

# An Empirical Analysis of Range for 3D Object Detection

Neehar Peri<sup>1</sup>, Mengtian Li<sup>1</sup>, Benjamin Wilson<sup>2</sup>, Yu-Xiong Wang<sup>3</sup>, James Hays<sup>2</sup>, Deva Ramanan<sup>1</sup>  
 Carnegie Mellon University<sup>1</sup>, Georgia Institute of Technology<sup>2</sup>, University of Illinois Urbana-Champaign<sup>3</sup>  
 {nperi,mtli,deva}@andrew.cmu.edu, {benjaminrwilson,hays}@gatech.edu, yxw@illinois.edu

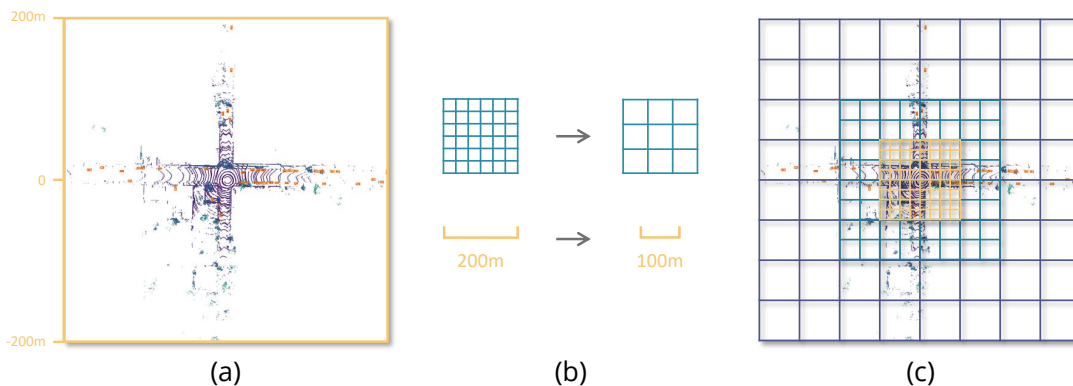


Figure 1: (a) Existing 3D LiDAR detectors struggle to detect far away objects (*e.g.*, 300m) due to time and compute constraints. (b) We explore two ways to reduce compute: adopt a coarser grid and limit the processing range. Somewhat surprisingly, we find that range is the single most effective “knob” for trading off accuracy-vs-latency, implying that detectors *should* “give up” on far-field detection to manage compute budgets. Moreover, we find that near-range detectors can exploit finer voxel sizes for higher resolution processing, while far-range detectors benefit from larger voxels. We denote models optimized for specific ranges as *range experts*. (c) To avoid blindly giving up on far-field objects, we simply combine range experts by ensembling; *e.g.*, combine 0-50m detections from the 50m expert with 50-100m detections from the 100m expert. Despite improved performance, the runtime of this naive range ensemble increases linearly with the number of range experts. To address this, we introduce near-far range-ensembles, which take inspiration from hierarchical controllers to run near-field detectors (for near-term collision avoidance) at a higher rate than far-field detectors (for long-horizon planning).

## Abstract

*LiDAR-based 3D detection plays a vital role in autonomous navigation. Surprisingly, although autonomous vehicles (AVs) must detect both near-field objects (for collision avoidance) and far-field objects (for longer-term planning), contemporary benchmarks focus only on near-field 3D detection. However, AVs must detect far-field objects for safe navigation. In this paper, we present an empirical analysis of far-field 3D detection using the long-range detection dataset Argoverse 2.0 to better understand the problem, and share the following insight: near-field LiDAR measurements are dense and optimally encoded by small voxels, while far-field measurements are sparse and are better encoded with large voxels. We exploit this observation to build a collection of range experts tuned for near-vs-far field detection, and propose simple techniques to efficiently ensemble models for long-range detection that improve efficiency by 33% and boost accuracy by 3.2% CDS.*

## 1. Introduction

3D object detection is a critical component of the autonomy stack. Despite the maturity of methods in existing literature, most treat detection range as a constant instead of an adjustable hyperparameter [35, 16, 36], likely because existing benchmarks primarily evaluate near-field detections. Motivated by highway driving and long-horizon planning, we present an empirical analysis of far-range perception and share insights that are widely applicable across model architectures. Contemporary solutions for near-field 3D detection make use of 3D voxel representations, often encoded with a bird’s-eye view (BEV) feature map. While quite intuitive, such representations scale quadratically with the spatial range of the map. We find that a primary challenge for effectively addressing long-range 3D detection is managing compute and latency demands.

**Accuracy-vs-Latency.** In this paper, we study different factors for trading off compute-vs-latency, like voxel resolution and detection range, in the context of bird’s-eye view

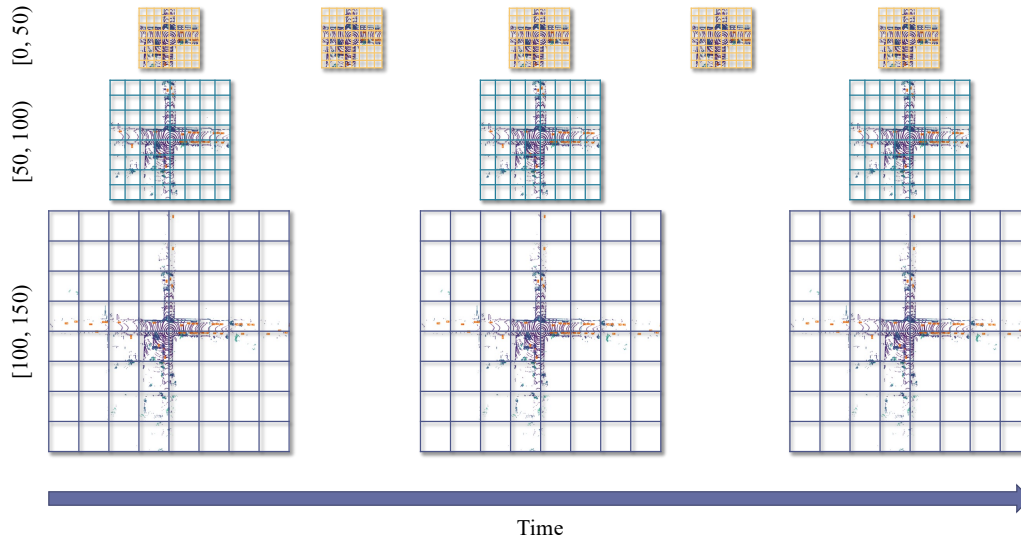


Figure 2: **Near-Far Ensemble.** We can achieve considerable efficiency gains by processing different range experts *asynchronously* at different frequencies. Importantly, near-field detectors need to be processed more frequently than far field detectors for autonomous navigation. We run the high resolution near-field model at every timestamp and process the medium and long-range models at lower frequencies. To estimate object locations for frames without far-field processing, we forecast previous detections using a constant velocity model. By simply running far-field detectors less often, we achieve a 33% increase in efficiency over the naive range ensemble.

(BEV) based 3D detection. BEV-based detectors operate on a dense 2D BEV feature map whose spatial dimensions are directly determined by the processing range and the voxel size (grid density). Interestingly, the existing literature on 2D detection has converged on image resolution and backbone depth as the handy “knobs” for trading off accuracy-vs-latency [29, 17, 32]. We revisit these questions in the context of 3D detection, and find somewhat surprisingly that *range* is an even more effective parameter for trading off these quantities (c.f. Fig. 1). For example, we show that even if the sensor (dataset) includes object annotations up to 150m, optimal accuracy-vs-latency tradeoffs may be achieved by artificially limiting the range of the model to 100m, essentially “giving up” on far-field detections during training. We posit that this is due to the distribution of annotations (e.g. fewer objects are labeled in the far-field).

**Range Ensemble.** One interesting by-product of giving-up on the far-field is that the additional compute can be reallocated to the near-field via smaller (higher resolution) voxels. Our analysis reveals that bird’s-eye view (BEV) representations can be tuned for particular ranges by adjusting other hyperparameters such as voxel resolution. We denote models optimized for specific ranges as *range experts*. Ultimately, we would like to avoid giving up on the far-field as it is important for highway driving and long-horizon planning. To avoid doing so, we simply combine range experts by ensembling; e.g., combine 0-50m detec-

tions from the 50m expert with 50-100m detections from the 100m expert. We find such an ensemble greatly boosts detection accuracy. Perhaps unsurprisingly, such an architecture is performant because it exploits a well-known but under-emphasized property of LiDAR: *farther range implies greater sparsity*.

**LiDAR Sparsity.** Interestingly, prior work [37, 25, 14] has exploited sparsity in the context of spherical voxelization or range-view processing. While performant for tasks such as semantic segmentation [40], most SOTA architectures for 3D cuboid detection still make use of rectilinear voxel grids. One reason may be that spherical warping introduces perspective distortions that warp far-field objects, making it difficult to explicitly tune range. In contrast, our range ensemble can be seen as a rectilinear approximation of spherical voxelization that *avoids* voxel distortion. Moreover, due to the popularity of rectilinear detectors, there exist more mature methods for data augmentation [9] and temporal fusion, either at the sensor level [35, 16, 36, 22] or at the feature level [19, 13], which is essential for long-range detection. Due to the strong empirical performance of 3D BEV detectors [36, 39, 18, 1], we argue that improving their range-efficiency will be increasingly important as LiDAR sensors themselves increase in range and density.

**Near-Far Ensembles.** Finally, we demonstrate that one can trivially speed up a multi-range ensemble via range-specific *asynchronous* processing. We take inspiration from

hierarchical “slow-fast” planners that run a low-frequency planner together with a high-frequency reactive controller. From a perception perspective, autonomous agents need to quickly react to near-field objects (that represent potential collisions), while far-field objects may be used for more strategic long-term planning. Concretely, we run near-range experts at high frequency and run far-range experts at a lower frequency (Fig. 2). Our results highlight an interesting observation: sometimes it is more effective to *forecast* the location of a far-field object from a previous frame’s detection than to directly process the far-field of the current frame. One reason is that the object may have appeared in the near-field of the previous frame, making it far easier to detect. We find that near-far ensembles reduce runtime by 33% with little performance decrease.

We summarize our contributions as follows:

1. We study the impact of range as a tunable parameter for 3D object detection. We draw analogy to image resolution and find the surprising conclusion that the best solution to optimize the accuracy-vs-latency tradeoff is to “give up” on far-field detection.
2. We study how detectors can generalize across ranges (e.g. train on 50m, but deploy at 100m) due to fully convolutional processing. We find that certain architectural design choices, such as voxel encoding and detector head design greatly impact across-range generalization.
3. We present a simple extension of range ensembles that takes inspiration from hierarchical controllers by running near-field detectors at higher frequency and far-field detectors at lower frequency. We show improved efficiency over the naive range-ensemble, reducing latency by 33%!

## 2. Related Work

3D detection models can be roughly categorized as: bird’s-eye view, voxel-grid, point, graph, and range-view representation models. Unlike 2D images, point clouds are amenable to a number of different representations, each with distinct advantages and disadvantages, particularly in the context of long-range detection.

**Bird’s-eye view Representations.** 3D perception using 2D convolutions enables fast, efficient feature aggregation due to mature, highly optimized kernels available in open-source libraries. However, these methods must be carefully designed to encode geometric information in the height dimension. PointPillars [16] represents a point cloud as a “pseudo-image”, applying a PointNet [23] encoding to a set of sparse pillars in the BEV. MV3D [6] explores a multi-sensor fusion model which consists of a bird’s-eye and range view of LiDAR sensor data, and RGB imagery.

However, ego-centric point clouds are *not* dense in the BEV, which consequently wastes both memory and computation. Specialized sparse operators *may* address the issues of density, but are often not as well-tuned as 2D convolutions for GPU-based computation. Further, we find that different implementations of sparse convolutions can have a significant impact on latency.

**Voxel-grid Representations.** 3D convolution provides rich, expressive geometric features at the cost of a cubic run-time w.r.t. the quantized grid dimensions leading to considerable compute challenges. VoxelNet [38] introduced the first end-to-end learning approach for 3D object detection by augmenting point features with positional encodings within a voxel-grid. SECOND [35] exploits the sparsity of a point cloud through 3D *sparse* convolutions, greatly improving run-time to speeds suitable for real-time applications. [30] combines voxel and point level processing to exploit the *efficiency* of a regular grid and the *geometric richness* of point-level features. Similar to our work, [37, 40] emphasize that point clouds are *sparse* at range, leading to an *imbalanced* spatial distribution of points. Prior works address this observation by representing point clouds with polar and cylindrical representations, respectively. However, this can lead to spatial distortions that break the translation equivariance assumed by convolutional filters. Prior work also explores multi-resolution voxel-based approaches. [11] proposes a density-aware RoI grid pooling module using kernel density estimation and self-attention with point density positional encoding to efficiently voxelize and encode LiDAR points. [15] proposes a bottom-up multi-scale voxel encoder and a top-down multi-scale feature map aggregator.

**Graph Representations.** Graph representations of point clouds encode *dynamic* neighborhoods between points while also permitting a sparse representation in the form of a sparse matrix or adjacency list. PointNet++ [24] encodes a point cloud as a hierarchical set of point-feature abstractions for 3D object classification. [33] introduces the EdgeConv operator which directly aggregates *edge* features of point clouds while maintaining permutation invariance. Point-RCNN [26] operates directly on point clouds without voxelization, using point-wise feature vectors for bottom up proposal generation. [12] proposes using random sampling to process large point clouds. Graph representations oftentimes require *costly* neighborhood computation using k-nearest or fixed-radius nearest neighbors. Despite their flexible representation, 3D detection models on state-of-the-art leaderboards are still dominated by voxel-grid and BEV based methods [2, 27].

**Range-view Representations.** Range-view refers to the projection of an unordered set of three-dimensional coordinates onto a two-dimensional grid which represents the distance from a *visible* point to the sensor. Unlike voxel-grid or graph representations, range-view is not information-

preserving for 3D data, i.e., each sensor return must have a clear line-of-sight between itself and the vantage point. LaserNet [20] combines a range-view representation with probabilistic cuboid encoding for 3D detection. [4] explores applying different kernels to the range-view image to counteract perspective distortions and large depth gradients w.r.t. to inclination and azimuth. [28] construct a two-stage approach, first performing foreground segmentation in range-view and applying sparse convolutions on the remaining points. However, much like spherical voxelization, range view suffers from perspective distortions and far-away objects have a smaller footprint in a range image.

### 3. Approach

In this section, we propose several approaches for effectively tuning range-experts (Sec. 3.1), efficiently constructing range-ensembles (Sec 3.2), and further optimizing ensemble runtime with near-far networks (Sec 3.3).

#### 3.1. Range Experts

Detection range is largely considered as a constant in the literature. However, properly tuning the detection range together with voxel size can yield a better performance-latency trade off. We derive model-families by tuning the range parameter and evaluate the accuracy and latency across different range intervals. Since the baseline detector is fully-convolutional (as are many other detectors), we can run inference at a different range from training. Taking this into consideration, we introduce a new notation of  $r_1/s \rightarrow r_2$ , where  $r_1$  represents the range the model is trained at,  $s$  represents the reciprocal of the voxel size and  $r_2$  represents the inference range. Note that the voxel size must remain the same during training and testing.

**Train-Time Range Masking.** In order to maximize the performance of a far-field range expert, one may naturally assume that allocating model capacity to near-field regions of the LiDAR sweep may negatively impact model performance. Concretely, when training a 100m range expert to detect objects between 50-100m, it seems wasteful to spend processing time on the 0-50m region of the point cloud. In particular, the point density for near-field regions are significantly higher than far-field regions. Additionally, more objects are annotated in the near-field, so we expect that this distributions shift will negatively affect generalization. In fact, the standard practice *already* crops out too-far regions for each model (by simply limiting their max range). To address this concern, we train range experts with a masked out “donut hole” (c.f. Fig. 3) to remove LiDAR points and ground truth annotations outside of the region of interest. Somewhat surprisingly, we find that this “donut hole” range masking during training time hurts performance. In practice, it seems that learning to detect near-field cars helps to detect far-field cars.

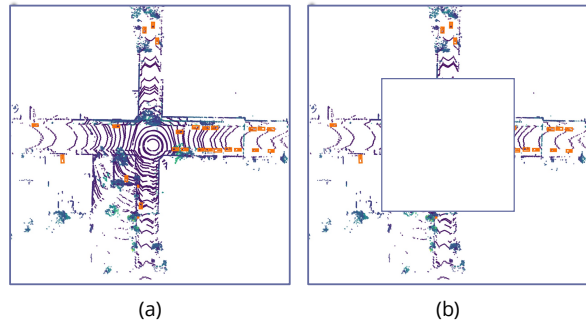


Figure 3: **Range Masking.** We evaluate the effectiveness of range masking when training and evaluating range experts. We find that (a) range masking negatively impacts performance during training. Surprisingly, learning to detect near-range cars improves the detection of far-field cars. However, (b) range masking reduces latency during inference as the additional sparsity increases the efficiency of sparse voxel encoders.

**Test-Time Range Masking.** Although we find that “donut hole” range masking hurts performance during training, we find that only applying it during test-time does not affect detection accuracy (c.f. Fig 3). In fact, it provides a modest speedup because it makes better use of sparse computation. Note that this speedup is largely dependent on the sparse voxelization encoder. We find that this provides considerable improvement for models using sponcv 1.0, but limited improvement for models using sponcv 2.0.

**Up-sampling Far-Field Objects.** One reason that near-field detections may help improve far-field detections is because there are simply too few objects annotated in the far field. Taking inspiration from long-tailed detection literature [39, 21], we can upsample LiDAR sweeps with more far-field ground truth objects and paste more examples of far-field objects into each sweep. However, we find that this does even worse than range masking at training time. We posit that the distribution of objects seen during training time is considerably different than that seen during inference, leading to a significant performance drop.

#### 3.2. Range Ensembles

After training each range-expert, we can ensemble their detections by combining predictions from the respective expert models. One strategy is to pool all detections together and perform non-maximal suppression to remove duplicate detections across range experts. However, we find that this approach introduces more false positives and negatively impacts overall performance. Instead, we run each range expert as normal, but simply post-process detections from each range expert such that they only contribute to detect-



ing objects within their tuned range. However, this wastes a considerable amount of compute. Specifically, we must still run our 100m range expert on the dense 0-50m regions of the point cloud, despite discarding these predictions during post processing. Instead, we opt to exploit sparse convolutions to speed up inference using test-time range masking.

### 3.3. Near-Far Ensembles

We explore the idea of near-far ensembles, as shown in Fig. 2, to considerably improve the run-time of a range ensembles. We can achieve these efficiency gains by processing different range experts *asynchronously* at different frequencies. Intuitively, we care more about changes to our near-field to avoid immediate collisions and can update the far-field region less frequently for longer-term planning. We run the high resolution near-field model at every timestamp and process the long-range models at every other timestep. To estimate object locations for frames without far-field processing, we forecast previous detections using a constant velocity model. We are able to predict per-object velocity estimates from our models because the input to our detectors is a stack of aggregated LiDAR sweeps which implicitly encodes object motion. This constant velocity forecast, which simply updates the past object location using the predicted object velocity estimate (e.g. `box.center += box.velocity*time_delta`), is reasonable because we are only forecasting 0.5 seconds into the future. Intuitively, all object motion can be linearized given a sufficiently small time delta. Importantly, since all detections are in the ego-vehicle coordinate frame, forecasting past detections into the current frame requires ego-motion compensation between frames. We present simplified python-like pseudo-code in Alg. 1.

## 4. Experiments

In this section, we demonstrate how detection range affects the accuracy-latency trade off for 3D detectors. Next, we evaluate a number of popular 3D BEV-based detectors, including PointPillars [16], CBGS[39], CenterPoint [36], and TransFusion [1] on Argoverse 2.0 [34], a long-range detection dataset.

### 4.1. Dataset and Metrics

We conduct our experiments on Argoverse 2.0 [34], an autonomous driving dataset with data collected in six US cities. It labels 26 semantic classes for the 3D detection task. Notably, Argoverse 2.0 produces long-range LiDAR point clouds and object annotations (up to  $\pm 150$ m). In comparison, KITTI [10] only annotates up to +70m (with a front facing LiDAR), nuScenes [2] annotates up to  $\pm 50$ m, and Waymo [27] annotates up to  $\pm 75$ m. Following standard training protocols used in the nuScenes setup, we adopt 5-frame aggregation for LiDAR densification. We assume

---

**Algorithm 1:** We present python style pseudo-code for the near-far range ensemble. Concretely, we run the near-range expert model at every time step, but only run the far-range expert every *freq* timesteps (default is 2). We assume that the forecaster compensates for ego-motion.

---

```

#near_expert: Near-range expert detector
#far_expert: Far-range expert detector
#freq: Frequency of of far-range detector
#donut_crop: Removes near-range lidar points
#forecast: Constant-velocity forecast
#dets: Dict[List] of detections

for time, lidar_sweep in enumerate(data):
    # Run near-field range expert
    near_dets = near_expert(lidar_sweep)

    if time % freq == 0:
        # Run far-field range expert
        cropped_sweep = donut_crop(lidar_sweep)
        far_dets = far_expert(cropped_sweep)
    else:
        # Forecast prev. detections
        far_dets = forecast(dets[time - 1])

    dets[time] = {near_dets, far_dets}

```

---

that we are provided with ego-vehicle pose for prior frames to align all LiDAR sweeps to the current ego-vehicle pose. Since LiDAR returns are sparse, this densification step is essential for long-range detection.

We evaluate our model using the composite detection score (CDS), a summary metric defined as the product of average precision, computed as an average of four different true-positive thresholds (0.5, 1.0, 2.0, and 4.0 meters) and the sum of the complement of the normalized true positive errors (average translation error (ATE), average scale error (ASE), and average orientation error (AOE)). We refer readers to [5] for a detailed description of this metric.

### 4.2. Implementation Details

BEV-based detectors often follow the same general architecture. First, sparse LiDAR points are voxelized to form a dense feature map. This dense map is then processed by the SECOND [35] backbone and FPN neck. Lastly, PointPillars and CBGS process this BEV feature using a minimal SSD-like detection head. CenterPoint uses a center-based detection head which predicts object center’s using a heatmap and regresses all other attributes, and TransFusion uses a DETR-like transformer decoder, which directly predicts amodal bounding boxes. All four models predict object semantics and regression bounding box location, size, orientation, and instantaneous velocity. We keep the model architecture fixed for our study, only tuning the range and

ID	Model	Method	Point Proc.	Backbone	Neck	Head	Post Proc.
1	PointPillars	50/4 → 50	10.5 ± 3.0	3.5 ± 0.2	1.9 ± 0.1	1.2 ± 0.1	58.2 ± 1.6
2	CBGS	50/12.5 → 50	43.6 ± 4.0	4.7 ± 0.3	2.5 ± 0.1	1.2 ± 0.2	55.9 ± 3.5
3	CenterPoint	50/12.5 → 50	45.8 ± 5.5	2.7 ± 0.3	0.8 ± 0.03	42.8 ± 0.6	440.9 ± 48.1
4	TransFusion-L	50/12.5 → 50	264.9 ± 45.8	4.5 ± 0.2	1.3 ± 0.3	9.8 ± 3.4	1.5 ± 0.5
1	PointPillars	100/4 → 100	25.6 ± 7.3	10.6 ± 0.1	13.1 ± 0.1	4.1 ± 0.1	62.0 ± 1.3
2	CBGS	100/6.25 → 100	40.0 ± 3.2	4.7 ± 0.1	2.5 ± 0.1	1.2 ± 0.1	58.6 ± 1.9
3	CenterPoint	100/6.25 → 100	42.3 ± 6.1	4.8 ± 0.2	0.8 ± 0.1	42.7 ± 0.6	448.1 ± 54.8
4	TransFusion-L	100/6.25 → 100	257.7 ± 34.9	4.5 ± 0.4	1.3 ± 0.1	9.3 ± 2.6	1.5 ± 0.2
1	PointPillars	150/2 → 150	4.0 ± 1.2	6.6 ± 0.3	8.1 ± 0.2	2.7 ± 0.3	60.0 ± 9.1
2	CBGS	150/3.125 → 150	35.5 ± 1.9	3.0 ± 0.1	1.7 ± 0.1	1.1 ± 0.1	58.8 ± 1.5
3	CenterPoint	150/3.125 → 150	33.5 ± 4.4	3.3 ± 0.2	0.6 ± 0.1	26.1 ± 1.0	291.1 ± 66.9
4	TransFusion-L	150/3.125 → 150	240.9 ± 31.6	3.3 ± 0.7	0.8 ± 0.1	9.0 ± 2.1	1.5 ± 0.5

Table 1: **Impact of Range on Timing.** We find that increasing range and proportionally decreasing voxel resolution keeps run time (in milliseconds) approximately constant. Within a fixed compute budget, tuning range and voxel resolution are the two key “knobs” to trade off latency and accuracy. Further, we find that the point-processing takes a majority of the run time (excluding post-processing). Lastly, we note that CenterPoint’s head is more than four times slower than the transformer head in TransFusion and ten times slower than the anchor head in PointPillars. Empirically, we find that this slowdown is due to inefficient bounding box decoding from the CenterPoint regression heads (which can be significantly optimized).

voxel size. We use the open source implementations of these four detectors from mmdetection3d [7, 1, 21]. We adopt a basic set of data augmentations, including global 3D transformations, flip in BEV, and point shuffling during training. We train our model with 8 RTX 3090 GPUs and a batch size of 1 per GPU. The training noise (from random seed and system scheduling) is  $< 1\%$  of the accuracy (standard deviation normalized by the mean). We also report the mean timing of three runs for key experiments. For consistent measurement of model runtime, we evaluate with a batch size 1 on a Tesla V100 GPU [29, 17, 31].

**Timing of Individual Components:** We provided a detailed component-wise runtime analysis for the models in Table 1. CBGS is architecturally identical to PointPillars, but uses a VoxelNet encoder instead of a PillarEncoder. CenterPoint is architecturally identical to CBGS, but uses a center regression head instead of an anchor-based detector head. TransFusion-L (the LiDAR-only variant of TransFusion) is architecturally identical to CenterPoint, but uses a DETR-like transformer decoder as a detector head. Although CBGS, CenterPoint and TransFusion use the same VoxelNet encoder, we find that TransFusion’s point processing is nearly 5x slower, likely because it uses spconv1.0 rather than spconv2.0 [8]. Since the SECOND backbone and neck are identical between the four models, timing numbers are consistent. Further, we note that the anchor-based detector head used in PointPillars and CBGS is the fastest, followed by TransFusion’s head. CenterPoint’s head

is the heaviest, notably taking 40x longer than the anchor-based head. TransFusion’s post-processing time is significantly faster than other models because the transformer head decoding stage does not perform non-maximal suppression (NMS).

In general, post-processing takes a considerable fraction of the runtime for all models. This is a result of using research-level code and can be further optimized, but it is beyond the scope of this work. We find that this post-processing time is relatively constant within model-families. For subsequent timing results we omit the post-processing time since this is a dominating factor which makes analysis more difficult. In practice, this can be sped up using GPU implementations of max-pooled NMS instead of the standard (greedy association) NMS [3].

### 4.3. Empirical Analysis

We evaluate range experts for a variety of detector architectures. Unsurprisingly, we find that the range ensemble consistently outperforms range experts, but is nearly 3x slower. The near-far ensemble provides a “sweet spot” that is 33% faster than the range ensemble, while also often performing better than the best range-expert. Interestingly, we find that some architectures are better at generalizing across-range.

**Tuning Range Experts.** We evaluate a simple change to our training protocol which modifies the data distribution to specialize to a particular range interval. As shown in Tab 2,

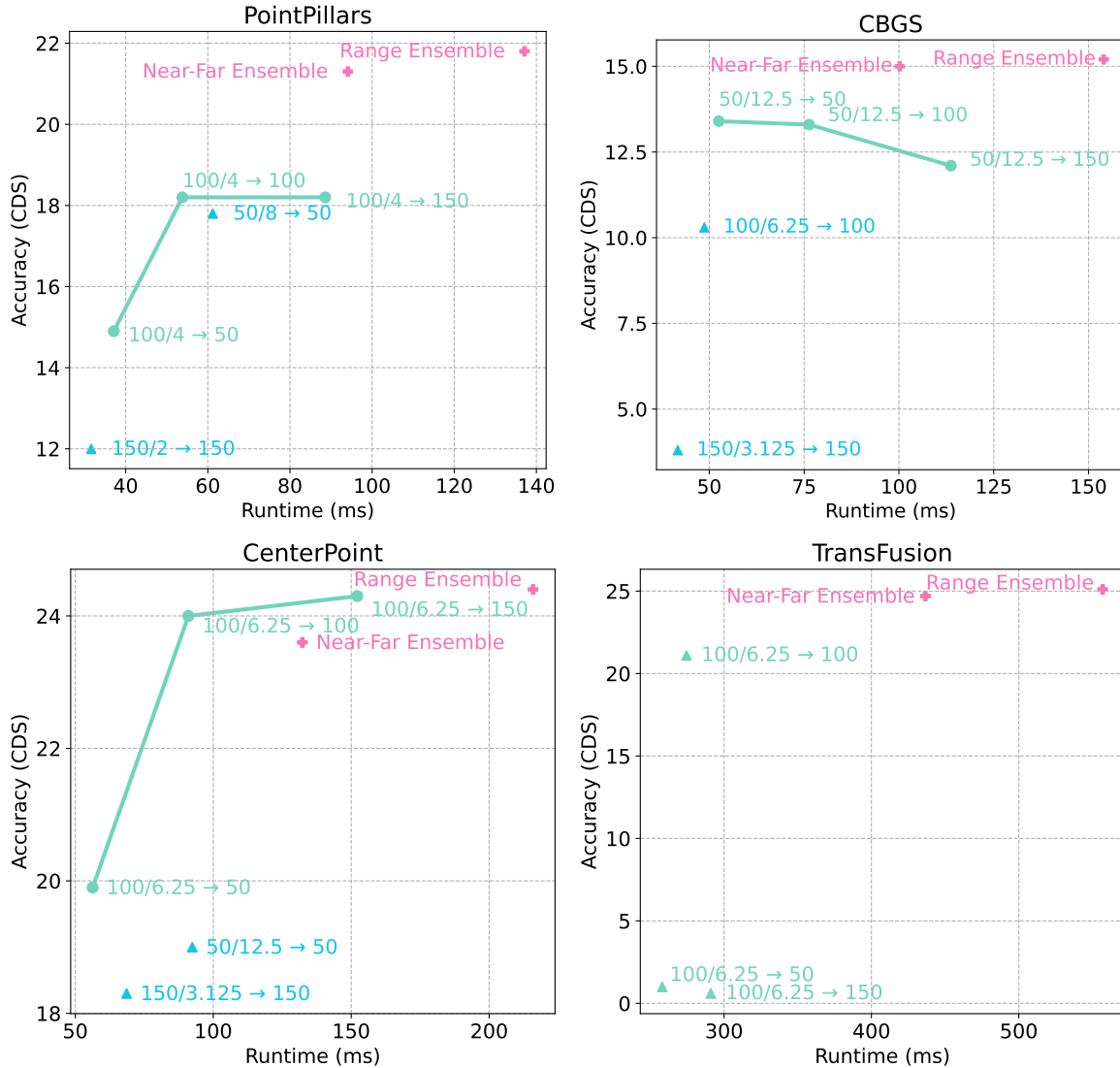


Figure 4: Without re-training, we can re-run a 100m trained PointPillars model (top left, 100/4 green dots) at different ranges (top left, green curve). Interestingly, the 100m PointPillars range expert outperforms both the 50m and 150m range experts (top left, blue triangles), suggesting that we should “give up” on far-field detections. However, to avoid giving up on long-range detection, we ensemble our range experts to achieve considerably higher performance than any single PointPillars model, but is almost 3x as slow as the 100m range expert (top left, pink star). In contrast, the near-far ensemble achieves a “sweet spot” between the speed of the single range expert and the performance of the naive range ensemble. We see a similar trend with the CenterPoint model family (bottom left). Again, the 100m CenterPoint range expert beats the 50m and 150m range experts. However, the CenterPoint 100m range ensemble evaluated at 150m almost matches the performance of the range ensemble, serving as the best single model. In such cases, we find that the range-ensemble does not provide significant benefit. Unlike the PointPillars and CenterPoint model families, we see that the best-performing CBGS range expert is the 50m model (top right). However, we note that the performance is considerably lower than either PointPillars or CenterPoint. Further, running the 50m CBGS range expert at 100m and 150m leads to degraded performance. This suggests that CBGS does not generalize well to far-field detection. Lastly, we consider TransFusion, a recent state-of-the-art transformer-based detector (bottom right). Unlike the prior three models, we find that we cannot easily run TransFusion in a “fully-convolutional” mode. Specifically, although we note strong performance for the 100m range expert when run at 100m, we notice that performance drops to nearly 0 CDS when run on either the 50m or 150m ranges. We posit that TransFusion’s use of relative positional encoding rather than metric encoding leads to catastrophically poor across-range generalization.

forcing models to specialize to particular ranges by masking out “donut-holes” in the point cloud and upsampling far-field objects does not result in better performance compared to the standard approach of training on the full range of LiDAR points. We find that train-time range masking hurts performance, likely because the model is trained with less data overall. For example, in the 50-100m range, the 100m range expert achieves 13.1 CDS whereas the model trained with a “donut” shaped LiDAR sweep only attains 9.6 CDS. Similarly, in the 100-150m range, we see that the 150m range expert achieves 5.3 CDS whereas the model trained with a “donut” shaped LiDAR sweep gets 3.6 CDS. Similarly, We find that upsampling classes within a specific range interval does worse than the baseline, likely because the distribution of objects seen during train time is significantly different than that seen at test-time. Based on this investigation, range experts should simply train on the full range without specializing for specific range intervals. However, we find that masking out “donut holes” in the point cloud during inferences does not affect detection performance, and provides a modest speedup due to sparse computation.

**Generalization Across Range.** As shown in Figure 4, some models generalize well across ranges (e.g. CenterPoint (bottom right) green curve increases when evaluated on a range beyond the training range), some generalize poorly (e.g. CBGS (top left) green curve decreases when evaluated beyond the training range.), and some don’t generalize at all (e.g. TransFusion achieves nearly 0% CDS when evaluated on a range that is different than the training range). We examine these trends through the lens of model architectures and training losses.

First, we note that PointPillars has *some* generalization capability across ranges. Notably, when we evaluate

ID	Method	0-50m.	50-100m	100-150m
1	50/8 → 50	31.5		
2	100/4 → 100	26.9	13.1	
3	+ Range Masking		9.6	
4	+ Up-sampling		7.7	
5	150/2 → 150	16.9	9.4	5.3
6	+ Range Masking			3.6
7	+ Up-sampling			2.1

Table 2: **Range Specialization.** We evaluate range-masking and object up-sampling using PointPillars, and find that both negatively impact the performance of the range-expert, suggesting that the best strategy for training range experts is to generalize to other ranges outside of the region of interest. This conclusion holds for both the 100/4 → 100 and 150/2 → 150 range experts.

the 100m range expert at 150m, the performance does not change, indicating that the model likely predicts all far-field detections with a lower confidence than near-field detections. We argue that knowing what you don’t know is a form of generalization.

Second, as described in Section 4.2, CBGS is architecturally identical to PointPillars, but uses a VoxelNet encoder instead of a PointNet encoder. Unlike PointPillars, CBGS predicted far-field detections with higher confidence than some near-field detections, resulting in lower performance when evaluating long-range detections. We posit that PointPillars’ PointNet encoder captures local features that generalized better than the global features encoded by VoxelNet.

Next, we consider the surprising across-range generalization capabilities of CenterPoint. CenterPoint is architecturally identical to CBGS, but uses a center regression head instead of an anchor-based detector head. Although these are architecturally similar, we posit that the difference in training loss significantly impacts generalization. CBGS attempts to maximize the IOU of its predictions with the ground truth. In contrast, CenterPoint learns to regress a heatmap of Gaussian targets. We posit that these “soft targets” act as a form of data augmentation, which make it easier to train the model.

Lastly, we consider TransFusion, which doesn’t generalize across-ranges at all. Specifically, although we observe strong performance for the 100m range expert when evaluating at 100m, we notice that performance drops to nearly 0 CDS when run on either the 50m or 150m ranges. TransFusion-L (the LiDAR-only variant of TransFusion) is architecturally identical to CenterPoint, but uses a DETR-like transformer decoder as a detector head. We posit that using metric positions for positional encoding rather than relative positions may yield better across-range generalisation.

## 5. Conclusion

We provide analysis on the effect of detection range for 3D object detectors, showing that range is an important “knob” to trade off accuracy and latency. We use our analysis to build a simple ensemble of range experts that exploits a fundamental property of LiDAR; namely that sensor returns become sparse at range, allowing for coarser voxel binning. While highly performant, an ensemble of range experts can be slow, and is unsuitable for real-time applications like autonomous navigation. To address this limitation, we propose near-far ensembles, which run near-field detectors at higher frequency (for immediate collision avoidance and far-field detectors at lower frequency (for long-horizon planning). We also explore the generalization of BEV-based 3D detectors across range and find that certain combinations of voxel encoders and detector heads lead to better across-range generalization.



## References

- [1] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1090–1099, 2022. 2, 5, 6
- [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 multi-modal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 3, 5
- [3] Lile Cai, Bin Zhao, Zhe Wang, Jie Lin, Chuan Sheng Foo, Mohamed Sabry Aly, and Vijay Chandrasekhar. Maxpool-nms: getting rid of nms bottlenecks in two-stage object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9356–9364, 2019. 6
- [4] Yuning Chai, Pei Sun, Jiquan Ngiam, Weiyue Wang, Benjamin Caine, Vijay Vasudevan, Xiao Zhang, and Dragomir Anguelov. To the point: Efficient 3d object detection in the range image with graph convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2021. 4
- [5] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3D tracking and forecasting with rich maps. In *CVPR*, 2019. 5
- [6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 3
- [7] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. <https://github.com/open-mmlab/mmdetection3d>, 2020. 6
- [8] Spconv Contributors. Spconv: Spatially sparse convolution library. <https://github.com/traveller59/spconv>, 2022. 6
- [9] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2918–2927, 2021. 2
- [10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 5
- [11] Jordan SK Hu, Tianshu Kuai, and Steven L Waslander. Point density-aware voxels for lidar 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8469–8478, 2022. 3
- [12] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020. 3
- [13] Rui Huang, Wanyue Zhang, Abhijit Kundu, Caroline Pantofaru, David A. Ross, Thomas A. Funkhouser, and Alireza Fathi. An lstm approach to temporal 3d object detection in lidar point clouds. In *ECCV*, 2020. 2
- [14] Yanqin Jiang, Li Zhang, Zhenwei Miao, Xiatian Zhu, Jin Gao, Weiming Hu, and Yu-Gang Jiang. Polarformer: Multi-camera 3d object detection with polar transformers. *arXiv preprint arXiv:2206.15398*, 2022. 2
- [15] Hongwu Kuang, Bei Wang, Jianping An, Ming Zhang, and Zehan Zhang. Voxel-fpn: Multi-scale voxel feature aggregation for 3d object detection from lidar point clouds. *Sensors*, 20(3):704, 2020. 3
- [16] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 1, 2, 3, 5
- [17] Mengtian Li, Yuxiong Wang, and Deva Ramanan. Towards streaming perception. In *ECCV*, 2020. 2, 6
- [18] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird’s-eye view representation. *ArXiv*, abs/2205.13542, 2022. 2
- [19] 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
- [20] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12677–12686, 2019. 4
- [21] Neehar Peri, Achal Dave, Deva Ramanan, and Shu Kong. Towards long tailed 3d detection. *Conference on Robot Learning*, 2022. 4, 6
- [22] Neehar Peri, Jonathon Luiten, Mengtian Li, Aljosa Osep, Laura Leal-Taixe, and Deva Ramanan. Forecasting from lidar via future object detection. *arXiv:2203.16297*, 2022. 2
- [23] 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. 3
- [24] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 3
- [25] Meytal Rapoport-Lavie and Dan Raviv. It’s all around you: Range-guided cylindrical network for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2992–3001, 2021. 2
- [26] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3d object proposal generation and detection from point

- cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019. [3](#)
- [27] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. [3](#), [5](#)
- [28] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5725–5734, 2021. [4](#)
- [29] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. [2](#), [6](#)
- [30] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, pages 685–702. Springer, 2020. [3](#)
- [31] Chittesh Thavamani, Mengtian Li, Nicolas Cebren, and Deva Ramanan. Fovea: Foveated image magnification for autonomous navigation. In *ICCV*, 2021. [6](#)
- [32] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 13029–13038, 2021. [2](#)
- [33] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. [3](#)
- [34] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [5](#)
- [35] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. [1](#), [2](#), [3](#), [5](#)
- [36] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, pages 11784–11793, 2021. [1](#), [2](#), [5](#)
- [37] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020. [2](#), [3](#)
- [38] 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. [3](#)
- [39] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019. [2](#), [4](#), [5](#)
- [40] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9939–9948, 2021. [2](#), [3](#)