.

Concretely, we sample N points from the point cloud, and use their (x, y) coordinates as proposals. In this work, we explore two sampling algorithms: random uniform sampling, and farthest point sampling (FPS), which are compared in Appendix D and visualized in Appendix F. Random uniform sampling provides a simple and effective baseline because the sampling method biases towards densely populated regions of space. In contrast, farthest point sampling (FPS) selects individual points sequentially such that the next point selected is maximally far away from all previous points selected, maximizing the spatial coverage across the point cloud. This approach permits varying the number of proposals from a small, sparse set to a large, dense set that covers point cloud space.

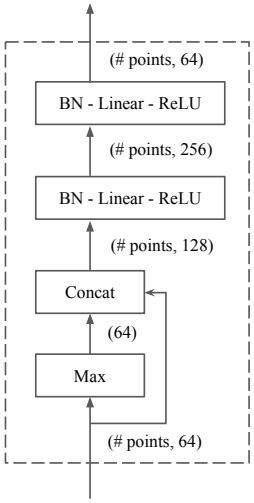


Figure 2: **StarNet Block**. We annotate edges with tensor dimensions for clarity: (# points, feature dimension) represents a point cloud with # points with attached features.

3.2 Featurizing local point clouds

We experimented with several architectures for featurizations of native point cloud data [15, 21] but the StarNet featurizer most closely followed [22]. The resulting architecture is agnostic to the number of points provided as input [15, 21, 22].

StarNet blocks (Figure 2) take as input a set of points, where each point has an associated feature vector. Each block first computes aggregate statistics (max) across the point cloud. Next, the global statistics are concatenated back to each point's feature. Finally, two fully-connected layers are applied, each composed of batch normalization (BN), linear projection, and ReLU activation. StarNet Blocks are stacked to form a featurizer (Figure 8).

3.3 Anchor Boxes.

We use a grid of $G \times G$ total anchor offsets relative to each cell center, and each offset can employ different rotations or anchor dimension priors. We project each featurized cell to D dimensional feature vectors at each location offset from which we predict classification logits and bounding box regression logits following [10, 8]. Ground truth labels are assigned to individual anchors based on their intersection-over-union (IoU) overlap [10, 8]. To make final predictions, we employ an oriented, 3-D multi-class version of non-maximal suppression (NMS) [23].

4 Results

In this work, we focus on 3D vehicle and pedestrian detection from LiDAR point cloud data. All experiments in this section were performed on the Waymo Open Dataset, but we provide KITTI [1] results in Appendix A, and ablations regarding the importance of data augmentation on KITTI in Appendix B. Additionally, a comparison of sampling techniques can be found in Appendix D.

4.1 Flexible detection on a large-scale self-driving dataset

To demonstrate StarNet's flexible design, we show two strategies for altering a single trained model's computational cost at inference time: varying the number of proposals, and because the point featurizer is agnostic to the number of points, varying the number of points used as input to our featurizer

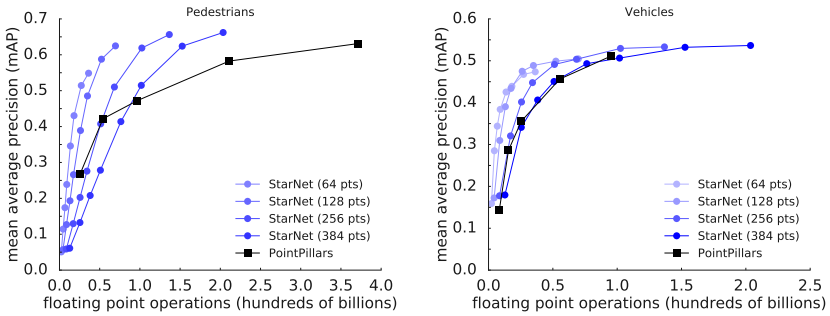


Figure 3: **Flexible computational cost of detection** for (left) pedestrians and (right) vehicles. Across 5 separately-trained PointPillars models [8], computational cost grows quadratically with increased spatial resolution for the LiDAR pseudo-image. All curves for StarNet arise from a *single* set of saved model weights. Each curve traces out StarNet accuracy on the *Validation set* for a fixed number of point cloud points. Points along on a single curve indicate 64 to 1024 selected centers.

for each proposed region. Each blue curve in Figure 3 shows the computational cost versus predictive performance of the same model evaluated with a different number of input points per proposal, while varying the number of proposals. We compare our single StarNet model to a family of baseline PointPillars models (black curve) trained at 5 different grid resolutions.[1][2] Validation accuracy was computed on all annotated bounding boxes in the Waymo Open Dataset with 5+ LiDAR points.

We observe in Figure 3 that a single StarNet model can outperform our baseline in both mean average precision and FLOPs, and in the case of pedestrians offers a more dramatic improvement. It is also evident that PointPillars (black curve) increases in precision for both pedestrians and vehicles with increased grid resolution, but with a computational cost that grows quadratically.

4.2 Targeted computation for point cloud detection

Our approach enables proposals to be ranked by importance, which can be leveraged as a mechanism for targeting computation. We demonstrate this by first ranking each proposal by distance from the vehicle. Next, we select a subset K = $128 - 1024$ proposals from N = 1024 proposals, and evaluate our mean average precision (mAP) in 10 meter bins from the vehicle (Figure 6). We find that the same model with one eighth the proposals and computational cost achieves the same mAP up to 10 meters. Likewise, the same model with fewer than half the proposals and computational cost achieves the same mAP as the default model up to 20 meters.

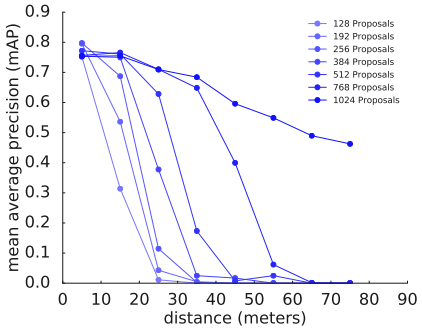


Figure 4: **Spatially targeting detection**. We selected 128-1024 spatially-closest proposals and evaluate the mAP in 10m distance bins.

5 Discussion

In this work, we presented a 3D object detection system that operates on native point cloud data. Our approach leverages the sparse distribution of point cloud data, and allows for flexible targeting across a range of computational priorities. We find that the system is competitive with state-of-the-art systems on the large-scale Waymo Open Dataset. We demonstrate how in principle the detection system may be employed to target spatial locations without retraining nor sacrificing the prediction quality. For instance, depending on evaluation settings, a single trained pedestrian model may exceed the predictive performance of a baseline convolutional model by $\sim 48\%$ at a similar computational demand; or, the same model may achieve the same predictive performance but with $\sim 20\%$ of the computational FLOPS cost.

[1] We use resolutions of $[128, 192, 256, 384, 512]$ pixels, with 16k to 32k featurized locations (pillars).

[2] Following [8], vehicle models have an output stride of 2, and pedestrian models an output stride of 1

References

[1] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[3] Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843. IEEE, 2014.

[4] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[5] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[6] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[7] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting HD maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155, 2018.

[8] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *arXiv preprint arXiv:1812.05784*, 2018.

[9] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[10] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[11] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Ipod: Intensive point-based object detector for point cloud. *arXiv preprint arXiv:1812.05276*, 2018.

[12] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[13] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[15] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[16] 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*, pages 91–99, 2015.

[17] 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*, pages 91–99, 2015.

[18] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[20] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[21] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. *arXiv preprint arXiv:1811.07246*, 2018.

[22] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[23] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

Targeted Computation for Object Detection in Point Clouds: Supplementary Material

A KITTI Results

We evaluate StarNet on the popular self-driving benchmark KITTI [1]. Results here arise from two separately trained models, one for vehicles, and another for pedestrians and cyclists, similar to [8]. Our StarNet model uses 256 farthest point sampling centers (proposals) during evaluation, and a series of 5 StarNet blocks. Full training details are available in already open sourced code available at **Anonymized**.

We demonstrate that our performance is comparable on the KITTI test set to several competitive LiDAR only baselines. Because KITTI is relatively small compared to other deep learning datasets, data augmentation has been used extensively for all competing methods. We note how gains in predictive performance due to data augmentation (up to +18.0, +16.9 and +30.5 mAP on car, pedestrian and cyclist) are substantially larger than gains in performance observed across advances in detection architectures, and provide a data augmentation ablation for KITTI below in Appendix B. As such, we focus the majority of the above paper on the much larger Waymo Open Dataset.

Table 1: **Results on the KITTI *test* object detection benchmark** for object detection systems using 3-D evaluation. All detection results and comparisons based only on LIDAR data. mAP calculated with an IOU of 0.7, 0.5 and 0.5 for vehicles, cyclists and pedestrians, respectively.

	Car			Pedestrian			Cyclist		
Model Name	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
VoxelNet [9]	77.47	65.11	57.73	39.48	33.69	31.5	61.22	48.36	44.37
SECOND [10]	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
PointPillars [8]	79.05	74.99	68.30	52.08	43.53	41.49	75.78	59.07	52.92
StarNet	81.63	73.99	67.07	48.58	41.25	39.66	73.14	58.29	52.58

B Data Augmentation on KITTI

Table 2: **Data augmentation improves performance substantially on KITTI.** All results reported on 3D detection metric for *validation* data. Global data augmentations consist of randomly flipping about y-axis, uniformly randomly rotating all points $\pm 45^\circ$, uniformly randomly scaling all points $[0.95, 1.05]$ and randomly translating individual points along the z-axis with std of 0.35. Bounding box augmentations consist of augmenting the bounding boxes from a ground truth data base (< 25 boxes per scene) [10] and uniformly randomly rotating boxes by $\pm 9^\circ$.

	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
no augmentations	77.8	68.2	65.7	45.5	40.8	36.9	56.1	36.9	35.5
+ global augment	85.4	74.8	72.7	46.3	39.4	36.2	75.8	51.9	48.7
+ bbox augment	86.6	77.0	74.6	63.9	56.5	52.4	81.5	58.3	54.8
+ both	90.4	79.7	77.7	71.8	61.0	53.8	86.5	65.2	66.0

We employed the following data augmentations culled from a survey of the recent literature [6, 5, 7, 9, 10, 8]. per-bounding box rotation ($-\frac{\pi}{20}$, $\frac{\pi}{20}$), y-axis scene flipping, world coordinate scaling (0.95, 1.05), global rotation ($-\frac{\pi}{4}$, $\frac{\pi}{4}$), and ground-truth bounding box augmentation (copy-pasting objects from different scenes).

In order to parse the relative benefit of various data augmentations strategies to the overall performance of the model, we selectively removed data augmentations before training the model and report the corresponding results in Table 2. We find that the addition of data augmentations for the bounding box labels as well as augmentations for global alterations of the point clouds substantially improved the detection performance. Furthermore, both sets of augmentations are additive in terms of improving predictive performance. In particular, we note that some of the gains in predictive performance (up to +30.5 mAP on cyclist) are substantially larger than the gains in performance observed across advances in detection architectures [10, 8].

C Architecture and Training of StarNet

C.1 Training Details

We briefly describe our training procedure for our Waymo Open Dataset models, but encourage readers to reference our already open sourced code at **Anonymized**. All models are trained with the Adam [24] optimizer with a learning rate of 0.001, and a exponential decay to 0 over 75 epochs (with decay starting at epoch 5). We use gradient clipping with a value of 5, a L2 regularization loss weight of 3e-5 for vehicles, and 3e-6 for pedestrians. We employ a batch size of 32 during training.

Both vehicle and pedestrian StarNet models use a featurizer architecture consisting of 5 StarNet blocks, each with an input dimensionality of 128, a hidden dimensionality (the first fully connected layer) of 256, and an output dimensionality of 64. After concatenation of each blocks global features, the output dimensionality of the featurizer is 384. We calculate the mean of the classes dimensions (height, width, and length) from the training set, and use 4 rotations ($[0, \frac{\pi}{2}, \frac{3\pi}{4}, \frac{\pi}{4}]$) for our box prior. This box prior is placed in a 5x5 equispaced grid around each sampled center, giving us 100 anchors per sampled region. For each sampled region, we project the features from our StarNet featurizer to 128 features for each anchor offset. We then use a linear layer to classify the class of each anchor with a focal sigmoid loss ($\alpha = 0.25, \gamma = 2$), and another linear layer to regress the 7 parameters of our box. Rotation is encoded as $sin(\theta)$.

Finally, we apply a 3D oriented, per-class non-maximum suppression (NMS) to our predictions, keeping the top 512 boxes. We found for vehicles the best NMS parameters, while not overly sensitive, were a 3D IoU threshold of 0.03, and an NMS score threshold of 0.31. For pedestrians, we found a higher 3D IoU threshold ideal, with a value of 0.46, and a score threshold of 0.01. Again, the model was not overly sensitive to these values, beyond the discrepancy of IoU threshold between classes.

C.2 Ground removal from point clouds

One important feature for the efficiency of the proposal system is to remove the points associated with ground reflections. We follow previous work and remove points with a z-dimension below a certain threshold [10, 8], although more sophisticated methods are possible [20]. For KITTI, $z > -1.35$. For the Waymo Open Dataset, we computed the 10^{th} and 90^{th} percentile of the center z location of all objects and kept only points with z coordinate in that range. In spite of this heuristic, FPS still covers many ground points; high-quality ground removal would decrease the number of centers required.

C.3 Constructing final predictions

The current design uses a grid of $G \times G$ total anchor offsets relative to each cell center, and each offset can employ different rotations or anchor dimension priors. We emphasize that unlike single-stage detectors [8, 10], the anchors are data-dependent since they are based on the proposals.

We project each featurized cell to $G \times G$ D-dimensional feature vectors; the same projection weights are shared for all cells, but different weights are used for each grid offset. These feature vectors are then used as input to classification and regression heads that produce classification logits and bounding box regression logits. The regression logits predict $\delta x, \delta y, \delta z$ corresponding to residuals of the location of the anchor bounding box; $\delta h, \delta w, \delta l$ corresponding to residuals on height, width and length; and a residual on the heading orientation $\delta\theta$, parameterized following [10, 8]. We use a smoothed-L1 loss on each predicted variate [8]. The classification logits are trained with a focal cross-entropy loss on the class label [25].

Algorithm 1: StarNet

Input : $N, K, R, G, X = \{\vec{x}_i\}, F = \{\vec{f}_i\}, A = \{anchors\}$, where $|A| = G^2$
Output: Coordinates and scores of boxes B
// Samples N centers from X.
$C \longleftarrow Sample(X, N)$
$B \longleftarrow \emptyset$
for $i \in C$ **do**
  // Samples K points near $\vec{x}_i$.
  $P = Neighbor(\vec{x}_i, K, R)$
  // Featurizes points around $\vec{x}_i$.
  $\vec{u} = Featurize(\{\vec{x}_j | j \in P\}, \{\vec{f}_j | j \in P\})$
  // Predicts boxes and their scores, given
  // feature $\vec{u}$ and anchors A.
  $B' = PredictBoxes(\vec{u}, A)$
  $B \longleftarrow B \cup B'$
end
$B \longleftarrow NonMaximumSuppression(B)$

Ground truth labels are assigned to individual anchors based on their intersection-over-union (IoU) overlap [10, 8]. We compute the intersection-over-union (IoU) for each anchor and ground truth label and assign labels to foreground or background if the IoU exceeds or falls short of prescribed foreground an background thresholds (E.g., the pedestrian model for Waymo Open Dataset used 0.6 and 0.45 as foreground and background IoU threshold). Otherwise, objects are ignored. We also perform force-matching if an object is not assigned as foreground to any anchor: we assign the object as foreground to the highest matching anchor if (a) the highest

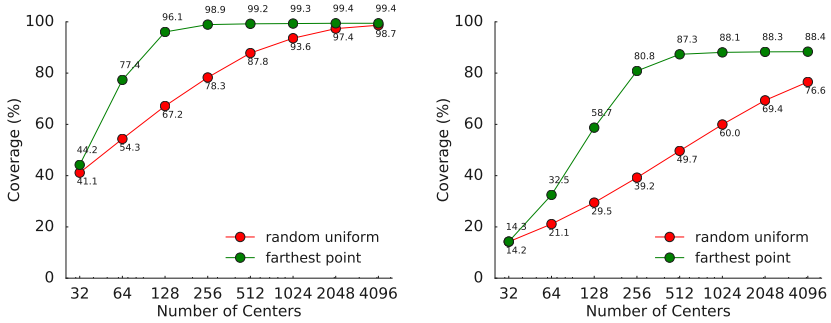


Figure 5: **Simple sampling procedures have good coverage over ground truth bounding boxes.** The coverage of proposals for cars is plotted against the number of samples on KITTI (left) and Waymo Open Dataset (right). Error bars (not shown) range from 0.5%-3.0%. See text for details.

matching anchor is not assigned to foreground of any object and (b) the IoU to the matching anchor is greater then zero.

To make final predictions, we employ an oriented, multi-class version of non-maximal suppression (NMS) [23]. NMS removes predictions of the same class that heavily overlap with one another. Through preliminary experiments, we found that setting the IoU threshold to 0.46 resulted in the best predicted results for the pedestrian model for Waymo Open Dataset. Note that NMS is not used during training resulting in potentially faster training times than other detection approaches [16, 17].

D Sampling strategies for point cloud detections

As a first step in constructing the object detection system, we explored two simple strategies for naively sampling point clouds: random sampling and farthest point sampling (Section 3.1). Figure 7 in Appendix F showcases typical results of what each proposal mechanism generates on scenes from the Waymo Open Dataset. Note that random sampling samples many centers in dense locations, whereas farthest point sampling maximizes spatial coverage of the scene.

To quantify the efficacy of each proposal method, we measure the coverage as a function of the number of proposals. Coverage is defined as the fraction of annotated objects with 5+ points that have IoU > 0.5 with the sampled boxes. Figure 5 plots the coverage for each method for a fixed IOU of 0.5 for cars in KITTI [1] and the Waymo Open Dataset. All methods achieve monotonically higher coverage with greater number of proposals with coverage on KITTI exceeding 98% within 256 samples. Because random sampling is heavily biased to regions which contain many points, there is a tendency to oversample large objects and undersample regions containing few points. Farthest point sampling (FPS) uniformly distributes samples across the spatial extent of the point cloud data. We observe that FPS provides uniformly better coverage across a fixed number of proposals and we employ this technique for the rest of the work. Finally, in Figure 6 we show how this relationship translates to mean average precision (mAP) for a single StarNet Pedestrian model evaluated with both sampling methods.

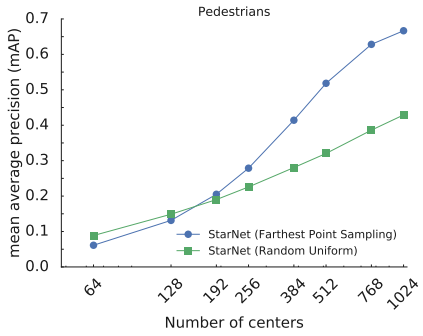


Figure 6: **Adaptive computation with a single trained model**. A Pedestrian StarNet, trained with 1024 proposals, evaluated on Waymo Open Dataset using 64 to 1024 proposals from either farthest point or random uniform sampling.

E Details of Waymo Open Dataset

The Waymo Open Dataset contains both LiDAR points and camera images – although this work focuses on the former. The dataset was collected in both urban and suburban areas, in a variety of weather conditions, and across varying times of the day. The LiDAR spans a circular volume centered about the self-driving car with a radius of 75m. The dataset is split into training, validation. The training split consists of 158,361 frames across 798 distinct scenes. The validation split consists of 40,077 frames across 202 scenes. Each scene consists of about 200 frames recorded at 10 Hz.

The dataset was annotated exhaustively for vehicles and pedestrians. Objects were human-labeled based on camera and LiDAR images. Thus, many annotated objects are minimally observable based on the LiDAR and contain very few reflected points. The training split contains approximately 2.2M pedestrians and 4.8M vehicles. The validation split contains 539k pedestrians and 1.25M vehicles. Vehicles were annotated across a large array of sizes from mopeds to large trucks. Pedestrians were annotated even in crowded environments indicating that some pedestrian boxes may overlap and lie within closer proximity.

F Visualizing Sampling Methods

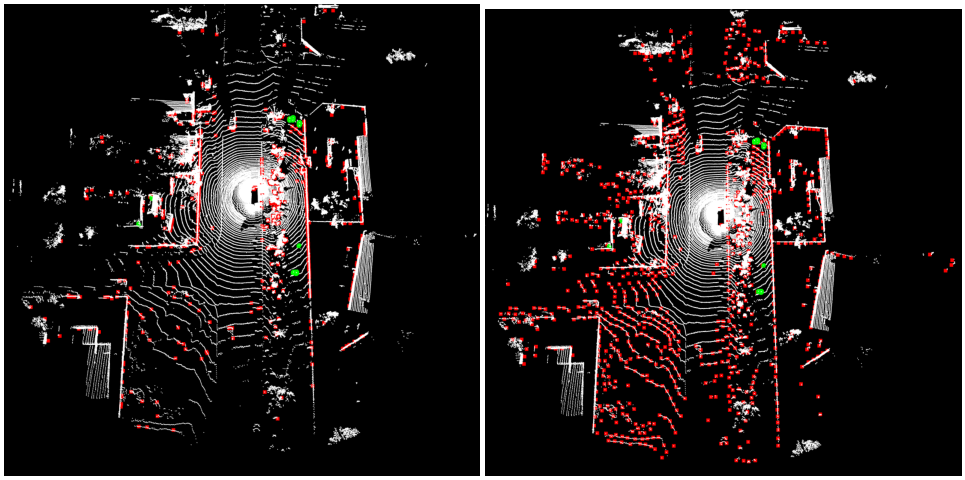
[h] 

Figure 7: Example of random uniform sampling (left) and farthest point sampling (right) with the same number of samples. Red indicate selected centers. Green indicate pedestrians. Note that random uniform sampling biases towards high density regions, while farthest point sampling evenly covers the space. Neither place any proposals in empty space.

G StarNet Featurizer

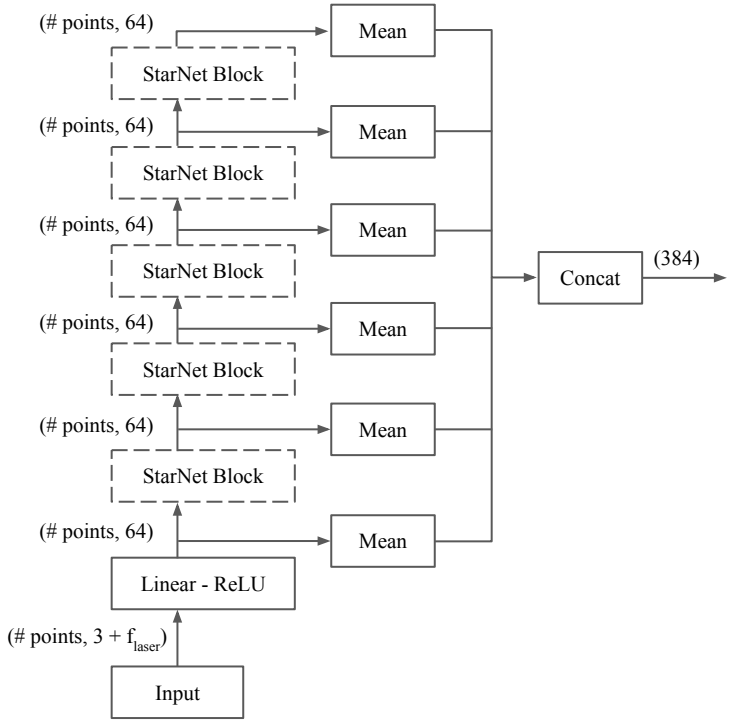


Figure 8: We stack multiple StarNet blocks to build the full network for featurization. There are readout connections from each block to the final feature representation.