Streaming Object Detection for 3-D Point Clouds

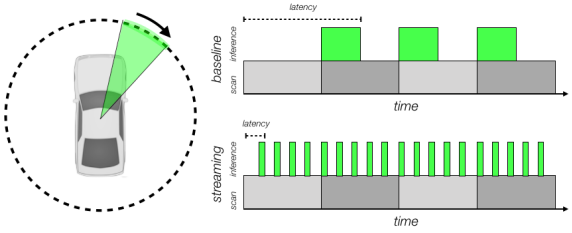
Wei Han[1], Zhengdong Zhang[1], Benjamin Caine[1], Brandon Yang[1], Christoph Sprunk[2], Ouais Alsharif[2], Jiquan Ngiam[1], Vijay Vasudevan[1], Jonathon Shlens[1], and Zhifeng Chen[1]

[1] Google Brain
[2] Waymo, LLC
{weihan,zhangzd}@google.com

Abstract. Autonomous vehicles operate in a dynamic environment, where the speed with which a vehicle can perceive and react impacts the safety and efficacy of the system. LiDAR provides a prominent sensory modality that informs many existing perceptual systems including object detection, segmentation, motion estimation, and action recognition. The latency for perceptual systems based on point cloud data can be dominated by the amount of time for a complete rotational scan (e.g. 100 ms). This built-in data capture latency is artificial, and based on treating the point cloud as a camera image in order to leverage camera-inspired architectures. However, unlike camera sensors, most LiDAR point cloud data is natively a *streaming* data source in which laser reflections are sequentially recorded based on the precession of the laser beam. In this work, we explore how to build an object detector that removes this artificial latency constraint, and instead operates on native streaming data in order to significantly reduce latency. This approach has the added benefit of reducing the peak computational burden on inference hardware by spreading the computation over the acquisition time for a scan. We demonstrate a family of streaming detection systems based on sequential modeling through a series of modifications to the traditional detection meta-architecture. We highlight how this model may achieve competitive if not superior predictive performance with state-of-the-art, traditional non-streaming detection systems while achieving significant latency gains (e.g. $1/15^{\text{th}} - 1/3^{\text{rd}}$ of peak latency). Our results show that operating on LiDAR data in its native streaming formulation offers several advantages for self driving object detection – advantages that we hope will be useful for any LiDAR perception system where minimizing latency is critical for safe and efficient operation.

1 Introduction

Self-driving cars are typically equipped with an array of sensors to robustly identify objects across highly variable environmental conditions [7,54,13,3]. In turn, driving in the real world requires responding to this large array of data with



meta-architecture	baseline	streaming			
localized RF		✓	✓	✓	✓
stateful NMS			✓	✓	✓
stateful RNN				✓	✓
larger model					✓
accuracy (mAP)					
pedestrians	54.9	40.1	52.9	53.5	60.1
vehicles	51.0	10.5	39.2	48.9	51.0

Fig. 1: **Streaming object detection pipelines computation to minimize latency without sacrificing accuracy**. LiDAR accrues a point cloud incrementally based on a rotation around the z axis. Instead of artificially waiting for a complete point cloud scene based on a 360° rotation (*baseline*), we perform inference on subsets of the rotation to pipeline computation (*streaming*). Gray boxes indicate the duration for a complete rotation of a LiDAR (e.g. 100 ms [53,13]). Green boxes denote inference time. The expected latency for detection – defined as the time between a measurement and a detection decreases substantially in a streaming architecture (dashed line). At 100 ms scan time, the expected latency reduces from $\sim$ 120 ms (baseline) versus $\sim$ 30 ms (streaming), i.e. $>3\times$ (see text for details). The table compares the detection accuracy (mAP) for the baseline on pedestrians and vehicles to several streaming variants [32].

minimal latency to maximize the opportunity for safe and effective navigation [28].

LiDAR represents one of the most prominent sensory modalities in SDC systems [7,54] informing object detection [60,64,58,59], region segmentation [36,33] and motion estimation [27,62]. Existing approaches to LiDAR-based perception derive from a family of camera-based approaches [49,39,19,5,25,9], requiring a complete 360° scan of the environment. This artificial requirement to have the complete scan limits the minimum latency a perception system can achieve, and effectively inserts the LiDAR scan period into the latency [3]. Unlike CCD cameras, many LiDAR systems are *streaming* data sources, where data arrives sequentially as the laser rotates around the z axis [2,21].

Object detection in LiDAR-based perception systems [60,64,58,59] presents a unique and important opportunity for re-imagining LiDAR-based meta architec-

[3] LiDAR typically operates with a 5-20 Hz scan rate. We focus on 10 Hz (i.e. 100 ms period), because several prominent academic datasets employ this scan rate [53,13], however our results may be equally applied across the range of scan rates available.

tures in order to significantly minimize latency. In particular, streaming LiDAR data permits the design of meta-architectures which operate on the data as it arrives, in order to pipeline the sensory readout with the inference computation to significantly reduce latency (Figure 1).

In this work, we propose a series of modifications to standard meta architectures that may generically adapt an object detection system to operate in a streaming manner. This approach combines traditional elements of single-stage detection systems [48,39] as well as design elements of sequence-based learning systems [6,26,18]. The goal of this work is to show how we can modify an existing object detection system – with a minimum set of changes and the addition of new operations – to efficiently and accurately emit detections as data arrives (Figure 1). We find that this approach matches or exceeds the performance of several baseline systems [32,43], while substantially reducing latency. For instance, a family of streaming models on pedestrian detection achieves up to 60.1 mAP compared to 54.9 mAP for a baseline non-streaming model, while reducing the expected latency $> 3\times$. In addition, the resulting model better utilizes computational resources by pipelining the computation throughout the duration of a LiDAR scan. We demonstrate through this work that designing architectures to leverage the native format of LiDAR data achieves substantial latency gains for perception systems generically that may improve the safety and efficacy of SDC systems.

2 Related Work

2.1 Object detection in camera images

Object detection has a long history[12,8,55,10,35]. The re-emergence of convolutional neural networks (CNN) for computer vision [31,30] inspired the field to harness both the rich image features and final training objective of a CNN model for object detection [50]. In particular, the features of a CNN trained on an image classification task proved sufficient for providing reasonable candidate locations for objects [15]. Subsequent work demonstrated that a single CNN may be trained in an end-to-end fashion to sub-serve for both stages of an object detection system [49,14]. The resulting two-stage systems, however, suffered from relatively poor computational performance [24]. This inspired researchers to pursue one stage object detection systems [39,48] at the cost of lower predictive performance [39,48].

2.2 Object detection in videos

Strategies to tackle object detection in videos include breaking up the problem into a computationally-heavy detection phase and a computationally-light tracking phase [37], and building blended CNN recurrent architectures for providing memory between time steps of each frame [41,38]. Recent methods have also explored the potential to persist and update a memory of the scene[22,4]. This

problem reflects possibly the closest set of methods relevant to our proposed work. predictions.

In our work, we examine the possibility of dividing a *single frame* into slices that can be processed in a streaming fashion. A time step in our setup correspond to a slice of one frame. An object may appear across multiple slices, but generally, each slice contains distinct objects. This may require similar architectures to video object detection (e.g., convolutional LSTMs) in order to provide a memory and state of earlier slices for refining or merging detections [57,44,56].

2.3 Object detection in point clouds

The prominence of LiDAR systems in self-driving cars necessitated the application of object detection to point cloud data [13,3]. Much work has employed object detection systems originally designed for camera images to point cloud data by projecting such data from a Bird's Eye View (BEV) [60,40,59,32] (but see [42]) or a 3-D voxel grid [64,58]. Alternatively, some methods have re-purposed two stage object detector design with a region-proposal stage but replacing the feature extraction operations [61,52,45].

In parallel, others have pursued replacing a discretization operation with a featurization based on native point-cloud data [46,47]. Such methods have led to methods for building detection systems that blend aspects of point cloud featurization and traditional object detectors on cameras [32,63,43] to achieve favorable performance for a given computational budget.

3 Methods

3.1 Streaming LiDAR inputs

A LiDAR system for an SDC measures the distance to objects by shining multiple lasers at fixed inclinations and measuring the reflectance in a sensor [7,54]. The lasers precess around the z axis, and make a complete rotation at a 5-20 Hz scan rate. Typically, SDC perception systems artificially wait for a complete 360° rotation before processing the data.

This work simulates a streaming system with the Waymo Open Dataset [53] by artificially manipulating the point cloud data [4]. The native format of point cloud data are *range images* whose resolution in height and width correspond to the number of lasers and the rotation speed and laser pulse rate [42]. In this work, we artificially slice the input range image into n vertical strips along the image width to experiment with streaming detection models.

[4] KITTI [13] is the most popular LiDAR detection dataset, however this dataset provides annotations within a 90° frustum. The Waymo Open Dataset provides a completely annotated 360° point cloud which is necessary to demonstrate the efficacy of the streaming architecture across all angular orientations.

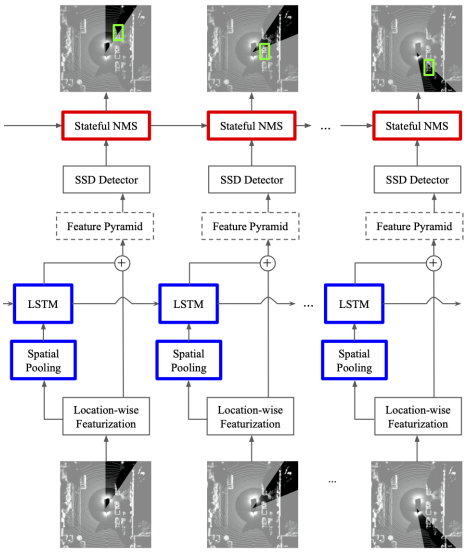


Fig. 2: **Diagram of streaming detection architecture.** A streaming object detection system processes a spatially restricted slice of the scene. We introduce two stateful components: Stateful NMS (red) and a LSTM (blue) between input slices. Detections produced by the model are denoted in green boxes. Dashed line denotes feature pyramid uniquely employed in [32].

3.2 Streaming object detection

This work introduces a meta-architecture for adapting object detection systems for point clouds to operate in a streaming fashion. We employ two models as a baseline to demonstrate how the proposed changes to the meta-architecture are generic, and may be employed in notably different detection systems.

We first investigate PointPillars [32] as a baseline model because it provides competitive performance in terms of predictive accuracy and computational budget. The model divides the x-y space into a top-down 2D grid, where each grid cell is referred to as a *pillar*. The points within each non-zero pillar are featurized using a variant of a multi-layer perceptron architecture designed for point cloud data [46,47]. The resulting d-dimensional point cloud features are scattered to the grid and a standard multi-scale, convolutional feature pyramid [34,64] is computed on the spatially-arranged point cloud features to result in a global activation map. The second model investigated is StarNet [43]. StarNet is an entirely point-based detection system which uses sampling instead of a learned region proposal to operate on targeted regions of a point cloud. StarNet avoids usage of global information, but instead targets the computational demand to regions of interest, resulting in a locally targeted activation map. See the Appendix for architecture details for both models. For both PointPillars and StarNet, the resulting activation map is regressed on to a 7-dimensional target parameterizing

the 3D bounding box as well as a classification logit [58]. Ground truth labels are assigned to individual anchors based on intersection-over-union (IoU) overlap [58,32]. To generate the final predictions, we employ oriented, 3-D multi-class non-maximal suppression (NMS) [15].

In the streaming object detection setting, models are limited to a restricted view of the scene. We carve up the scene into n slices (Section 3.1) and only supply an individual slice to the model (Figure 2). Note that the scene division is performed in polar coordinates. We simplify the parameterization by requiring that slices are non-overlapping (i.e., the stride between slices matches the slice width). We explore a range of n in the subsequent experiments.

For the PointPillars convolutional backbone, we assume that sparse operators are employed in the convolutional backbone to avoid computation on empty pillars [17]. Note that no such implementation is required for StarNet because the model is designed to only operate on populate regions of the point cloud [43].

3.3 Stateful non-maximum suppression

Objects which subtend large angles of the LiDAR scan present unique challenges to a steaming detection system which necessarily have a limited range of sensor input (Table 1). However, to our knowledge, there is no prior work on NMS with state. Hence, we explore a modified NMS technique that maintains *states for the streaming setting*. Generically, NMS with state may take into account detections in the previous k slices to determine if a new detection is indeed unique. Therefore, detections from the current slice can be suppressed by those in previous slices. Stateful NMS does not require a complete LiDAR rotation and may likewise operate in a streaming fashion. Furthermore. In our experiments, for a broad range of n, we found that $k = 1$ achieves as good of performance as $k{=}n - 1$ which would correspond to a global NMS available to a non-streaming system. We explore the selection of k in Section 4.2.

3.4 Adding state with recurrent architectures

Given a restricted view of the point cloud scene, streaming models can be limited by the context available to make predictions. To increase the amount of context that the model has access to, we consider augmenting the baseline model to maintain a recurrent memory across consecutive slices. This memory may be placed in the intermediate representations of the network.

We select a standard single layer LSTM as our recurrent architecture, although any other RNN architecture may suffice [23]. The LSTM is inserted after the final global representation from either baseline model before regressing on to the detection targets. For instance, in the PointPillars baseline model, this corresponds to inserting the LSTM after the convolutional representation. The memory input is then the spatial average pooling for all activations of the final

convolutional layer before the feature pyramid [5]. Note that the LSTM memory does not hard code the spatial location of the slice it is processing.

Based on earlier preliminary experiments, we employed LSTM with 128 hidden dimensions and 256 output dimensions. The output of the LSTM is summed with the final activation map by broadcasting across all spatial dimensions in the hidden representation. More sophisticated elaborations are possible, but are not explored in this work [37,41,38,57,44,56].

4 Results

We present all results on the Waymo Open Dataset [53]. All models are trained with Adam [29] using the Lingvo machine learning framework [51] built on top of TensorFlow [1] [6]. No data augmentation is used. We perform hyper-parameter tuning through cross-validated studies and final evaluations on the corresponding test datasets. In our experiments, we explore how the proposed meta-architecture for streaming 3-D object detection compares to a standard object detection system, i.e. PointPillars [32] and StarNet [43]. All experiments use the first return from the medium range LiDAR (labeled TOP) in the Waymo Open Dataset, ignoring the four short range LiDARs for simplicity. This results in slightly lower baseline and final accuracy as compared to previous results [43,53,63].

We begin with a simple, generic modification to the baseline architecture by limiting its spatial view to a single slice. This modification permits the model to operate in a streaming fashion, but suffers from a severe performance degradation. We address these issues through stateful variants of non-maximum suppression (NMS) and recurrent architectures (RNN). We demonstrate that the resulting streaming meta-architecture restores competitive performance with favorable computation and latency benefits. Importantly, such a meta-architecture may be applied generically to most other point cloud detection models [43,64,58,60,40,59,42].

4.1 Spatially localized detection degrades baseline models

To build a baseline model for streaming, we modify existing models by restricting the operation to a slice of point cloud (Figure 2). The model may in principle operate with a reduced latency since inference can proceed before a complete LiDAR rotation (Figure 1).

Two free parameters that govern a localized receptive field are the angle of restriction and the stride between subsequent inference operations. To simplify this analysis, we parameterize the angle based on the number of slices n where the angular width is $360°/n$. We specify the stride to match the angle such that each inference performs inference on non-overlapping slices; overlapping slices is

[5] The final convolutional layer corresponds to the third convolutional layer in PointPillars [32] after three strided convolutions. When computing the feature pyramid, note that the output of the LSTM is only added the corresponding outputs of the third convolutional layer.

[6] Code available at `http://github.com/tensorflow/lingvo`

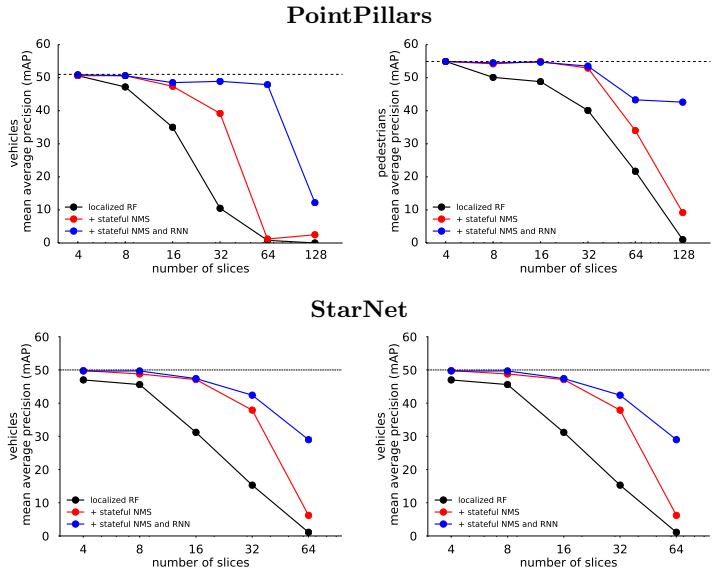


Fig. 3: **Streaming object detection may achieve comparable performance to a non-streaming baseline.** Mean average precision (mAP) versus the number of slices n within a single rotation for (left) vehicles and (right) pedestrians for (top) PointPillars [32] and (bottom) StarNet [43]. The solid black line (localized rf) corresponds to the modified baseline that operates on a spatially restricted region. Dashed lines corresponds to baseline models that processes the entire laser spin (vehicle = 51.0%; pedestrian = 54.9%). Each curve represents a streaming architecture (see text).

an option, but requires recomputing features on the same points. We compare the performance of the streaming models against the baselines in Figure 3.

As the number of slices n grows, the streaming model receives a decreasing fraction of the scene and the predictive performance monotonically decreases (black solid line). This reduction in accuracy may be expected because less sensory data is available for each inference operation. All objects sizes appear to be severely degraded by the localized receptive field, although there is a slight negative trend for increasing sizes (Table 1). The negative trend is consistent with the observation that vehicle mAP drops off faster than pedestrians, as vehicles are larger and will more frequently cross slices boundaries. For instance, at n=16, the mean average precision is 35.0% and 48.8% for vehicles and pedestrians, respectively. A large number of slices n=128 severely degrades model performance for both types.

Beyond only observing fewer points with smaller slices, we observe that even at a modest 8 slices the mAP drops compared to the baseline for both pedestrians and vehicles. Our hypothesis is that when NMS only operates per slice, there are

	0-5°	5-15°	15-25°	25-35°	>35°
baseline	12.6	30.9	47.2	69.1	83.4
localized RF	2.6	6.6	9.2	13.1	14.0
	-79%	-79%	-80%	-81%	-83%
+stateful NMS	9.0	24.1	36.0	54.9	68.0
	-28%	-22%	-23%	-20%	-18%
+stateful NMS and RNN	10.8	29.9	46.0	65.3	82.1
	-14%	-3%	-3%	-6%	-2%

Table 1: **Localized receptive field leads to duplicate detections.** Vehicle detection performance (mAP) across subtended angle of ground truth objects for localized receptive field and stateful NMS (n=32 slices) for [32]. Red indicates the percent drop from baseline.

slices		localized	global	stateful
16	car	35.0	47.5	47.4
	ped	48.8	54.8	54.9
32	car	10.5	39.2	39.0
	ped	40.1	53.1	52.9

Table 2: **Stateful NMS achieves comparable performance gains as global NMS.** Table entries report the mAP for detection on vehicles (car) and pedestrians (ped) [32]. Localized indicates the mAP for a spatially restricted receptive field. Global NMS boosts performance significantly but is a non-streaming heuristic. Stateful NMS achieves comparable results but is amenable to a streaming architecture.

false positives or false negatives created on any object that crosses a border of two consecutive slices. As a result, we next turn to investigating ways in which we can improve NMS in the intra-frame scenario to restore performance compared to the baseline while retaining the streaming model's latency benefits.

4.2 Adding state to non-maximum suppression boosts performance

NMS provides a standard heuristic method to remove highly overlapping predictions [12,15]. With a spatially localized receptive field, NMS has no ability to suppress overlapping detections arising from distinct slices of the LiDAR spin.

We can verify this failure mode by modifying NMS to operate over the concatenation of all detections across *all* n slices in a complete rotation of the LiDAR. We term this operation *global* NMS. (Note that global NMS does not enable a streaming detection system because a complete LiDAR rotation is required to finish inference.) We compute the detection accuracy for global NMS as a point of reference to measure how many of the failures of a spatially localized

receptive field can be rescued. Indeed, Table 2 (localized vs. global) indicates that applying global NMS improves predictive performance significantly.

We wish to develop a new form of NMS that achieves the same performance as global NMS but may operate in a streaming fashion. We construct a simple form of *stateful* NMS that stores detections from the previous slice of the LiDAR scene and use the previous slice's detections to rule out overlapping detections (Section 3.3). Stateful NMS does not require a complete LiDAR rotation and may operate in a streaming fashion. Table 2 (global vs. stateful) indicates that stateful NMS provides predictive performance that is comparable to a global NMS operation within ± 0.1 mAP. Indeed, stateful NMS boosts performance of the spatially restricted receptive field across all ranges of slices for both baseline models (Figure 3, red curve). This observation is also consistent with the fact that a large fraction of the failures across object size are largely systematically recovered (Table 1), suggesting that indeed duplicate detections across boundaries hamper model performance. These results suggest that this generic change to introduce state into the NMS heuristic may recover much of the performance drop due to a spatially localized receptive field.

4.3 Adding a learned recurrent model further boosts performance

Adding stateful NMS substantially improves streaming detection performance, however, the network backbone and learned components of the system only operate on the current slice of the scene and the outputs of the previous slice.

This section examines how a recurrent model may improve predictive performance by providing access to (1) more than just the previous LiDAR slice, and (2) the lower-level features of the model, instead of just the outputs. In particular, we provide an alteration to the meta-architecture by adding a recurrent layer to the global representation of the network featurization (Figure 2). We investigated variations of recurrent architectures and hyper-parameters, and settled on a simple single layer LSTM, but most RNN's produces similar results.

The blue curve of Figure 3 shows the results of adding a recurrent bottleneck on top of a spatially localized receptive field and a stateful NMS for both baseline architectures. Generically, we observe that predictive performance increases across the entire range of n. The performance gains are more notable across increasing number of slices n, as well as all vehicle object sizes (Table 1). These results are expected given that we observed systematic failure introduced for large objects that subtend multiple slices (Figure 3). For instance, with PointPillars at $n = 32$ slices, the mAP for vehicles is boosted from 39.2% to 48.9% through the addition of a learned, recurrent architecture (Figure 3, blue curves). These results also demonstrate how objects close to the scanner may be divided across slices, thus leading to higher error, since the object size can be approximately measured by the angle subtended in the θ direction. Specifically, our streaming model only introduces a 1.3 mAP drop for close-by objects (> 35 degrees), and 1.8 drop in mAP for faraway objects (< 5 degrees). Overall, this streaming model is only marginally lower than the mAP of the original model that has access to the complete LiDAR scan.

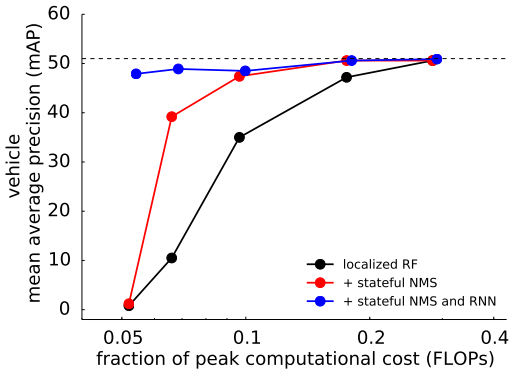


Fig. 4: Fraction of peak computational demand (FLOPs) versus detection accuracy (vehicle mAP) across varying number of slices n for [32]. Note the logarithmic scale on the x-axis. Each curve represents streaming architecture (see text). Dashed line indicates non-streaming baseline.

Although a streaming model sacrifices slightly in terms of predictive performance, we expect that the resulting streaming model would realize substantial gains in terms of computational demand and latency. In the following section we explore this question in detail.

4.4 Streaming detection reduces latency and peak computation

We define the **end-to-end** latency as the duration of time between the earliest observation of an object (i.e. the earliest time at which reflecting LiDAR points are received) and the identification of a localized, labeled object. For a non-streaming model the latency corresponds to the summation of the time for a complete (worst-case) 360^o rotation and the subsequent inference operation on the complete LiDAR scene. The latency in a streaming detection system should be improved for two reasons: (1) the computational demand for processing a fraction of the LiDAR spin should be roughly $\frac{1}{n}$ of the complete scene and (2) the inference operation does not require artificially waiting for the full LiDAR spin and may be pipelined. In this section, we focus our analysis on [32] because of previously reported latencies.

To test the first point, we compute the theoretical peak computational demand (in FLOPS) for running single frame inference on the baseline model as well as streaming models. Figure 4 compares the peak FLOPS versus the detection accuracy across varying slices n for each of the 3 streaming architectures. We display the compute of each approach in terms of the *fraction* of peak FLOPS required for the non-streaming baseline model (Note the log x-axis). We observe that a model with a localized receptive field (black curve) reveals a trade-off

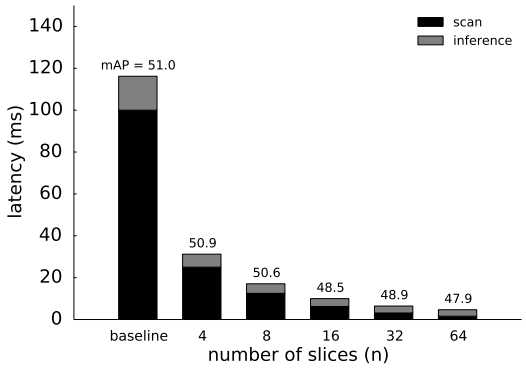


Fig. 5: Worst-case latency from the initial measurement of the vehicle to the detection for non-streaming (baseline) and streaming detection model (stateful NMS and RNN), broken down by phase. Scan phase latency is based on 10 Hz LiDAR period [53]. Inference phase latency estimated from baseline GPU implementation [32]. Numbers on top of bar are vehicle mAP.

in accuracy versus the amount of computational demand. However, subsequent models with stateful NMS (red curve) and stateful NMS and RNN (blue curve) require much fewer peak FLOPS to achieve most of the detection performance of the baseline. Furthermore, the stateful NMS and RNN achieves nearly baseline predictive performance across a wide range of slices n with a computational cost of roughly $\frac{1}{n}$. Thus, the streaming model requires less peak computational demand with minimal degradation in predictive performance.

Note that latency is very heavily determined by the hardware, as well as the rotational period of the LiDAR system. To estimate reasonable speed up gains, we test these ideas on a previously reported implementation speed for the PointPillars model [32] (Section 6), and employ the rotational period of the Waymo Open Dataset (i.e. 100 ms for 10 Hz) [53]. Figure 5 plots the latency versus the detection accuracy across the streaming model variants; we assume that the scan time is equivalent to the worst case delay between the first measurement of the object and the triggering of inference (the end of the slice): e.g., with 4 slices and 10Hz period, each slice triggers inference every 25ms. We estimate inference latency by scaling the floating point computation time [32] by the fraction of point cloud scene $\frac{1}{n}$ observed within an angular wedge necessary for inference.

The streaming models reduce end-to-end latency significantly in comparison to a non-streaming baseline model. In fact, we observe that for $n = 8$, a streaming model with a stateful NMS and RNN achieves competitive accuracy with the non-streaming model, but with $\frac{1}{15}$ of the latency (17ms vs 116ms). We take these results as a conceptual proof that a streaming detection system may significantly reduce end-to-end latency without significantly sacrificing predictive accuracy.

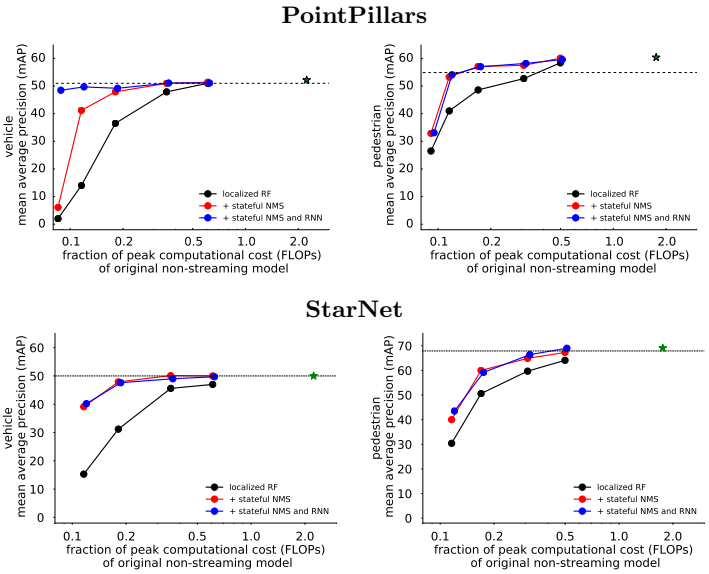


Fig. 6: **A larger streaming model may exceed the baseline performance, but with a fraction of the peak computational budget.** Results presented for vehicle (left) and pedestrian (right) detection on the larger (top) PointPillars [32] and (bottom) StarNet [43] relative to the original non-streaming baseline. Green star indicates the relative peak FLOPS and accuracy of the larger non-streaming model. Computational cost varies inversely with the number of slices(n) from 4 to 64.

4.5 Increasing model size exceeds baseline with fewer FLOPs

This section explores whether the computational resources that are freed up by switching to streaming settings may instead be harnessed (with a still reduced latency) to boost performance above and beyond the baseline model. Broadly speaking, we attempt to double the computational cost of each baseline network in order to attempt to boost the overall performance. For instance, for PointPillars we increase the size of the feature pyramid by systematically increasing the spatial resolution of the activation map. Specifically, we increase the top-down view grid size from 384 to 512 for pedestrian models, and from 512 to 784 for vehicle models, resulting in models with 1.75× and 2.23× relative expense, respectively. For StarNet, we increase all hidden units by 1.44×.

Figure 6 shows the predictive accuracy of the resulting both streaming baseline models on vehicles (left) and pedestrians (right) across slices n in comparison to the non-streaming baseline for both architectures. As a point of reference, we show the relative peak computational cost of these larger models when run in a non-streaming node (green star). The x-axis measures logarithmically the peak computational demand of the resulting model expressed as a fraction of the non-

streaming baseline model. Importantly, we observe that even though the peak computational demand is a small fraction of the non-streaming baseline model, the predictive performance matches or exceeds the non-streaming baseline model (e.g. 60.1 mAP versus 54.9 mAP for pedestrians with PointPillars). Note that in order to achieve these gains in the non-streaming model requires increasing the peak computational cost by 2.25× (green star). Moreover, at the lower peak FLOPS count, the stateful NMS and RNN model retains the most accuracy of the original baseline, yet may achieve $> 3\times$ latency gains. We take these results to indicate that much opportunity exists for further improving streaming models to well exceed a non-streaming model, while maintaining significantly reduced latency compared to a non-streaming architecture.

5 Discussion

In this work, we have described streaming object detection for point clouds for self-driving car perceptions systems. Such a problem offers an opportunity for blending ideas from object detection [49,14,39,48], tracking [37,38,11] and sequential modeling [56]. Streaming object detection offers the opportunity to detect objects in a SDC environment that significantly minimizes latency (e.g. 3-15×) and better utilizes limited computational resources.

We find that restricting the receptive field, adding temporal state to the NMS, and adding recurrence provide competitive accuracy on a large-scale self-driving dataset. The resulting system achieves favorable computational performance ($\sim 1/10^{th}$) and improved expected latency ($\sim 1/15^{th}$) with respect to a baseline non-streaming system. Such gains provide headroom to scale up the system to surpass baseline performance (60.1 vs 54.9 mAP) while maintaining a peak computational budget far below a non-streaming model.

This work offers opportunity for further improving this methodology, or application to new streaming sensors (e.g. high-resolution cameras that emit rasterized data in slices). While this work focuses on streaming models for a single frame, it is possible to also extend the models to incorporate data across multiple frames. We note that a streaming model may be amendable towards tracking problems since it already incorporates state. Finally, we have explored meta-architecture changes with respect to two competitive object detection baselines [32,43]. We hope our work will encourage further research on other point cloud based perception systems to test their efficacy in a streaming setting [60,40,59,32,42,64,58,43].

Acknowledgements

We thank the larger teams at Google Brain and Waymo for their help and support. We also thank Chen Wu, Pieter-jan Kindermans, Matthieu Devin and Junhua Mao for detailed comments on the project and manuscript.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). pp. 265–283 (2016) 7
2. Ackerman, E.: Lidar that will make self-driving cars affordable [news]. IEEE Spectrum **53**(10), 14–14 (2016) 2
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019) 1, 4
4. Chai, Y.: Patchwork: A patch-wise attention network for efficient object detection and segmentation in video streams. In: IEEE Conference on Computer Vision and Pattern Recognition (2019) 3
5. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017) 2
6. Chiu, C.C., Sainath, T.N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R.J., Rao, K., Gonina, E., et al.: State-of-the-art speech recognition with sequence-to-sequence models. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4774–4778. IEEE (2018) 3
7. Cho, H., Seo, Y.W., Kumar, B.V., Rajkumar, R.R.: 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). pp. 1836–1843. IEEE (2014) 1, 2, 4
8. Dean, T., Ruzon, M.A., Segal, M., Shlens, J., Vijayanarasimhan, S., Yagnik, J.: Fast, accurate detection of 100,000 object classes on a single machine. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1814–1821 (2013) 3
9. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2758–2766 (2015) 2
10. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2), 303–338 (2010) 3
11. Feichtenhofer, C., Pinz, A., Zisserman, A.: Detect to track and track to detect. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3038–3046 (2017) 14
12. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE transactions on pattern analysis and machine intelligence **32**(9), 1627–1645 (2010) 3, 9
13. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013) 1, 2, 4
14. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015) 3, 14

15. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014) 3, 6, 9
16. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256 (2010) 19
17. Graham, B., van der Maaten, L.: Submanifold sparse convolutional networks. CoRR **abs/1706.01307** (2017), http://arxiv.org/abs/1706.01307 6
18. Graves, A.: Sequence transduction with recurrent neural networks. arXiv preprint arXiv:1211.3711 (2012) 3
19. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017) 2
20. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015) 20
21. Hecht, J.: Lidar for self-driving cars. Optics and Photonics News **29**(1), 26–33 (2018) 2
22. Henriques, J.F., Vedaldi, A.: Mapnet: An allocentric spatial memory for mapping environments. In: IEEE Conference on Computer Vision and Pattern Recognition (2018) 3
23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997) 6
24. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7310–7311 (2017) 3
25. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2462–2470 (2017) 2
26. Jaitly, N., Sussillo, D., Le, Q.V., Vinyals, O., Sutskever, I., Bengio, S.: A neural transducer. arXiv preprint arXiv:1511.04868 (2015) 3
27. Jeon, H.H., Ko, Y.H.: Lidar data interpolation algorithm for visual odometry based on 3d-2d motion estimation. In: 2018 International Conference on Electronics, Information, and Communication (ICEIC). pp. 1–2. IEEE (2018) 2
28. Kim, J., Kim, H., Lakshmanan, K., Rajkumar, R.R.: 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. pp. 31–40. ACM (2013) 2
29. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 7, 19, 20
30. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009) 3
31. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (2012) 3
32. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. arXiv preprint arXiv:1812.05784 (2018) 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 19

33. Lim, K.L., Drage, T., Bräunl, T.: Implementation of semantic segmentation for road and lane detection on an autonomous ground vehicle with lidar. In: 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). pp. 429–434. IEEE (2017) 2
34. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017) 5
35. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014) 3
36. Lindner, P., Richter, E., Wanielik, G., Takagi, K., Isogai, A.: Multi-channel lidar processing for lane detection and estimation. In: 2009 12th International IEEE Conference on Intelligent Transportation Systems. pp. 1–6. IEEE (2009) 2
37. Liu, M., Zhu, M.: Mobile video object detection with temporally-aware feature maps. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5686–5695 (2018) 3, 7, 14
38. Liu, M., Zhu, M., White, M., Li, Y., Kalenichenko, D.: Looking fast and slow: Memory-guided mobile video object detection. arXiv preprint arXiv:1903.10172 (2019) 3, 7, 14
39. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016) 2, 3, 14
40. Luo, W., Yang, B., Urtasun, R.: 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. pp. 3569–3577 (2018) 4, 7, 14
41. McIntosh, L., Maheswaranathan, N., Sussillo, D., Shlens, J.: Recurrent segmentation for variable computational budgets. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 1648–1657 (2018) 3, 7
42. Meyer, G.P., Laddha, A., Kee, E., Vallespi-Gonzalez, C., Wellington, C.K.: Lasernet: An efficient probabilistic 3d object detector for autonomous driving. arXiv preprint arXiv:1903.08701 (2019) 4, 7, 14
43. Ngiam, J., Caine, B., Han, W., Yang, B., Chai, Y., Sun, P., Zhou, Y., Yi, X., Alsharif, O., Nguyen, P., et al.: Starnet: Targeted computation for object detection in point clouds. arXiv preprint arXiv:1908.11069 (2019) 3, 4, 5, 6, 7, 8, 13, 14, 20
44. Pinheiro, P., Collobert, R.: Recurrent convolutional neural networks for scene labeling. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 82–90. PMLR, Bejing, China (22–24 Jun 2014) 4, 7
45. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 918–927 (2018) 4
46. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017) 4, 5
47. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems. pp. 5099–5108 (2017) 4, 5

48. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016) 3, 14
49. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015) 2, 3, 14
50. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229 (2013) 3
51. Shen, J., Nguyen, P., Wu, Y., Chen, Z., Chen, M.X., Jia, Y., Kannan, A., Sainath, T., Cao, Y., Chiu, C.C., et al.: Lingvo: a modular and scalable framework for sequence-to-sequence modeling. arXiv preprint arXiv:1902.08295 (2019) 7
52. Shi, S., Wang, X., Li, H.: PointRCNN: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–779 (2019) 4
53. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 2, 4, 7, 12
54. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., et al.: Stanley: The robot that won the darpa grand challenge. Journal of field Robotics **23**(9), 661–692 (2006) 1, 2, 4
55. Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. International journal of computer vision **104**(2), 154–171 (2013) 3
56. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016) 4, 7, 14
57. Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems. pp. 802–810 (2015) 4, 7
58. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018) 2, 4, 6, 7, 14
59. Yang, B., Liang, M., Urtasun, R.: Hdnet: Exploiting HD maps for 3d object detection. In: Conference on Robot Learning. pp. 146–155 (2018) 2, 4, 7, 14
60. Yang, B., Luo, W., Urtasun, R.: Pixor: Real-time 3d object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7652–7660 (2018) 2, 4, 7, 14
61. Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: Ipod: Intensive point-based object detector for point cloud. arXiv preprint arXiv:1812.05276 (2018) 4
62. Zhang, J., Singh, S.: Visual-lidar odometry and mapping: Low-drift, robust, and fast. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). pp. 2174–2181. IEEE (2015) 2
63. Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., Vasudevan, V.: End-to-end multi-view fusion for 3d object detection in lidar point clouds. In: Conference on Robot Learning (CoRL) (2019) 4, 7
64. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4490–4499 (2018) 2, 4, 5, 7, 14

Appendix

A Architecture and Training Details

A.1 Training data

There are actually two different training settings in the paper: (1) for the non-streaming models and (2) for the streaming models with stateful NMS. In the first settings, there is no RNN so we use the original full scene data. In the second setting, we only use sliced data to train the streaming models with LSTM.

Operation	Stride	# In	# Out	Activation	Other
Streaming detector					
Featurizer MLP		12	64		With max pooling
Convolution block	1	64	64	ReLU	Layers=4
Convolution block	2	64	128	ReLU	Layers=6
Convolution block	2	128	256	ReLU	Layers=6
LSTM		256	256		Hidden=128
Deconvolution 1	1	64	128	ReLU	
Deconvolution 2	2	128	128	ReLU	
Deconvolution 3	4	256	128	ReLU	
Convolution/Detector	1	384	16		Kernel=3×3
Convolution block (S, C_{in}, C_{out}, L)					
Convolution 1	S	C_{in}	C_{out}	ReLU	Kernel=3×3
Convolution $2, ..., L-1$	1	C_{out}	C_{out}	ReLU	Kernel=3×3
Normalization	Batch normalization before ReLU for every convolution and deconvolution layer				
Optimizer	Adam [29] ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$)				
Parameter updates	40,000 - 80,000				
Batch size	64				
Weight initialization	Xavier-Glorot[16]				

Table 3: PointPillars detection baseline [32].

Operation	# In	# Out	Activation	Other
Streaming detector				
Linear	4	64	ReLU	
StarNet block × 5	64	64	ReLU	Final feature is the concat of all layers
LSTM	384	384		Hidden=128
Detector	384	16		
StarNet block				
Max-Concat	64	128		
Linear	128	256	ReLU	
Linear	256	64	ReLU	
Normalization	Batch normalization before ReLU			
Optimizer	Adam [29] ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$)			
Parameter updates	100,000			
Batch size	64			
Weight initialization	Kaiming Uniform [20]			

Table 4: StarNet detection baseline [43].