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
{jngiam,bencaine}@google.com

Abstract

Detecting objects from LiDAR point clouds is an important component of self-driving car technology as LiDAR provides high resolution spatial information. Previous work on point-cloud 3D object detection has re-purposed convolutional approaches from traditional camera imagery. In this work, we present an object detection system called StarNet designed specifically to take advantage of the sparse and 3D nature of point cloud data. StarNet is entirely point-based, uses no global information, has data dependent anchors, and uses sampling instead of learned region proposals. We demonstrate how this design leads to competitive or superior performance on the large Waymo Open Dataset [1] and the KITTI [13] detection dataset, as compared to convolutional baselines. In particular, we show how our detector can outperform a competitive baseline on Pedestrian detection on the Waymo Open Dataset by more than 7 absolute mAP while being more computationally efficient. We show how our redesign—namely using only local information and using sampling instead of learned proposals—leads to a significantly more flexible and adaptable system: we demonstrate how we can vary the computational cost of a single trained StarNet without retraining, and how we can target proposals towards areas of interest with priors and heuristics. Finally, we show how our design allows for incorporating temporal context by using detections from previous frames to target computation of the detector, which leads to further improvements in performance without additional computational cost.

1. Introduction

Detecting and localizing objects forms a critical component of any autonomous driving platform [13, 6]. As a result, self-driving cars (SDC) are equipped with a variety of sensors such as cameras, LiDARs, and radars [8, 39], which the perception system must use to create an accurate 3D representation of the world. Due to the nature of the driving task, the perception system must operate in real-time and in a highly variable operating environment [19]. LiDAR is one of the most critical sensors as it natively provides high resolution, accurate 3D data about the environment. However, LiDAR based object detection systems for SDCs look remarkably similar to systems designed for generic camera imagery.

Object detection research has matured for camera images with systems evolving to solve camera-specific challenges such as multiple overlapping objects, large intra-class scale variance due to camera perspective, and object occlusion [15, 14, 35, 25, 26]. These modality-specific challenges make the task of localizing and classifying objects in imagery uniquely difficult, as an object may occupy any pixel, and neighboring objects may be as close as one pixel apart. This necessitates treating every location and scale in the image equally, which naturally aligns with the use of convolutional networks for feature extraction [15, 14]. While convolutional operations have been heavily optimized for parallelized hardware architectures, scaling these methods to high resolution images is difficult as computational cost scales quadratically with image resolution.

In contrast, LiDAR is naturally sparse; 3D objects have real world scale with no perspective distortions, and rarely overlap. Additionally, in SDC perception, every location in the scene is not equally important [50, 4, 2], and that importance can change dynamically based on the local environment and context. 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 [47, 28, 46, 22], or a 3D voxel grid [53, 45]. Alternatively, methods that operate directly on point clouds have re-purposed two stage object detector design, replacing feature extraction operations but still adopting the same camera-inspired region proposal stage [48, 38, 31].

In this paper, we revisit the design of object detection systems in the context of 3D LiDAR data, and propose a new framework which better matches the data modality and the demands of SDC perception.

We start by recognizing that 3D region proposals are fundamentally distinct. Every reflected point must belong to an object or surface. In this setting, we demonstrate that efficient sampling schemes on point clouds – with zero learned parameters – are sufficient for generating region proposals. In addition to being computationally inexpensive, sampling has the advantage of implicitly exploiting the sparsity of the data by matching the data distribution of the scene.

Departing from the trend of increasing use of global context, we process each proposed region completely independently. This independence, and non-learned proposal mechanism also allows us to inject priors into the proposal process, which we show the value of by leveraging temporal context in the form of seeding sampling with the previous frames detections. Finally, we entirely avoid any discretization procedure and instead featurize the native point cloud [32, 33] in order to classify objects and regress bounding box locations [35, 36].

The resulting detector is as accurate as the state of the art at lower inference cost, and more accurate at similar inference costs. In addition, these design decisions result in several key benefits. First, the model does not waste computation on empty regions because the proposal method naturally exploits point cloud sparsity. Second, one can dynamically vary the number of proposals and the number of points per proposal at inference time since the model operates locally. This feature permits a single trained model to operate at different computational budgets. Third, the model can easily leverage contextual information (HD maps, temporal context) to target computation. For example, detection outputs from preceding frames can be used to inform the current frame's sampling locations.

In summary, our main contributions are as follows:

- Introduce a flexible, local point-based object detector geared towards SDC perception. In the process we demonstrate that cheap proposals on point clouds, paired with a point-based network, results in a system that is competitive with state-of-the-art performance on self-driving car benchmarks.
- Demonstrate the computational-flexibility of our model through showing how a single model designed in this fashion may adapt its inference cost. For in-

stance, a single trained pedestrian model may exceed the predictive performance of a baseline convolutional model by $\sim 48\%$ at similar computational demand; or, the same model without retraining may achieve similar predictive performance but with $\sim 20\%$ of the computational demand.

- Demonstrate the ability of the model to selectively target specific locations of interest. We show how temporal context (using only the *outputs* of previous frames) can be used with the model to improve detection mAP scores by $\sim 40\%$.

2. Background

2.1. Object detection in images

Early object detection systems consist of two stages: first, to propose candidate detection locations, and next, to discriminate whether a given proposal is an object of interest [12, 10, 37, 40]. The advent of convolutional neural networks (CNN) [21, 24], showed that a CNN-based featurization may provide superior proposals as well as improve the second discriminative stage [15, 36, 14]. Modern CNN-based detection systems maximize prediction performance by densely sampling an image for all possible object locations. This requires a computationally-heavy first stage featurization to provide high quality bounding box proposals [15, 14]. In addition, the second stage of an object detector will need to be run on each proposal within a *single* image. These heavy computational demands are prohibitive in constrained environments (e.g. mobile and edge devices) such that speed versus accuracy trade-offs must be considered [18].

To address these concerns, recent work has focused on *one stage* object detectors that attempt predict bounding box locations and object identity in a single inference operation of a CNN [27, 34]. Although single-stage systems provide faster inference, these systems generally exhibit worse predictive performance than two-stage systems [18]. That said, recent advances in redesigning the loss functions have mitigated this disadvantage significantly [26, 51, 23]. The speed and reduced complexity advantages associated with a one stage model do however come with an associated cost: by basing the inference procedure on convolutions which densely sample an image, the resulting model must treat all spatial locations equally. In the context of self-driving cars, this design decision hampers the ability to adapt computation to the current scene or latency requirements.

2.2. Point cloud featurization

Raw data arising from many types of sensors come in the form of point cloud data (e.g. LIDAR, RGB-D). A point cloud consists of a set of N 3-D points $\{\vec{x}_i\}$ indexed by i which may contain an associated feature vector $\{\vec{f}_i\}$. The
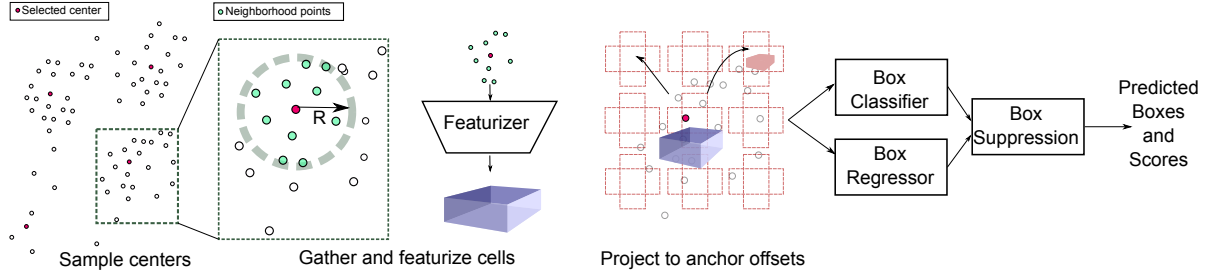
Figure 1: **StarNet overview**.

set of points are unordered and may be of arbitrary size depending on the number of reflections identified by a sensor on a single scan. Ideally, learned representations for point clouds aim to be permutation invariant with respect to i and agnostic to the number of points N in a given example [32, 33]. On-going efforts have attempted to design models that operate directly on point cloud data, some of which are derived to mimic convolutions [41, 49].

2.3. Object detection with point clouds

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 traditional CNN techniques for object detection on the resulting images or voxel grids [47, 53, 45, 28, 46, 52, 29].

VoxelNet partitions 3-D space and encodes LiDAR points within each partition with a point cloud featurization [53]. The result is a fixed-size feature map, on which a conventional CNN-based object detection architecture may be applied. Likewise, PointPillars [22] 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 [25]; 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.

LaserNet [29] opts to work on the Range Image representation of the LiDAR instead of a point cloud or a voxelized view. The range image data takes on a dense perspective view that LaserNet applies convolutions to. While this method has the advantage of working on a dense representation, it also faces challenges (similar to camera images) of having a perspective effect on the scale objects. Objects in a range image may have a large variance in scale.

In the vein of two stage detection systems, PointRCNN [38] employs a point cloud featurizer [33] to make proposals via an expensive per-point segmentation network into foreground and background. Subsequently, a second stage operates on cropped featurizations to perform the final classification and localization. Other works propose bounding boxes through a computationally intensive, learned proposal system operating on paired camera images [48, 31], 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 better aligns with the requirements of a SDC perception system, taking advantage of the sparsity of the data, allowing us to target where to spend computation, and operating on the native data. To address these goals, we propose a sparse targeted object detector, termed *StarNet* (Figure 1): Given a sparse sampling of locations in the point cloud, the model extracts a small (random) subset of neighboring points. The model featurizes the point cloud [32], classifies the region, and regresses bounding box parameters. The object location is predicted *relative* to the selected location and only uses local information. This setup ensures that each spatial location may be processed by the detector independently.

The structure of the proposed system confers two advantages. First, inference on each cell proposal occurs completely independently, enabling computation of each location to be parallelized to decrease inference latency. Second, contextual information information [4, 46] may be used to inform importance of each proposal.

The rest of this section describes the architecture of StarNet in more detail.

3.1. Center selection

We propose using an inexpensive, data-dependent algorithm to generate proposals from LiDAR with high recall. In contrast to prior work [47, 22, 45], 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. To avoid
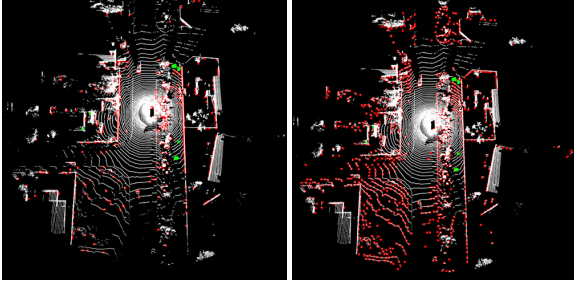
Figure 2: Example of random uniform sampling (left) and farthest point sampling (right) with the *same number of samples*. Red indicate selected centers. Green indicate pedestrians. Note that random uniform sampling biases towards high density regions, while farthest point sampling evenly covers the space. Neither place any proposals in empty space.

sampling regions on the ground, we follow previous works [45, 22] and only allow sampling of points between a certain z-dimension range. For KITTI [13] this is $z \in [-1.35, \inf]$, and for the Waymo Open Dataset [1] we calculate the 10th and 90th percentile of the center z location of all objects. Note that these points are only excluded for sampling, and will be present in later stages.

In this work, we explore three sampling algorithms: random uniform sampling, farthest point sampling (FPS), and a hybrid approach of seeding FPS with preceding frame detections (Figure 2, Section 4.4). 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. Finally, in Section 4.4, we show how to leverage the previous frame detection outputs as seed locations for FPS. We show that this is a light-weight and effective way to leverage temporal information.

3.2. Featurizing local point clouds

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. LiDAR features associated with each point are also used as part of the input. We experimented with several architectures for featurizations of native point cloud data [33, 42] but most closely followed [44]. The resulting architecture is agnostic to the number of points provided as input [33, 42, 44].

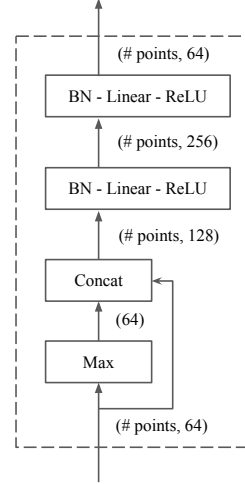StarNet blocks (Figure 3) take as input a set of points,

Figure 3: **StarNet Block**. We annotate edges with tensor dimensions for clarity: (# points, 64) represents a point cloud with # points, where each point has an associated 64-dimensional feature.
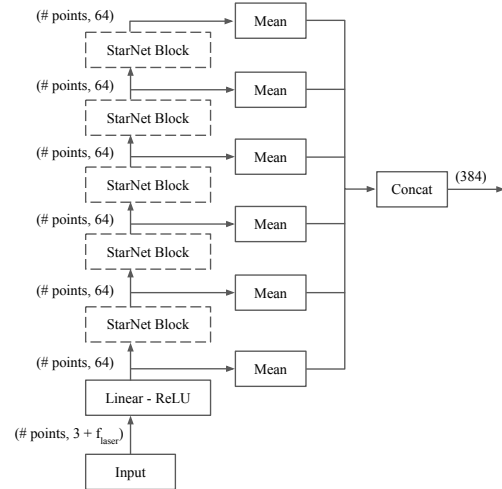

Figure 4: **StarNet point cloud featurizer.** StarNet blocks are stacked, where each block's output is read out using mean aggregation. The readouts are concatenated together to form the featurization for the point cloud.

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 5-layer featurizer (Figure 4) that outputs a 384-dimensional feature.

The StarNet point featurizer (Figure 4) stacks multiple StarNet blocks, following ideas from graph neural networks [44]. We experimented with different choices network ar-
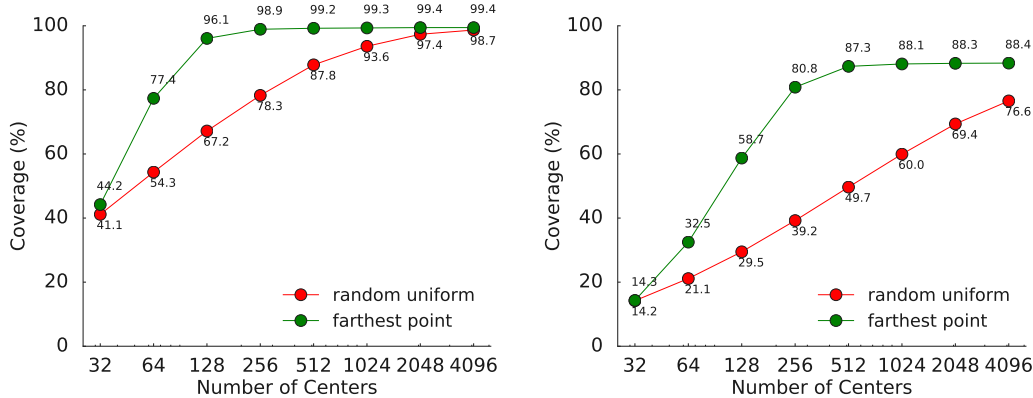
Figure 5: **Simple sampling procedures have good coverage over ground truth bounding boxes.** The coverage of proposals for cars and vehicles is plotted against the number of samples on KITTI (left) and Waymo Open Dataset (right). Error bars (not shown) range from 0.5%-3.0%. See text for details.

chitectures and found that using max aggregation, concatenate combination, and mean readout performed well. By design, the same trained network can be used with varying number of input points, giving it a large degree of flexibility.

3.3. Constructing final predictions from bounding box proposals.

For each cell center, a grid of $G \times G$ total anchor offsets are placed relative to each cell center, where each offset can employ different rotations or anchor dimension priors. We emphasize that unlike single-stage detectors [22, 45], the anchor grid placement is data-dependent since it is based on the proposals.

For each grid offset, we compute a D dimensional feature vector using a learned linear projection from the cell's 384-dimensional feature; each offset has a different projection. The D dimensional feature is shared across the rotations and dimensions at the grid offset. From this feature, we predict classification and regression logits. The bounding box 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 $\delta \theta$.

We use a smoothed-L1 loss on each predicted variate [45, 22, 52]. For the rotation loss, We use a direction invariant loss $\mathrm{SmoothL1}(sine(\delta\theta - \delta\theta_{\mathrm{gt}}))$ for all experiments, except for models where we report heading accuracy weighted average precision (mAPH). For direction aware models, we use $\mathrm{SmoothL1}(WrapAngle(\delta\theta - \delta\theta_{\mathrm{gt}}))$, where WrapAngle ensures the angular difference is between $-\pi$ to π. The classification logits are trained with a focal cross-entropy loss on the class label [26].

Ground truth labels are assigned to anchors based on their intersection-over-union (IoU) overlap [45, 22]. We compute the IoU for each anchor and ground truth label and assign labels to foreground if $IoU > 0.6$ or background if $IoU < 0.45$. Anchors with IoU matches between the two thresholds 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 its highest matching anchor if (a) the highest matching anchor is not assigned to foreground of any object and (b) the IoU with the matching anchor is greater than zero. Final predictions use an oriented, multi-class non-maximal suppression (NMS) [15].
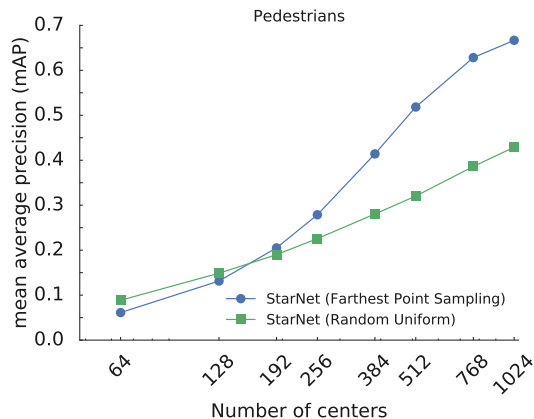

Figure 6: **Adaptive computation with a single trained model**. Waymo Open Dataset *Validation set* mAP on pedestrians of a single StarNet model trained with 1024 proposals, evaluated with 64 to 1024 proposals.

4. Results

We present results on the KITTI object detection benchmark [13] and the Waymo Open Dataset [1]. We train models using the Adam [20] optimizer with an exponentially-decaying learning rate schedule starting at 1e-3 and de-

3D detection	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
VoxelNet [53]	77.47	65.11	57.73	39.48	33.69	31.5	61.22	48.36	44.37
SECOND [45]	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
PointPillars [22]	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

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.

caying over 650 epochs for KITTI, and 75 epochs for the Waymo Open Dataset. We perform some hyper-parameter tuning on the validation set and perform final evaluations on the corresponding test datasets. Full hyperparameters can be found in our already open-sourced code (http://github.com/tensorflow/lingvo).

4.1. Sampling strategies for point cloud detections

We first investigate sampling strategies for center selection, evaluating on KITTI and Waymo Open Dataset. We explore two strategies for naively sampling point clouds: random sampling and farthest point sampling (Section 3.1). We observe 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 our sampled anchor boxes. Figure 5 plots the coverage for each method for a fixed IOU of 0.5 for cars in KITTI [13] and the Waymo Open Dataset [1]. 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 (see Methods). 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.

4.2. KITTI Dataset

When evaluating StarNet on the KITTI dataset, we found that data augmentation important to obtain good performance. We employed standard data augmentations for point clouds and bounding box labels [47, 53, 45, 28, 46, 22]. We found that the gains in predictive performance due to data augmentation (up to +18.0, +16.9 and +30.5 mAP on car, pedestrian and cyclist respectively) were substantially larger than gains in performance observed across advances

in detection architectures. Additionally, we found checkpoint selection to be extremely important due to the small size of the dataset, and submission filtering (e.g. removing detections where the 2D projected height of our 3D bounding box predictions were smaller than 25 pixels so they are not erroneously labeled as false positives) unique challenges to the KITTI benchmark.

We take our best system for 3-D object detection with the same data augmentations and compare the efficacy of this model to previously reported state-of-the-art systems that only operate on point cloud data [53, 45, 22, 46]. Table 1 reports the 3-D detection results on the KITTI *test* server. StarNet provides competitive mAP scores on car, pedestrian and cyclist to other state-of-the-art methods, exceeding subsets of each category strata.

We found that decisions apart from model design play a significant role in KITTI test set performance: this included data augmentation, checkpoint selection, post process filtering, among others. Since we are interested in determining the efficacy of our modeling approach, we focus the majority of our following experiments on the larger Waymo Open Dataset, which is annotated with high quality labels.

4.3. Waymo Open Dataset

We now focus on the performance of StarNet on the Waymo Open Dataset [1], which is substantially larger and exhibits tremendous diversity.

To demonstrate the relative merits of StarNet, we trained models on pedestrians and vehicles and compared the relative performance of each model to a family of baseline models. Data augmentation was not used in these experiments. We employed PointPillars as a baseline model[1], training 5 different grid resolutions of this model for each class and validated accuracy on all annotated bounding boxes with 5+ LiDAR points. Each version employs a different input spatial resolution for the pseudo-image (128, 192, 256, 384, and 512 pixel spatial grids), with 16K to 32K non-zero featurized locations (pillars). In slight deviation from the original PointPillars paper [22], we use an output stride

---

[1] We note that our custom implementation of PointPillars achieves 74.5, 57.1, and 59.0 mAP for for cars, pedestrians, and cyclists, respectively on KITTI validation at moderate difficulty. This is slightly lower than [22].

Model	Directional?	mAP	mAPH
PointPillars* [22]	✓	60.0	47.3
StarNet	–	**70.1**	35.6
StarNet	✓	67.8	**59.9**
StarNet (with temporal context)	–	**72.1**	37.0

(a) Pedestrian Detection

Model	Directional?	mAP	mAPH
PointPillars* [22]	✓	62.2	**61.7**
StarNet	–	**64.7**	45.5
StarNet	✓	61.5	61.0
StarNet (with temporal context)	–	**65.0**	45.6

(b) Vehicle Detection

Table 2: Waymo Open Dataset *Test set* results on the *LEVEL_1* category (≥ 5 points) for StarNet versus a reimplemented PointPillars [22] baseline model. Directional indicates a directional aware heading loss was used. Temporal context models use the top 512 highest confidence detected locations from the *previous* frame as anchor centers, with the remaining 512 centers drawn using Farthest Point Sampling.
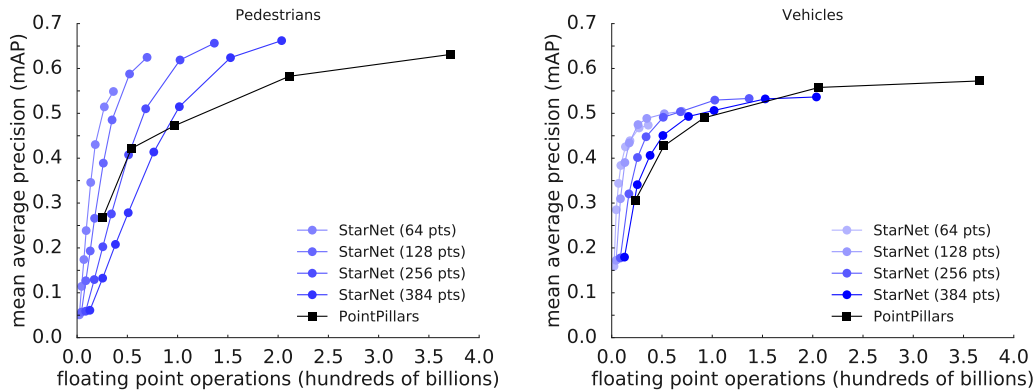

Figure 7: **Flexible computational cost of detection** for (left) pedestrians and (right) vehicles. Across 5 separately-trained PointPillars models [22], computational cost grows quadratically with increased spatial resolution for the LiDAR pseudo-image. All curves for StarNet arise from a *single* set 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.

Model	Pedestrian mAP	Vehicle mAP
PointPillars* [22]	62.1	57.2
Multi View Fusion [52]	65.3	**62.9**
StarNet	**66.8**	53.7

Table 3: Waymo Open Dataset *Validation set* results of LEVEL_1 difficulty for comparison with Multi-View Fusion [52] and a reimplemented PointPillars [22].

of 1 for both vehicles and pedestrian models, as it exhibits substantially higher performance. We hypothesize that a single-stage object detector would exhibit trade-offs in detecting small objects based on the resolution of the image projection. Indeed, we observe in Figure 7 (black points) that higher spatial resolutions achieve higher precision for pedestrians and vehicles, but with a computational cost that grows quadratically.

We also examined the performance of a *single* StarNet model across two strategies for altering computational demand: varying the number of proposals, and varying the number of points supplied to the model per proposed region. Each blue curve in Figure 7 traces out the computational cost versus predictive performance for a given number of points per region, while varying the number of proposals from 64 to 1024. Many of these points lie above the baseline model indicating that StarNet provides favorable performance. In particular, the same pedestrian detection model (e.g., StarNet-128 with 1024 centers) may achieve $\sim 48\%$ relative gain in predictive performance for a similar computational budget as the baseline pillars model ($\sim 100 GFlops$); or, the same model achieves similar predictive performance as the most accurate Pillars model but with $\sim 20\%$ of the computational budget. We emphasize that all StarNet points arise from a *single* trained model, showing how to use a single trained StarNet in a flexible manner through manipulations at inference time.

Finally, we took the highest performing StarNet and PointPillars [22] models from Figure 7, and evaluated them on the Waymo Open Dataset[1] *Test set*. These results are summarized in Table 2, with full numbers including range
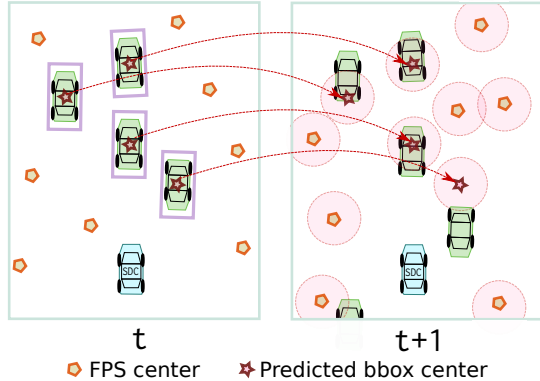
t   t+1

⬠ FPS center   ✹ Predicted bbox center

Figure 8: **Leveraging previous proposals**. Using the highest confidence predicted centers from the previous frames can help improve detection mAP in the next frame.

based breakdowns available in Appendix A. StarNet is competitive on Vehicle detection to our PointPillars [22] baseline, and significantly outperforms it for Pedestrians. If a directional loss is used, we outperform PointPillars by 7.8 mAP and 12.6 mAPH, and 10.1 mAP if a directionless loss is used. Note that forcing the network to learn directionality slightly hinders mAP. Additionally, using temporal context, detailed next, further improves performance. We also compare to Multi-View Fusion [52] in Table 3, showing validation set results as the Multi-View Fusion method does not yet report test set numbers.

4.4. Targeting computation with temporal context

One design benefit of StarNet is the ability to target computation. We now show how using the outputs of the previous time-step can significantly improve mAP over a single frame, while keeping the computation cost unchanged.

Intuitively, high-confidence bounding box proposals output on previous time-steps in 3D are a good prior on the location of objects in the current frame since objects have limited ranges of motion. Hence, one natural approach is to leverage these priors when sampling centers, combining them with random or farthest-point sampling. StarNet permits us to use the locations of the K highest-confidence bounding box predictions from the previous frame quite easily (Figure 8): we can replace the last K farthest-point-sampled (or random) center proposals for the current frame using the pose-corrected locations of the previous top K detection bounding boxes from the prior frame.

We apply this method to Pedestrian detection (Table 4). When using just 32 of the high-confidence predicted bounding box centers from the previous frame and a total of 384 centers, detection mAP on the validation set increases by over 10 absolute mAP, matching the (single frame) performance of sampling a total of 512 centers. When we seed the 384 sampled centers with the top 192 detects from the previous frame, detection mAP improves by nearly 17 absolute

# Previous Frame Detection Centers	# Total Centers	Detection mAP Pedestrians
0	384	41.8
32	384	53.2
192	384	58.0
0	1024	66.8
512	1024	69.7

Table 4: **Previous frame detection centers are good centers to use in the current frame.** StarNet enables using data-dependent centers from the detection outputs of the previous frame to improve detection performance in the current frame. Results reported on the *Validation set*.

mAP, or 40% higher. Surprisingly, when using 1024 total centers, using the best 512 previous detection centers improved mAP by about 3 mAP (2 mAP on the test set), showing that there is room for improvement even when sampling already covers much of the scene.

This experiment demonstrates that using StarNet enables research into smarter and efficient detection and tracking systems. One could employ the use of a tracker to estimate the velocity of detected objects in order to more precisely predict where to "look" in the next frame.

5. Discussion

In this work, we presented a non-convolutional detection system that operates on native point cloud data. The goal of the proposed method is to better match the sparsity of point cloud data, and also allow the system to be flexibly targeted across a range of computational priorities. We demonstrate that the resulting detector is competitive with state-of-the-art detection systems on the KITTI object detection benchmark [13], and can outperform a competitive convolutional baseline on the large-scale Waymo Open Dataset.

The system allows for targeted computation, enabling the use of temporal context from detection outputs of prior frames. We show up to a 40% relative improvement in mAP using prior frames to inform where to target computation for the current frame. We further demonstrate how in principle the detection system can target spatial locations without retraining nor sacrificing the prediction quality. For instance, depending on evaluation settings, a single trained pedestrian model can exceed the predictive performance of a baseline convolutional model by $\sim 48\%$ at a similar FLOPS; or, the same model may achieve the same predictive performance but with $\sim 20\%$ of the FLOPS.

We foresee multiple avenues for further improving the fidelity of the system including: multi-sensor fusion with cameras [48, 31, 30], employing semantic information such as road maps to spatially target detections [46], or restoring global context by removing conditional independence from each proposal [43]. While we have focused this first work on relatively simple sampling methods for proposals, more

expensive or learned methods may further improve the system [35]. For example, one could learn a ranking function to order the relative importance of proposals for a self-driving planning system [9, 5, 7]. Finally, we are particularly interested in studying how this system may be amenable to object tracking [3, 17, 16] as we suspect that because of the design, the computational demands may scale as the *difference* between successive time points as opposed to operating on the entirety of the scene [11].

References

[1] Waymo open dataset: An autonomous driving dataset, 2019. 1, 4, 5, 6, 7

[2] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 1

[3] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016. 9

[4] 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. 1, 3

[5] Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning (ICML-05)*, pages 89–96, 2005. 9

[6] 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. 1

[7] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007. 9

[8] 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. 1

[9] William W Cohen, Robert E Schapire, and Yoram Singer. Learning to order things. In *Advances in Neural Information Processing Systems*, pages 451–457, 1998. 9

[10] Thomas Dean, Mark A Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, accurate detection of 100,000 object classes on a single machine.

[11] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046, 2017. 9

[12] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 2

[13] 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. 1, 4, 5, 6, 8

[14] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 2

[15] 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. 1, 2, 5

[16] Daniel Gordon, Ali Farhadi, and Dieter Fox. Re^3: Real-time recurrent regression networks for visual tracking of generic objects. *IEEE Robotics and Automation Letters*, 3(2):788–795, 2018. 9

[17] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016. 9

[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. 2

[19] Junsung Kim, Hyoseung Kim, Karthik Lakshmanan, and Ragunathan Raj Rajkumar. Parallel scheduling for cyber-physical systems: Analysis and case study on a self-driving car. In *Proceedings of the ACM/IEEE 4th international conference on cyber-physical systems*, pages 31–40. ACM, 2013. 1

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[22] 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. 2, 3, 4, 5, 6, 7, 8, 12

[23] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 2

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015. 2

[25] 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. 1, 3

[26] 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. 1, 2, 5

[27] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[28] 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. 2, 3, 6

[29] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[30] Ming Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 8

[31] 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. 2, 3, 8

[32] 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. 2, 3

[33] 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. 2, 3, 4

[34] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali 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. 2

[35] 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. 1, 2, 9

[36] 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. 2

[37] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2013. 2

[38] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointR-CNN: 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. 2, 3

[39] 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. 1

[40] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. 2

[41] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018. 3

[42] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. *arXiv preprint arXiv:1811.07246*, 2018. 4

[43] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional ShapeContextNet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2018. 8

[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 4

[45] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 3, 4, 5, 6

[46] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting HD maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155, 2018. 2, 3, 6, 8

[47] 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. 2, 3, 6

[48] 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. 2, 3, 8

[49] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017. 3

[50] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 1

[51] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. Bottom-up object detection by grouping extreme and center points. *arXiv preprint arXiv:1901.08043*, 2019. 2

[52] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning (CoRL)*, 2019. 3, 5, 7, 8

[53] 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. 2, 3, 6

Supplementary Material:
Targeted Computation for Object Detection in Point Clouds

A. Full Waymo Open Dataset Results

	Overall	0-30m	30-50m	50m-Inf
PointPillars Pedestrians mAP	60.0/54.0	68.9/66.4	57.6/52.9	46.0/37.0
PointPillars Pedestrians mAPH	47.3/42.5	55.8/53.7	45.0/41.2	33.4/26.8
StarNet Pedestrians mAP	70.1/63.2	78.6/75.7	67.9/62.7	57.2/46.1
StarNet Pedestrians mAPH	35.6/32.1	40.3/38.9	34.5/31.8	28.0/22.6
StarNet (directional) Pedestrians mAP	67.8/61.1	76.0/73.1	66.5/61.2	55.3/44.5
StarNet (directional) Pedestrians mAPH	59.9/54.0	67.8/65.2	59.2/54.5	47.0/37.8
PointPillars Vehicles mAP	62.2/54.5	81.8/80.7	55.7/50.1	31.2/23.2
PointPillars Vehicles mAPH	61.7/54.0	81.3/80.2	55.1/49.6	30.5/22.7
StarNet Vehicles mAP	64.7/56.3	83.3/82.4	58.8/53.2	34.3/25.7
StarNet Vehicles mAPH	45.5/39.6	62.0/61.3	35.9/32.5	20.5/15.4
StarNet (directional) Vehicles mAP	61.5/54.9	82.2/81.3	56.6/49.5	32.2/23.0
StarNet (directional) Vehicles mAPH	61.0/54.5	81.7/80.8	56.0/49.0	31.8/22.7

Table 5: Waymo Open Dataset *Test set* results for StarNet versus a PointPillars [22] baseline model. Every table item is the LEVEL_1/LEVEL_2 mean average precision (mAP) or heading weighted mean average precision (mAPH).