3D-MAN: 3D Multi-frame Attention Network for Object Detection

Zetong Yang[1]* Yin Zhou[2] Zhifeng Chen[3] Jiquan Ngiam[3]
[1]The Chinese University of Hong Kong [2]Waymo LLC [3]Google Research, Brain Team
tomztyang@gmail.com yinzhou@waymo.com {zhifengc, jngiam}@google.com

Abstract

3D object detection is an important module in autonomous driving and robotics. However, many existing methods focus on using single frames to perform 3D detection, and do not fully utilize information from multiple frames. In this paper, we present 3D-MAN: a 3D multi-frame attention network that effectively aggregates features from multiple perspectives and achieves state-of-the-art performance on Waymo Open Dataset. 3D-MAN first uses a novel fast single-frame detector to produce box proposals. The box proposals and their corresponding feature maps are then stored in a memory bank. We design a multi-view alignment and aggregation module, using attention networks, to extract and aggregate the temporal features stored in the memory bank. This effectively combines the features coming from different perspectives of the scene. We demonstrate the effectiveness of our approach on the large-scale complex Waymo Open Dataset, achieving state-of-the-art results compared to published single-frame and multi-frame methods.

1. Introduction

3D object detection is an important problem in computer vision as it is widely used in applications, such as autonomous driving and robotics. Autonomous driving platforms require precise 3D detection to build an accurate representation of the world, which is in turn used in downstream models that make critical driving decisions.

LiDAR provides a high-resolution accurate 3D view of the world. However, at any point of time, the LiDAR sensor collects only a single perspective of the scene. It is often the case that the LiDAR points detected on an observed object correspond to only a partial view of it. Detecting these partially visible instances is an ill-posed problem because there exist multiple reasonable predictions (shown as red and blue boxes in the upper row of Figure 1). These potential ambiguous scenarios can be a bottleneck for single-frame 3D detectors (Table 1).

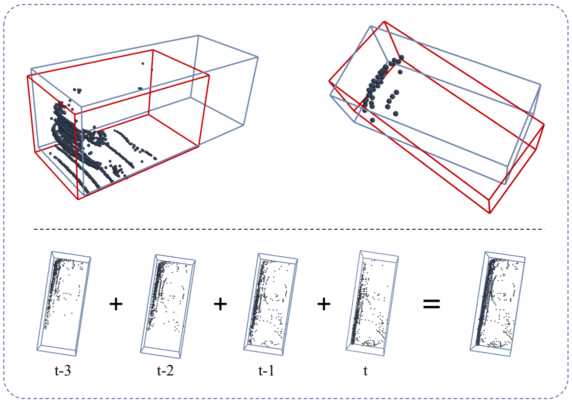

Figure 1. Upper row: Potential detections given LiDAR from a single frame demonstrating ambiguity between many reasonable predictions. Lower row: After merging the points aligned across 4 frames, there is more certainty for the correct box prediction.

IoU threshold	0.3	0.5	0.7
AP (%)	94.72	88.97	63.27

Table 1. We vary the intersection-over-union (IoU) threshold for considering a predicted box correctly matched to a ground-truth box, and measure the performance of the PointPillars model on the Waymo Open Dataset's validation set. A lower IoU threshold corresponds to allowing less accurate boxes to match. This shows that improving the box localization could significantly improve model performance.

In the autonomous driving scenario, as the vehicle progresses, the sensors pick up multiple views of the world, making it possible to resolve the aforementioned localization ambiguity. Multiple frames across time can provide different perspectives of an observed object instance. An effective multi-frame detection method should be able to extract relevant features from each frame and aggregate them, so as to obtain a representation that combines multiple perspectives (Figure 1). Research in 3D multi-frame detection has been limited due to a lack of available datasets with well-calibrated multi-frame data. Fortunately, recently released large-scale 3D sequence datasets (NuScenes [2], Waymo Open Dataset [23]) have made such data available.

A straight-forward approach to fusing multi-frame point

*Work done during an internship at Google Brain.

Model	Stationary (%)	Slow (%)	Medium (%)	Fast (%)
1-frame	60.01	66.64	65.02	71.90
4-frames	62.4	67.39	**66.68**	**77.99**
8-frames	**63.7**	**67.98**	66.29	72.30

Table 2. Velocity breakdowns of vehicle AP metrics for PointPillars models using point concatenation. For the 8-frame model, we find that its benefits come from slow-moving vehicles. Fast-moving objects no longer benefit from a large number of frames since the LiDAR points are no longer aligned across the frames.

clouds is to use point concatenation, which simply combines points across different frames together [2]. The combined point cloud is then used as input to a single-frame detector. This approach works well for static and slow-moving objects since the limited movement implies that the LiDAR points will be mostly aligned across the frames. However, when objects are fast-moving or when longer time horizons are considered, this approach may not be as effective since the LiDAR points are no longer aligned (Table 2).

As an alternative to point concatenation, Fast-and-furious [14] attempts to fuse information across frames by concatenating at a feature map level. However, this still runs into the same challenge with misaligned feature maps for fast-moving objects and longer time horizons. Recent approaches [10, 34] propose using recurrent layers such as Conv-LSTM or Conv-GRU to aggregate the information across frames. It turns out that these recurrent approaches are often computationally expensive.

Our Approach. We propose 3D-MAN: a 3D multi-frame attention network that is able to extract relevant features from past frames and aggregate them effectively. 3D-MAN has three components: (i) a fast single-frame detector, (ii) a memory bank, and (iii) a multi-view alignment and aggregation module.

The fast single-frame detector (FSD) is an anchor-free one-stage detector with a novel learning strategy. We show that a max-pooling based non-maximum suppression (NMS) algorithm together with a novel Hungarian-matching based loss is an effective method to generate high-quality proposals at real-time speeds. These proposals and the last feature map from FSD are then fed into a memory bank. The memory bank stores both predicted proposals and feature maps in previous frames so as to maintain different perspectives for each instance across frames.

The stored proposals and features in the memory bank are finally fused together through the multi-view alignment and aggregation module (MVAA), which produces fused multi-view features for target proposals that are used to regress bounding boxes for final predictions. MVAA has two stages: a multi-view *alignment* stage followed by a multi-view *aggregation* stage. The alignment stage works on each stored frame independently; it uses target proposals as queries into a stored frame to extract relevant features. The aggregation stage then merges across frames for each target proposal independently. This can be viewed as a form of factorization over the attention across proposals and frames.

We evaluate our model on large-scale Waymo Open Dataset [23]. Experimental results demonstrate that our method outperforms published state-of-the-art single-frame methods and multi-frame methods. Our primary contributions are listed below.

Key Contributions.

- We propose 3D-MAN: a 3D multi-frame attention network for object detection. We demonstrate that our method achieves state-of-the-art performance on the Waymo Open Dataset [23] and provide thorough ablation studies.
- We introduce a novel training strategy for a fast single-frame detector method that uses max-pooling to perform non-maximum suppression and a variant of Hungarian matching to compute a detection loss.
- We design an efficient multi-view alignment and aggregation module to extract and aggregate relevant features from multiple frames in a memory bank. This module produces features containing information from multiple perspectives that perform well for classification and bounding box regression.

2. Related Work

3D Single-frame Object Detection. Current 3D object detectors can be categorized into three approaches: voxel-based methods, point-based methods, and their combination. First, voxel-based methods transform via voxelization a set of unordered points into a fixed-size 2D feature map, on which convolutional neural networks (CNN) can be applied to generate detection results. Traditional approaches for voxel feature extraction rely on hand-crafted statistical quantities or binary encoding [25, 29], while recent works show that machine-learned features demonstrate favorable performance [37, 12, 28, 36, 20, 26]. Second, point-based methods [32, 19, 31, 15, 33] address detection problems by directly extracting features based on the point cloud, without an explicit discretization step. Finally, recent works have combined methods from both voxel-based and point-based feature representations [18] by using the voxel-based methods to generate proposals and the point-based methods to refine them.

2D Multi-frame Object Detection. 2D multi-frame object detection has been widely explored compared to 3D counterparts. 2D detection methods primarily focus on aligning objects in a target frame using motion and appearance

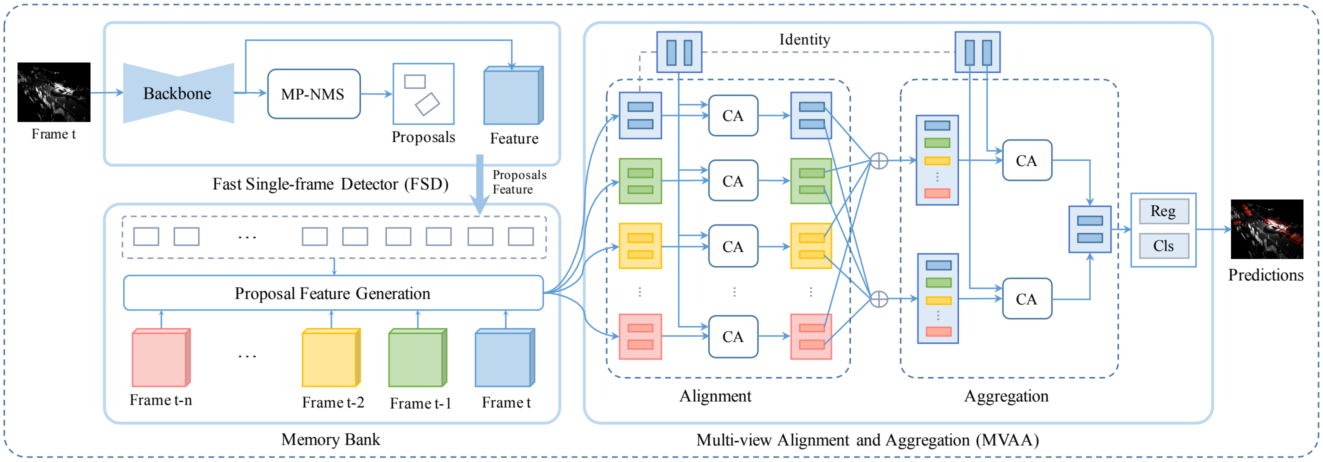

Figure 2. Framework for 3D-MAN: 3D multi-frame attention network. Given the point cloud for a target frame t, a fast single-frame detector first generates box proposals. These proposals (box parameters) with the feature map (last layer of the backbone network) are inserted into a memory bank that stores proposals and features for the last n frames. We use a proposal feature generation module to extract proposal features for each stored frame. Each small rectangle box denotes a proposal and its associated features extracted in different frames. The multi-view alignment and aggregation module performs attention across proposal features from the memory bank, using the target frame as queries to extract features for classification and regression. "MP-NMS" and "CA" represent MaxPoolNMS and cross-attention respectively. During training, we use classification and regression losses applied to the FSD proposals (L_{fsd}), the final outputs of the MVAA network (L_{mvaa}), and the outputs of the alignment stage (L_{cv}, an auxiliary cross-view loss).

features from previous frames. Relational modules with self-attention layers [9] are prevalent among these methods [6, 4, 27, 5, 21]. They usually take as input a target frame and multiple reference frames, from which proposals are generated per frame. Relation modules are applied to aggregate temporal features for more robust object detection. Most approaches use self-attention across all proposals in all previous frames. In contrast, our method factorizes the attention layer to first operate independently across frames (alignment stage), and then independently across proposals (aggregation stage).

3D Multi-frame Object Detection. A straight-forward approach to multi-frame detection is to concatenate the points from different frames together [2]. This has been demonstrated on the NuScenes dataset (improvement of 21.9% to 28.8% mAP [2]), and we also observe improvements in our experiments (Table 2). However, as we increase the number of frames concatenated, the improvement diminishes since the LiDAR points are less likely to be aligned across longer time horizons (Table 2). Fast-and-furious [14] side steps aligning the points by instead concatenating the intermediate features maps. However, this approach may still result in misalignment across the feature maps for fast-moving objects and longer time horizons. Recent approaches [10, 34] show further performance improvement by applying Conv-LSTM or Conv-GRU to fuse multi-frame information. However, the use of a single memory state that gets updated creates a potential bottleneck, and the high resolution of the feature maps make these methods computationally expensive.

3. 3D-MAN Framework

The 3D-MAN framework (Figure 2) consists of 3 components: (i) a fast single-frame detector (FSD) for producing proposals given input point clouds, (ii) a memory bank to store features from different frames and (iii) a multi-view alignment and aggregation module (MVAA) for combining information across frames to generate final predictions.

3.1. Fast Single-frame Detector

Anchor-free Point Pillars. We base our single-frame detector on the PointPillars architecture [12] with dynamic voxelization [36]. We start by dividing the 3D space into equally distributed pillars which are voxels of infinite height. Each point in the point cloud is assigned to a single pillar. Each pillar is then featurized using a PointNet [17] producing a 2D feature representation for the entire scene, which is subsequently processed through a CNN backbone. Each location of the final layer of the network produces a prediction for a bounding box relative to the corresponding pillar center. We regress the location residuals, bounding box sizes, and orientation. A binning approach is used for predicting orientation which first classifies the orientation into one bin followed by regression of the residual from the corresponding bin center [16, 19].

Non-maximum suppression (NMS) is often used to post-process the detections produced by the last layer of the network for redundancy removal. It first outputs the highest scoring box and then suppresses all overlapping boxes with that box, repeating this process until all boxes are processed. However, the sequential nature of this algorithm makes it slow to run in practice when there is a large number of predictions. We use a variant of NMS that leverages max-pooling to speed up this process. MaxPoolNMS [35] uses the max pooling operation to find local peaks on the objectness score map. The local peaks are kept as predictions, while all other locations are suppressed. This process is fast and highly parallelizable. We find that this approach can be up to 6× faster[1] than regular NMS when dealing with about 200k predictions.

Hungarian Matching. MaxPoolNMS is usually performed using the classification score as the ranking signal to indicate that one box is better than another. However, the classification score is a proxy metric: ideally, we want to have the highest scoring box to be the best localized box. The ideal score map should have a single peak which corresponds to the best localized box. We propose using the Hungarian matching algorithm [3, 22] to produce such a score map.

Given a set of bounding box predictions and a set of ground-truth boxes, we compute the IoU score for each pair of them. By applying the Hungarian matching algorithm to this matrix[2] of pair-wise scores, we can obtain a single match for each ground-truth box to a predicted box that maximizes the overall matching score. For each ground-truth box, we treat the matched predicted box as positive, and all unmatched boxes as negative. In this way, the model is encouraged to predict only one positive box per ground-truth box such that the box predicted corresponds to the highest IoU-scoring box.

It turns out that there are two challenges when using the Hungarian matching algorithm. First, the Hungarian matching algorithm is of order $O(n^3)$ and can be slow if there are a large number of predictions. Therefore, we choose to perform the Hungarian matching based-loss only after the MaxPoolNMS step. This ensures that only a few predictions remain, and enables the matching algorithm to complete quickly.

Second, the model can end up in a bad local minima by only predicting boxes which are far away from any ground-truth box (e.g., predicting boxes in locations where there are no points in the input point clouds). Consequently, these ground-truth boxes do not overlap at all with their matched prediction boxes. As a result, the model does not get any meaningful learning signals from these matches and is not able to converge to a good solution. To address this issue, we post-process the matches to reassign ground-truth boxes that have no overlap with their matched prediction box. We assign them instead to their closest pillar in the feature map, which may not be one retained by MaxPoolNMS. This encourages the model to avoid invalid assignments and converge well.

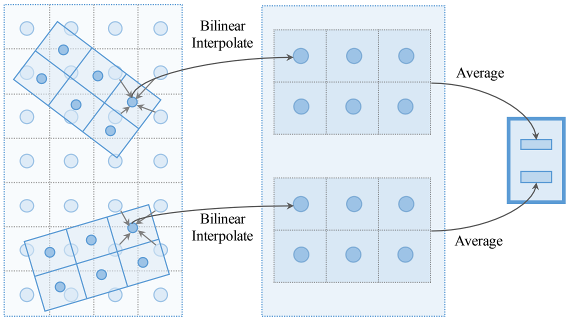

Figure 3. Illustration of rotated ROI feature extraction [13]. We first identify key points in each proposal box and then extract features using bilinear interpolation. Averaging pooling is further used to summarize each box into a single feature vector. Note that while the figure denotes key points over 3 × 2 locations, we use 7 × 7 for vehicles and 3 × 3 for pedestrians.

3.2. Memory Bank

Memory Bank. We use a memory bank to store the proposals and feature maps extracted by the FSD for the last n frames. When proposals and features from a new frame are added to the bank, those from the oldest frame are discarded.

Proposal Feature Generation. To obtain features from multiple perspectives, we propose to generate proposal features for each stored frame in the memory bank as well as the target frame. We find that it is useful to use all stored proposals regardless of which frame the proposal comes from to extract features from every stored frame. This allows the model to increase its recall since an object may be missed by FSD in a single frame because of occlusion or partial observation.

For each proposal, we extract its features using a rotated ROI feature extraction approach (Figure 3) [13]. Given a proposal, we identify $K \times K \times 1$ equally distributed key points with respect to the proposal box[3]. For each key point, we compute a feature by bilinear interpolation of its value in the feature map. Finally, we use average pooling across all the $K \times K \times 1$ key points to obtain a single feature vector for the proposal. It is worth noting that this feature extraction method can be performed without correcting the entire LiDAR point cloud for ego-motion of the autonomous vehicle. This facilitates deployment in a production autonomous driving system.

[1] Execution time for regular NMS depends on the number of output boxes desired, while MaxPoolNMS's speed is invariant the number of output boxes.

[2] In practice, we add dummy boxes to the ground-truth boxes so that a one-to-one match is always produced.

[3] We use $7 \times 7 \times 1$ for vehicles and $3 \times 3 \times 1$ for pedestrians.

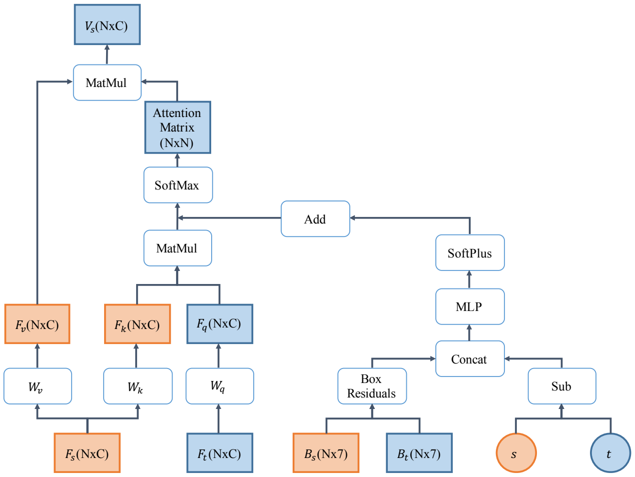

Figure 4. Cross-attention network in the multi-view alignment module. F_s and B_s represent features and box parameters of proposals in a stored frame while F_t and B_t are those for the target frame. We use s and t to denote the indices of the stored frame and target frame respectively. N and C stand for the number of proposals and channels. "Box residuals" produces a pairwise $N \times N \times 7$ tensor that encodes the differences in all pairs of boxes, using the same approach that is used to compute residuals for ground-truth boxes from anchor boxes [12]. V_s is the output of the cross-attention network, such that each input target box has one associated output feature vector with the corresponding stored frame.

The proposal features generated for the target frame will be used next in the MVAA module as the query features for the cross-attention networks, while proposal features for stored frames will be treated as keys and values.

3.3. Multi-view Alignment and Aggregation

These proposal features are then sent to the multi-view alignment and aggregation module (MVAA) to be extracted and aggregated. The alignment module is applied independently for each stored frame (attention is across boxes, performed separately for each frame), while the aggregation module is applied independently for each box (attention is across time). One can view this as a factorized form of attention.

Multi-view Alignment. Given a new frame's proposal, the multi-view alignment module is responsible for extracting its relevant information in each previous frame separately (Figure 2, MVAA-Alignment). To achieve this goal, the alignment stage has to figure out how to relate the identities of the proposals in the new frame to those in the stored frames. A naive approach could use nearest neighbor matching or maximum IoU overlap. However, when an instance is fast-moving or close to any other instance, there will often be ambiguity in the appropriate assignment. Furthermore, the naive approach does not learn interactions between the new proposal and other objects in previous frames that could provide contextual information.

We propose using a cross-attention network (Figure 4) to learn how to relate the new frame proposals to those of stored frames. This network could potentially learn to align the proposal identities and also model interactions across objects. Specifically, we apply projection layers to encode the new frame proposal features F_t as well as stored proposal features F_s so as to compute projected queries F_q, keys F_k and values F_v. These are used to compute an attention matrix. We further provide temporal and spatial information to the attention matrix through encoding the relative frame index and box residuals between all pairs of the query and stored boxes. The cross-attention network is applied between the target frame and each stored frame independently with shared parameters, generating a feature vector for each target proposal (V_s) from each stored frame.

Cross-view Loss. The alignment stage of MVAA is designed to extract features from each stored frame that are most relevant to each target proposal. To encourage the extracted features to be a relevant representation, we employ an auxiliary loss that encourages the extracted features to contain sufficient information to predict the corresponding ground-truth bounding box associated with the target proposal. Concretely, we add separate classification and regression heads that use each extracted feature vector of the alignment stage to predict the box residuals between target proposal and its corresponding ground-truth box.

Multi-view Aggregation. After the alignment module, each proposal in the target frame will have an associated feature for each stored frame. The multi-view aggregation layer (Figure 2, MVAA-Aggregation) is responsible for combining these features from different perspectives together to form a single feature for each proposal. Concretely, we use the new frame's proposal features as the attention query inputs, and its corresponding extracted features in previous frames as the keys and values.

We note that the aggregation module can enable the network to be robust to newly appearing objects. If an object appears for the first time in a new frame, the model can compute an attention matrix that will only focus on the new frame and ignore the past frames since they are not relevant.

Box Prediction Head. After MVAA, we have an updated feature for each proposal in the new frame. We regress objectness scores and box parameters from this feature representation. For the objectness score, we follow [18] and treat the IoU between proposals and their corresponding ground-truth bounding boxes as the classification target, with the sigmoid cross-entropy loss. The box parameter targets are encoded as residuals [12, 37] and trained with a smooth-$L1$ loss. The same formulations are used for the cross-view loss.

3.4. Losses

We minimize the total loss consisting of a fast single-frame detector (FSD) loss L_{fsd}, a multi-view prediction loss L_{mvaa}, and a cross-view loss L_{cv} with equal loss weights.

$$L_{total} = L_{fsd} + L_{mvaa} + L_{cv} \tag{1}$$

The same formulation for detection loss L_{det} is used in these three losses. This includes a objectness loss L_{obj} and a regression loss L_{reg}.

$$L_{det} = \frac{1}{|C|}\sum_{i \in C} L_{obj} + \frac{1}{|R|}\sum_{i \in R} L_{reg} \tag{2}$$

C represents the set of locations where we predict an objectness score. For L_{fsd}, this corresponds to the remaining pillars after MaxPoolNMS, while for L_{mvaa} and L_{cv}, this corresponds to the proposals after FSD (specifically, those that remain after MaxPoolNMS). For the objectness loss in L_{mvaa} and L_{cv}, we use the IoU overlap between the proposal and its assigned ground-truth as the target. For L_{fsd}, the output of the Hungarian matching is used to determine positive and negative assignments for the objectness loss.

R represents the set of locations which are associated with a ground-truth box. For all losses (L_{fsd}, L_{mvaa}, and L_{cv}), these are the matched boxes from Hungarian matching. For the regression losses, we use a smooth-$L1$ loss as the supervision of regressing the x, y, z center location residuals, and their corresponding dimensions. For orientation, we use a binning orientation loss [16, 19]. The model is expected to predict an angle bin first, followed by a residual from the bin center. We use 12 bins for L_{fsd} and 1 bin for L_{mvaa} and L_{cv}.

The cross-view identity loss L_{cv} is computed across all the outputs of the multi-view alignment stage, and averaged across all instances.

4. Experiments

We evaluate our method on Waymo Open Dataset [23], a large scale 3D object detection dataset. There are a total of 1150 sequences divided into 798 training, 202 validation, and 150 testing examples. Each sequence consists of about 200 frames at a frame rate of 10 Hz, where each frame includes a LiDAR point cloud and labeled 3D bounding boxes for vehicles, pedestrians, cyclists and signs. We evaluate our model and compare it with other methods using average precision (AP) and Average Precision Weighted by Heading (APH).

4.1. Implementation Details

Hyperparameters. Given the input point cloud in a target frame, we first set the detection range as $[-76.8m, 76.8m]$ for x and y axes and $[-2m, 4m]$ for the z-axis. We equally split this 3D range into $[512, 512]$ pillars among x and y axes respectively, following PointPillars [12]. For the MaxPoolNMS applied in FSD, we use a max-pooling kernel size of $[7, 7]$ for vehicles and $[3, 3]$ for pedestrians, with a stride of $[1, 1]$. After MaxPoolNMS, a set of 128 proposals per frame are passed to the memory bank.

Network Architectures. In our proposed FSD, we use the same backbone network illustrated in PointPillars [12]. The channel dimension C of the last feature map and proposal features is 384. For encoding the frame index and relative box residuals, we apply a 2-layer perceptron (MLP) networks with C output channels for the first layer, and 1 output for the second layer. These are used in the cross-attention layers of the MVAA module. In the prediction head, we first apply a 2-layer MLP network with C output channels to embed the aggregated multi-view features. These embeddings are transformed with two prediction branches for classification and regression.

Training Parameters. Our network is trained end-to-end using the ADAM [11] optimizer for a total number of 50 epochs with an initial learning rate of 0.0016 and a batch size of 32. We apply exponential decay to anneal the learning rate, starting at 5 epochs until 45 epochs. During training, we apply random flip and random rotation as our only data augmentation methods.

Utilizing a large number of frames. We enable 3D-MAN to exploit a large number of frames by combining it with point concatenation. Our best model uses 16 frames split into 4 windows of 4 frames. The point clouds in each window are concatenated together and used as input to the FSD. Each window thus becomes an entry in the memory bank, and the model is expected to produce predictions for only the last frame. This utilizes point concatenation for when movement is small with nearby frames and MVAA for large movement across a longer time range. We provide further ablation studies with varying sizes of input frames in Section 4.3.

4.2. Main Results

Waymo Validation Set. We compare our method with published state-of-the-art single-frame and multi-frame methods on the Waymo validation set on class Vehicle (Table 3) and class Pedestrian (Table 4). We first compare the performance between our model with and without multi-frame inputs (Table 3). When the model has access to 16 stored frames, the overall 3D AP (LEVEL_1) is improved by 5.50% on vehicles labeled as LEVEL_1 difficulty, illus-

Difficulty	Method	3D AP (IoU=0.7)				3D APH (IoU=0.7)			
		Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf
LEVEL_1	StarNet [15]	55.11	80.48	48.61	27.74	54.64	79.92	48.10	27.29
	PointPillars [12]	63.27	84.90	59.18	35.79	62.72	84.35	58.57	35.16
	MVF [36]	62.93	86.30	60.02	36.02	-	-	-	-
	AFDet [7]	63.69	87.38	62.19	29.27	-	-	-	-
	RCD [1]	68.95	87.22	66.53	44.53	68.52	86.82	66.07	43.97
	PV-RCNN [18]	70.30	91.92	69.21	42.17	69.49	91.34	68.53	41.31
	3D-MAN (Ours)	69.03	87.99	66.55	43.15	68.52	87.57	65.92	42.37
	PointPillars* [12]	65.41	85.58	61.51	39.51	64.88	85.02	60.95	38.91
	ConvLSTM* [10]	63.6	-	-	-	-	-	-	-
	3D-MAN* (Ours)	**74.53**	**92.19**	**72.77**	**51.66**	**74.03**	**91.76**	**72.15**	**51.02**
LEVEL_2	StarNet [15]	48.69	79.67	43.57	20.53	48.26	79.11	43.11	20.19
	PointPillars [12]	55.18	83.61	53.01	26.73	54.69	83.08	52.46	26.24
	PV-RCNN [18]	65.36	91.58	65.13	36.46	64.79	91.00	64.49	35.70
	3D-MAN (Ours)	60.16	87.10	59.27	32.69	59.71	86.68	58.71	32.08
	PointPillars* [12]	57.28	84.31	55.41	29.71	56.81	83.79	54.90	29.24
	3D-MAN* (Ours)	**67.61**	**92.00**	**67.20**	**41.38**	**67.14**	**91.57**	**66.62**	**40.84**

Table 3. 3D AP and APH Results on Waymo Open Dataset validation set for class Vehicle. *Methods utilize multi-frame point clouds for detection. We report PointPillars [12] based on our own implementation, with and without point concatenation. Difficulty levels are defined in the original dataset[23].

Method	LEVEL_1		LEVEL_2	
	3D AP	3D APH	3D AP	3D APH
StarNet [15]	68.32	60.89	59.32	52.76
PointPillars [12]	68.88	56.57	59.98	49.14
MVF [36]	65.33	-	-	-
3D-MAN (Ours)	**71.71**	**67.74**	**62.58**	**59.04**

Table 4. 3D AP and APH Results on Waymo Open Dataset *validation* set for class Pedestrian.

trating the effectiveness of our approach.

3D-MAN outperforms the current best published method (PV-RCNN [18]) by 3.56% (30-50m range) and 9.49% (>50m range) AP (LEVEL_1) on vehicles. At these further ranges, objects are often partially visible, where having more information from different perspectives could help. These improvements show that our model is able to effectively combine the information across multiple views to generate more accurate 3D predictions. Moreover, compared to existing multi-frame models, 3D-MAN also outperforms them by a large margin. Our method achieves a better 3D AP than the recently published Conv-LSTM method [10] by 10.93% on vehicle detection. For pedestrian detection, 3D-MAN also achieves the best performance (Table 4).

Waymo Testing Set. We also evaluate our model on Waymo testing set through a test server submission. For vehicle detection (Table 5), 3D-MAN achieves 78.71% AP and 78.28 APH, outperforming RCD [1] by 6.74% and 6.69% respectively, which is currently the best published method among results generated by a single model (not using any ensemble methods).

Method	Vehicle		Pedestrians	
	3D AP	3D APH	3D AP	3D APH
SECOND [28]	50.11	49.63	-	-
StarNet [15]	63.51	63.03	67.78	60.10
PointPillars [12]	68.62	68.08	67.96	55.53
SA-SSD [8]	70.24	69.54	57.14	48.82
RCD [1]	71.97	71.59	-	-
3D-MAN (Ours)	**78.71**	**78.28**	**69.97**	**65.98**

Table 5. 3D AP and APH Results on Waymo Open Dataset *testing* set for class Vehicle and Pedestrain among LEVEL_1 difficulty objects. Metric breakdowns for our model is available on the Waymo challenge leaderboard.

	Mask	Centeredness	Hungarian Matching
Ped. (%)	64.7	67.1	**70.2**
Veh. (%)	44.5	63.7	**64.8**

Table 6. Mini-validation AP comparison among different ground-truth assignment strategies using FSD for both Pedestrian and Vehicle classes.

4.3. Ablation Studies

We conduct all our ablation studies only for the vehicle class, and report LEVEL_1 difficulty results based on a subset of the full validation set. We created a mini-validation set by uniformly sampling 10% of the full validation set. This results in a dataset that allows us to experiment significantly faster. We note that there is a negligible performance gap between the mini-validation and full validation set: for example, our best model obtains 74.3% on the mini-validation set versus 74.5% on the full validation set.

Hungarian Matching. We compare the performance of using different assignment strategies in FSD, including the mask strategy, centeredness strategy and Hungarian matching strategy (Table 6). The mask strategy [24, 30] as-

Method	Baseline	Concat	Relation	MVAA
AP (%)	68.2	70.5	70.1	**72.5**

Table 7. Mini-validation AP comparison on class Vehicle among different multi-frame fusion approaches.

Supervision Method	None	Correspondence Loss	CV Loss
AP (%)	70.7	71.4	**72.5**

Table 8. Mini-validation AP comparison on class Vehicle using different auxiliary losses with the MVAA alignment stage.

signs interior pillars of any valid object positive and all other pillars negative. However, this can lead to a discrepancy between classification score and localization accuracy. Our experiments show that this performs the least well. Centeredness strategy [31, 24, 35] encourages pillars with closer distance to the instance center to have a higher classification score. However, the pillar in the center may not always draw the best localization prediction in point clouds: the LiDAR points often are on the surface of the vehicle and not in the interior. We find centeredness to perform better than mask, but worse than our proposed Hungarian matching approach. FSD achieves the highest AP with the Hungarian matching strategy, which validates our approach.

Multi-frame Approaches. We compare our method to other multi-frame approaches (Table 7), including the point concatenation approach and a self-attention approach across all previously detected boxes. Multi-frame models in the comparison have 4 frames as input and are expected to predict bounding boxes for only the last frame. We also perform ego-motion pose correction to map points from the earlier frames to the pose of the last frame.

The *Baseline* model is our *single-frame* two-stage model, which applies FSD to generate proposals with features and deploys box prediction head with an MLP network to refine these proposals. It achieves 68.2% AP on Vehicle and provides a baseline to compare the multi-frame models against.

In the *Concat* approach, points across all frames are combined together, and the merged point cloud is used as input to the *Baseline* model. This improves upon the baseline by 2.3% AP. The *Relation* approach first extracts box proposals from multiple frames and then uses a self-attention network on *all* past proposals directly to produce a prediction. This performs better than the *Baseline* but worse than the *Concat* model.

Our approach (MVAA) performs the best, outperforming the *Concat* approach and *Relation* approach by 2.0% and 2.4% respectively.

Cross-view loss. We find it useful to have an auxiliary cross-view loss to encourage the model to propagate relevant features in the alignment stage of MVAA. To evaluate the effectiveness of the cross-view loss, we compare it to not having an auxiliary loss and also an alternative auxiliary correspondence loss. The correspondence loss encourages elements of the attention matrix (of the alignment stage in MVAA) to be close to 1 if the query proposal matches the instance of the corresponding stored proposal, and zero otherwise. We compare these approaches for the auxiliary loss (Table 8), and find that using the cross-view loss outperforms having no auxiliary loss by 1.8% and using the correspondence loss by 1.1%.

Frames	1	4	7	10	13	16
AP (%)	68.2	72.5	73.4	73.5	73.8	74.3

Table 9. Mini-validation AP comparison for different number of input frames to the 3D-MAN model. All models are expected to predict only the last frame. Models with 7, 10, 13, and 16 frames use concatenated points (over windows of 4 frames) as input, with different amount of overlaps between adjacent windows.

Model	Stationary (%)	Slow (%)	Medium (%)	Fast (%)
4-frames	69.5	68.6	67.1	78.3
16-frames	**73.2**	**70.4**	**68.9**	**79.2**

Table 10. Velocity breakdowns of vehicle AP metrics for 3D-MAN with varying number of input frames.

Varying number of input frames. We further compare our model's performance on different number of available frames (Table 9). In order to draw a fair comparison between models with 7 through 16 frames, we fix the computation by using point concatenation over windows of 4 frames, with different degrees of overlaps between windows (similar to strides in convolution windows). We find that our model steadily improves as it has access to more input frames corresponding to longer time horizons.

Velocity breakdowns. We also compare our model's performance across different velocity breakdowns. Recall that the baseline multi-frame PointPillars model performance degrades when using 8-frames versus 4-frames (Table 2). Conversely, our model demonstrates an improvement when we increase the number of frames from 4 to 16 (Table 10). This shows that our approach is able to benefit fast-moving vehicles.

5. Conclusion

In this paper, we present a novel 3D object detection method, 3D-MAN, which utilizes attention networks to extract and aggregate features across multiple frames. We introduce a fast single-frame detector that utilizes a Hungarian matching strategy to align the objectness score with the best localized box. We show how the outputs of the single-frame detector can be used with a memory bank and a novel multi-view alignment and aggregation module to fuse the information from multiple frames together. Our method is effective across long time horizons and obtains state-of-the-art performance on a challenging large scale dataset.

References

[1] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. In *CoRL*, 2020. 7

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *CVPR*, 2020. 1, 2, 3

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, 2020. 4

[4] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection. In *CVPR*, 2020. 3

[5] Hanming Deng, Yang Hua, Tao Song, Zongpu Zhang, Zhengui Xue, Ruhui Ma, Neil Martin Robertson, and Haibing Guan. Object guided external memory network for video object detection. In *ICCV*, 2019. 3

[6] Jiajun Deng, Yingwei Pan, Ting Yao, Wengang Zhou, Houqiang Li, and Tao Mei. Relation distillation networks for video object detection. In *ICCV*, 2019. 3

[7] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. *CoRR*, 2020. 7

[8] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *CVPR*, June 2020. 7

[9] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018. 3

[10] 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. *CoRR*, 2020. 2, 3, 7

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 6

[12] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CVPR*, 2019. 2, 3, 5, 6, 7

[13] Ming Liang*, Bin Yang*, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019. 4

[14] 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 *CVPR*, 2018. 2, 3

[15] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, Zhifeng Chen, Jonathon Shlens, and Vijay Vasudevan. Starnet: Targeted computation for object detection in point clouds. *CoRR*, 2019. 2, 7

[16] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CVPR*, 2018. 3, 6

[17] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 3

[18] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020. 2, 5, 7

[19] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 2, 3, 6

[20] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-aˆ2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2019. 2

[21] Mykhailo Shvets, Wei Liu, and Alexander C. Berg. Leveraging long-range temporal relationships between proposals for video object detection. In *ICCV*, 2019. 3

[22] Russell Stewart, Mykhaylo Andriluka, and Andrew Y. Ng. End-to-end people detection in crowded scenes. In *CVPR*, 2016. 4

[23] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 1, 2, 6, 7

[24] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: fully convolutional one-stage object detection. *ICCV*, 2019. 7, 8

[25] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems XI*, 2015. 2

[26] Yue Wang, Alireza Fathi, Abhijit Kundu, David Ross, Caroline Pantofaru, Thomas Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In *ECCV*, 2020. 2

[27] Haiping Wu, Yuntao Chen, Naiyan Wang, and Zhao-Xiang Zhang. Sequence level semantics aggregation for video object detection. In *ICCV*, 2019. 3

[28] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 2, 7

[29] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: real-time 3d object detection from point clouds. In *CVPR*, 2018. 2

[30] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. *ICCV*, 2019. 7

[31] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector, 2020. 2, 8

[32] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. IPOD: intensive point-based object detector for point cloud. *CoRR*, 2018. 2

[33] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: sparse-to-dense 3d object detector for point cloud. *ICCV*, 2019. 2

[34] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In *CVPR*, 2020. 2, 3

[35] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, 2019. 4, 8

[36] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *CoRL*, 2019. 2, 3, 7

[37] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CVPR*, 2018. 2, 5