INT: Towards Infinite-frames 3D Detection with An Efficient Framework

Jianyun Xu, Zhenwei Miao†, Da Zhang, Hongyu Pan, Kaixuan Liu, Peihan Hao, Jun Zhu, Zhengyang Sun, Hongmin Li, and Xin Zhan

Alibaba Group
{xujianyun.xjy,zhenwei.mzw}@alibaba-inc.com †corresponding author

Abstract. It is natural to construct a multi-frame instead of a single-frame 3D detector for a continuous-time stream. Although increasing the number of frames might improve performance, previous multi-frame studies only used very limited frames to build their systems due to the dramatically increased computational and memory cost. To address these issues, we propose a novel on-stream training and prediction framework that, in theory, can employ an infinite number of frames while keeping the same amount of computation as a single-frame detector. This **in**fini**t**e framework (INT), which can be used with most existing detectors, is utilized, for example, on the popular CenterPoint, with significant latency reductions and performance improvements. We've also conducted extensive experiments on two large-scale datasets, nuScenes and Waymo Open Dataset, to demonstrate the scheme's effectiveness and efficiency. By employing INT on CenterPoint, we can get around 7% (Waymo) and 15% (nuScenes) performance boost with only 2~4ms latency overhead, and currently SOTA on the Waymo 3D Detection leaderboard.

Keywords: infinite, multi-frame, 3D detection, efficient, pointcloud.

1 Introduction

3D object detection from pointclouds has been proven as a viable robotics vision solution, particularly in autonomous driving applications. Many single-frame 3D detectors [27,23,42,15,36,39,10,26,20] are developed to meet the real-time requirement of the online system. Nevertheless, it is more natural for a continuous-time system to adopt multi-frame detectors that can fully take advantage of the time-sequence information. However, as far as we know, few multi-frame 3D detectors are available for long frame sequences due to the heavy computation and memory burden. It is desirable to propose a concise real-time long-sequence 3D detection framework with promising performance.

Existing works [39,22,11,38,5,32,37] demonstrate that multi-frame models yield performance gains over single-frame ones. However, these approaches require loading all the used frames at once during training, resulting in very limited frames being used due to computational, memory, or optimization difficulties. Taking the SOTA detector CenterPoint [39] as an example, it only uses two

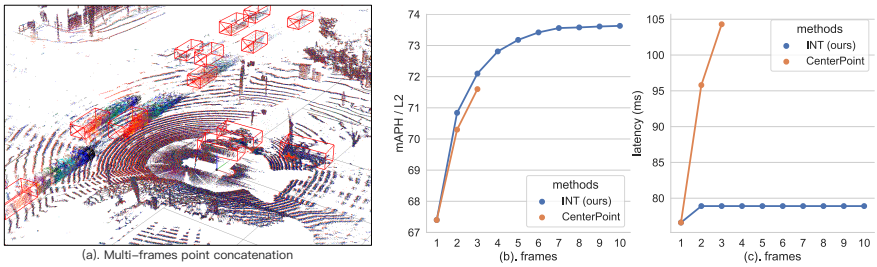

Fig. 1. Impact of frames used in detectors on Waymo *val* set. While CenterPoint's performance improves as the number of frames grows, the latency also increases dramatically. On the other hand, our INT keeps the same latency while increasing frames.

frames [39] on the Waymo Open Dataset [29]. While increasing the number of frames can boost the performance, it also leads to significant latency burst as shown in Fig. 1. Memory overflow occurs if we keep increasing the frames of CenterPoint, making both training and inference impossible. As a result, we believe that the number of frames used is the bottleneck preventing multi-frame development, and we intend to break through this barrier first.

There are two major problems that limit the number of frames in a multi-frame detector: 1) repeated computation. Most of the current multi-frame frameworks have a lot of repeated calculations or redundant data that causes computational spikes or memory overflow; 2) optimization difficulty. Some multi-frame systems have longer gradient conduction links as the number of frames increases, introducing optimization difficulties.

To alleviate the above problems, we propose INT (short for **in**fini**t**e), an on-stream system that theoretically allows training and prediction to utilize infinite number of frames. INT contains two primary components: 1) a Memory Bank (MB) for temporal information fusion and 2) a Dynamic Training Sequence Length (DTSL) strategy for on-stream training. The MB is a place to store the recursively updated historical information so that we don't have to compute past frames' features repeatedly. As a result, it only requires a small amount of memory but can fuse infinite data frames. To tackle the problem of optimization difficulty, we truncate the gradient of back propagation to the MB during training. However, an inherent flaw of iteratively updating MB on a training stream is that historical and current information are not given by the same model parameters, leading to training issues. To solve this problem, DTSL is employed. The primary idea of DTSL is to start with short sequences and quickly clear the MB to avoid excessive inconsistency between historical and current data; then gradually lengthen the sequence as training progresses since the gap between historical and current model parameters becomes negligible.

To make INT feasible, we propose three modules: SeqFusion, SeqSampler and SeqAug. SeqFusion is a module in MB for temporal fusion that proposes multiple fusion methods for two types of data commonly used in pointcloud detection, i.e., point-style and image-style. SeqSampler is a sequence index generator that DTSL

uses to generate training sequences of different lengths at each epoch. Finally, SeqAug is a data augmentation for on-stream training, capable of maintaining the same random state on the same stream.

Our contributions can be summarized as:

- We present INT, an on-stream multi-frame system made up of MB and DTSL that can theoretically be trained and predicted using infinite frames while consuming similar computation and memory as a single-frame system.
- We propose three modules, SeqFusion, SeqSampler and SeqAug, to make INT feasible.
- We conduct extensive experiments on nuScenes and Waymo Open Dataset to illustrate the effectiveness and efficiency of INT.

2 Related Work

2.1 3D Object Detection

Recent studies on 3D object detection can be broadly divided into three categories: LiDAR-based [27,23,42,15,36,39,10,16,19,43,35], image-based [14,4,31,17], and fusion-based [3,18,24,30,40,22,41]. Here we focus on LiDAR-based schemes.

According to the views of pointclouds, 3D detectors can be classified as point-based, voxel-based, range-based, and hybrid. PointRCNN [27] and VoteNet [23] are two representative point-based methods that use a structure like PointNet++ [25] to extract point-by-point features. These schemes are characterized by better preservation of pointclouds' original geometric information. Still, they are sensitive to the number of pointclouds that pose serious latency and memory issues. In contrast, voxel-based solutions, such as VoxelNet [42], PointPillars [15], Second [36], CenterPoint [39] and AFDetV2 [10] are less sensitive to the number of pointclouds. They convert the pointcloud into 2D pillars or 3D voxels first, then extract features using 2D or 3D (sparse) convolution, making them easier to deploy. Another category is rangeview-based schemes [16,19,6] that perform feature extraction and target prediction on an efficient unfolded spherical projection view. Meanwhile, they contend with target scale variation and occlusion issues, resulting in performance that is generally inferior to that of voxel-based schemes. Hybrid methods [26,43,35,20] attempt to integrate features from several views to collect complementing information and enhance performance.

We define two data styles according to the data format to facilitate the analysis of different detectors:

- ***Image-style***. Well-organized data in 2D, 3D, or 4D dimensions similar to that of an image.
- ***Point-style***. Disorganized data such as pointclouds and sparse 3D voxels.

The data in pointcloud-based detectors, including input, intermediate feature, and output, is generally point-style or image-style. We design the fusion algorithms for both point-style and image-style data in INT's Memory Bank, so that INT can be employed in most 3D detectors. In this work, we choose the

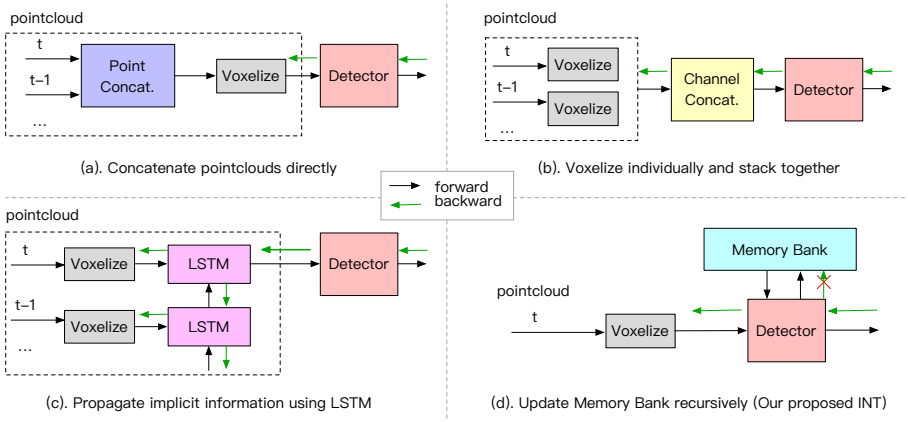

Fig. 2. Training phase of different multi-frame schemes. Operations inside dash rectangle either involve repetitive computation or raise memory burden, which leads to a very limited frame number for training.

recently popular CenterPoint [39] as the baseline for studies since it performs better in terms of efficiency and performance, and it is now scoring at SOTA level on the large-scale open datasets nuScenes [2] and Waymo Open Dataset [29].

2.2 Multi-frame Methods

There have been a variety of LSTM-based techniques in the field of video object detection, such as [7,13,33]. Transformer-based video detection schemes have recently emerged [9], however transformer may not be suitable for working on-stream because it naturally needs to compute the relationship between all frames, which implies a lot of repeated computations.

Recent methods for multi-frame pointclouds can be roughly divided into three categories as shown in Fig. 2(a), (b) and (c). [2,22,36] directly concatenate multi-frame pointclouds and introduce a channel indicating their relative timestamps, as shown in Fig. 2(a). While this method is simple and effective, it involves a lot of unnecessary computations and increases the memory burden, making it unsuitable for more frames. Instead of merging at the point level, MinkowskiNet [5] and MotionNet [32] combine multiple frames at the feature map level. They must voxelize multi-frame pointclouds independently before stacking the feature maps together and extracting spatio-temporal information using 3D or 4D convolution, as depicted in Fig. 2(b). Obviously, this approach requires repeated data processing and is memory intensive, thus the number of frames is very limited.

To overcome above difficulties, 2020An [11] and 3DVID [38] proposed LSTM or GRU-based solutions to solve the computational and memory issues in the inference phase. However, the gradient transfer to the history frames still results in a considerable memory overhead and introduce optimization difficulties dur-

ing training, so the number of frames cannot be high, as shown in Fig. 2(c). To handle the problem more thoroughly, we propose computing the gradient for the current data and not for the historical data during training, as shown in Fig. 2(d). We then employ a Dynamic Training Sequence Length (DTSL) strategy to eliminate the potential information inconsistency problem. Similarly, 3D-MAN [37] stores historical information in a Memory Bank that does not participate in the gradient calculation. However, 3D-MAN needs to store a fixed number of frames of historical proposals and feature maps, which increases the amount of memory required for its training as the number of frames increases. To get around this problem, we propose recursively updating the Memory Bank's historical information.

To the best of our knowledge, we are the first 3D multi-frame system that can be trained and inferred with infinite frames.

3 Methodology

We present INT framework in this section. The overall architecture is detailed in Sec. 3.1. Sec. 3.2 gives the sequence fusion (SeqFusion) methods of Memory Bank. Sec. 3.3 and 3.4 illustrate the training strategies, including sequence sampler (SeqSampler) and sequence data augmentation (SeqAug), respectively.

3.1 Overview of INT Framework

The INT framework in Fig. 3 is highly compact. The main body consists of a single-frame detector and a recursively updated Memory Bank (MB). The Dynamic Training Sequence Length (DTSL) below serves as a training strategy that is not needed for inference (inference only needs the pointclouds input in chronological order). In addition, there are no special requirements for single-frame detector selection. For example, any detector listed in Sec. 2.1 can be utilized.

Memory Bank (MB). The primary distinction between INT and a regular multi-frame detector is the MB, which stores historical data so that we do not have to compute past features repeatedly. MB is comparable to the hidden state in LSTM while it is more flexible, interpretable, and customizable. The user has complete control over where and what information should be saved or retrieved. For example, in Sec. 3.2, we show how to store and update several forms of data. Furthermore, we choose to update the MB recursively to solve the problem of excessive memory cost. To tackle the problem of optimization difficulty, we truncate the gradient of backpropagation to the MB during training.

Dynamic Training Sequence Length (DTSL). A problem with INT training on stream is the information gained from the current observation is not derived using the same model parameters as the past data in the MB. This could

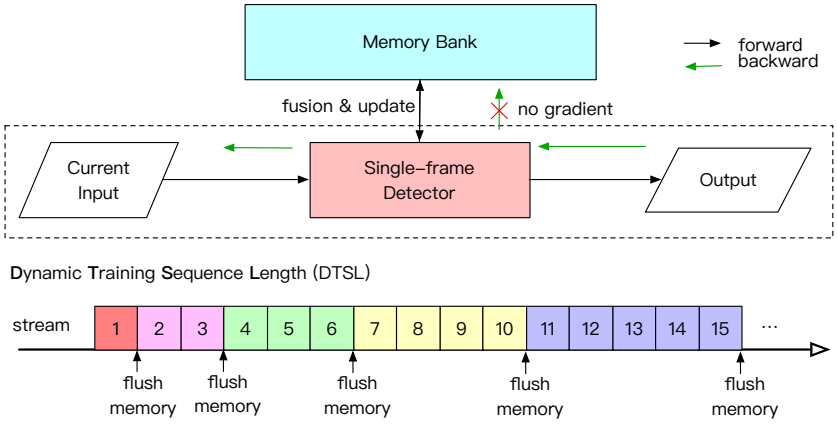

Fig. 3. Overview of INT framework. It consists of a single-frame detector and a Memory Bank. The Dynamic Training Sequence Length below serves as a training strategy.

lead to inconsistencies in training and prediction, which is one of the key reasons why prior multi-frame work was not trained on stream. To solve this problem, we offer the DTSL: beginning with a small sequence length and gradually increasing it, as indicated at the bottom of Fig. 3. This is based on the following observation: as the number of training steps increases, model parameter updates get slower, and the difference in information acquired from different model parameters becomes essentially trivial. As a result, when the model parameters are updated quickly, the training sequence should be short so that the Memory Bank can be cleaned up in time. Once the training is stable, the sequence length can be increased with confidence. DTSL could be defined in a variety of ways, one of which is as follows:

$$DTSL = max(1,\ \lfloor l_{max} \cdot min(1,\ max(0,\ 2 \cdot \frac{ep_{cur}}{ep_{all}} - 0.5))\rfloor) \tag{1}$$

where l_{max} is the maximum training sequence length, ep_{cur} and ep_{all} is current epoch and total epoch number, respectively.

3.2 SeqFusion

Temporal fusion in the Memory Bank is critical in the INT framework. As the type of data in a 3D detector is either point-style or image-style, as indicated in Sec. 2.1, we develop both the point-style and image-style fusion algorithms. In general, original pointcloud, sparse voxel, predicted object, etc., fall into the point-style category. Whereas dense voxel, intermediate feature map, final prediction map, etc., fall into the image-style category.

Point-style fusion. Here we propose a general and straightforward practice: concatenating past point-style data with present data directly, using a channel

to identify the temporal relationship. The historical point-style data is put into a fixed length FIFO queue, and as new observations arrive, foreground data is pushed into it, while oldest data is popped out. According to the poses of ego vehicle, the position information in the history data must be spatially transformed before fusion to avoid the influence of ego movement. The point-style fusion is formulated as:

$$T_{rel} = T_{cur}^{-1} \cdot T_{last}, \tag{2}$$

$$P_f = PointConcat(P_{cur},\ T_{rel} \cdot P_{last}) \tag{3}$$

where T_{last} and T_{cur} are the last and current frame's ego vehicle poses, respectively, while T_{rel} is the calculated relative pose between the two frames. P_{last} refers to the past point-style data in Memory Bank and P_f is the fused data of P_{last} and current P_{cur}.

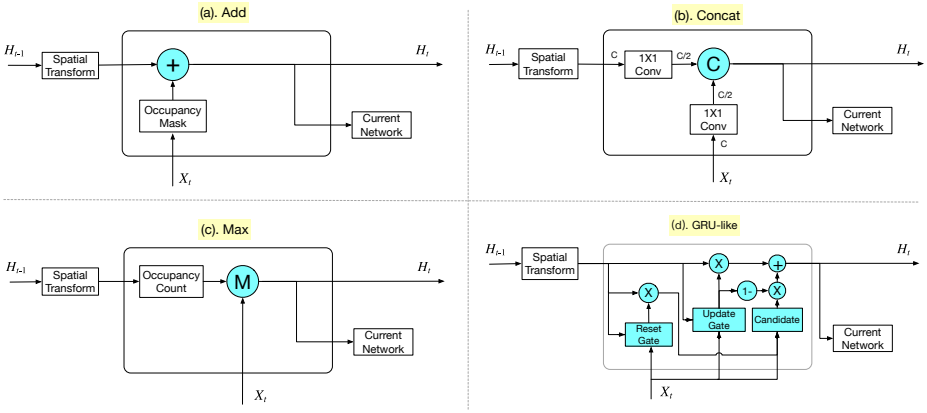

Fig. 4. Four temporal fusion methods for image-style data. Occupancy Mask and Occupancy Count in *Add* and *Max* are used to distinguish different moments.

Image-style fusion. We propose four fusion algorithms for the image-style data, including *Add*, *Max*, *Concat* and *GRU-like* as depicted in Fig. 4 (a), (b), (c) and (d). As the historical image-style data and current data should be identical in dimensions based on recursive updates, *Add* and *Max* are simple in design and implementation. The computational overhead of these two fusion methods is cheap. We also devise the *Concat* fusion approach, in which both the historical and the current feature channels are first compressed to 1/2 of the origin and then concatenated along channel dimension. To investigate the impact of long-term data, we develop a *GRU-like* fusion method with learnable parameters to select which data should be kept and which should be discarded. To eliminate

the effect of ego vehicle motion, historical image-style data must be spatially transformed first. The image-style fusion process can be summarized as follows:

$$\tilde{I}_{last} = F_{sample}(I_{last},\ F_{affine}(T_{rel},\ s)), \tag{4}$$

$$I_f = Fusion(I_{cur},\ \tilde{I}_{last}) \tag{5}$$

where T_{rel} is the same as Eq. 3. $F_{affine}(\cdot)$ and $F_{sample}(\cdot)$ refer to *affine grid* and *grid sample* operation respectively, which are proposed in [12] for image-style data transformation. I_{last} refers to the past image-style data in Memory Bank and I_f is the fused data of I_{last} and current I_{cur}. s is the shape of I_{last}. $Fusion(\cdot)$ can be *Add*, *Max*, *Concat* and *GRU-like*, as shown in Fig. 4.

3.3 SeqSampler

SeqSampler is the key to perform the training of INT in an infinite-frames manner. It is designed to split original sequences to target length, and then generate the indices of them orderly. If the sequence is infinite-long, the training or inference can go on infinitely. DTSL is formed by executing SeqSampler with different target lengths for each epoch.

The length of original sequences in a dataset generally varies, for as in the Waymo Open Dataset [29], where sequence lengths oscillate around 200. Certain datasets, such as nuScenes [2], may be interval labeled, with one frame labeled every ten frames. As a result, the SeqSampler should be designed with the idea that the source sequence will be non-fixed in length and will be annotated at intervals. The procedures of SeqSampler are as simple as *Sequence Sort* and *Sequence Split*, as indicated in Fig. 5. In *Sequence Sort*, we rearrange the random input samples orderly by sequences. Then split them to target length in *Sequence Sort*, and may padding some of them to meet the batch or iteration demands.

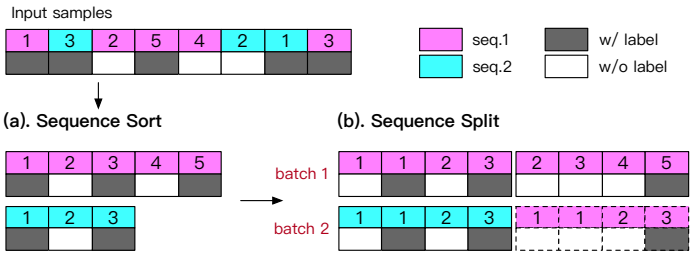

Fig. 5. An example of SeqSampler. There are two sequences: seq1 contains 5 frames and seq2 has 3, both are interval labeled. Given the desired batch size 2 and target length 4, we need to get the final iteration indices. First, the two sequences are sorted separately. Then, the original sequences are splitted to 3 segments in the target length, and a segment is randomly replicated (dashed rectangles) to guarantee that both batches have the same number of samples.

3.4 SeqAug

Data augmentation has been successful in many recent 3D detectors [36,15,39]. However, because of the shift in training paradigm, our suggested INT framework can not directly migrate current validated data augmentation methods. One of the main reasons for this is that INT is trained on a stream with a clear association between the before and after frames, whereas data augmentation is typically random, and the before and after frames could take various augmentation procedures. To solve this problem and allow INT to benefit from data augmentation, we must verify that a certain method of data augmentation on the same stream maintains the same random state at all times. We term the data augmentation that meets this condition SeqAug. According to the data augmentation methods widely employed in pointcloud detection, SeqAug can be split into two categories: Sequence Point Transformation (flipping, rotation, translation, scaling, and so on) and Sequence GtAug (copy and paste of the ground truth pointclouds).

Sequence Point Transformation. If a pointcloud is successively augmented by flipping T_f, rotation T_r, scaling T_s, and translation T_t, the other frames in the same stream must keep the same random state to establish a reasonable temporal relationship. In addition, because of these transformations, T_{rel} in Eq. 3 must be recalculated:

$$T_{rel} = T_t \cdot T_s \cdot T_r \cdot T_f \cdot T_{cur}^{-1} \cdot T_{last} \cdot T_f \cdot T_r^{-1} \cdot T_s^{-1} \cdot T_t^{-1} \tag{6}$$

where T_{last} and T_{cur} are the last and current frame's ego vehicle poses.

Sequence GtAug. Similarly, recording random states of the same stream is required to ensure that the sequential objects from the Gt database can be copied and pasted consecutively, as shown in Fig. 6.

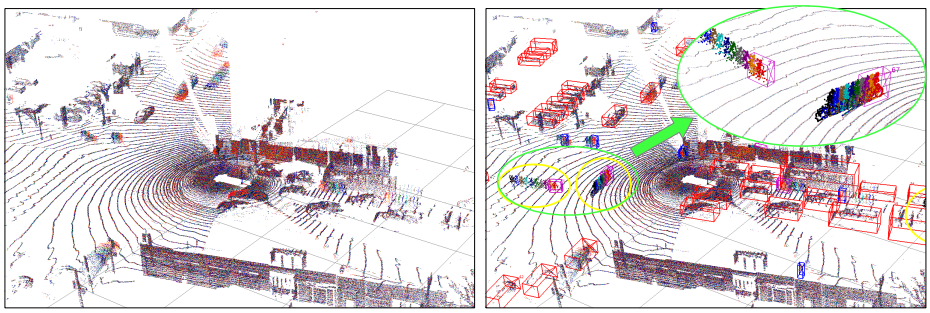

Fig. 6. An example of Sequence GtAug. The colors of pointclouds represent different moments, with red being the current frame.

4 Experiments

In this paper, we build the proposed INT framework based on the highly competitive and popular detector CenterPoint [39]. In the following sections, we first briefly introduce the datasets in Sec. 4.1, followed by a description of a few critical experimental setups in Sec. 4.2. The efficiency and effectiveness of the INT framework are then illustrated in Sec. 4.3 by comparing it to the baseline CenterPoint, followed by Sec. 4.4, which compares the results of INT on the Waymo test set to other SOTAs. Finally, Sec. 4.5 is several INT ablation experiments.

4.1 Datasets

This section briefly describes the two open datasets used in this paper.

Waymo Open Dataset. Waymo [29] comprises 798, 202 and 150 sequences for train, validation and test, respectively. Each sequence lasts around 20 seconds and contains about 200 frames. There are three categories for detection: VEHICLE, PEDESTRIAN, and CYCLIST. The mean Average Precision (mAP) and mAP weighted by heading accuracy (mAPH) are the official 3D detection metrics. There are two degrees of difficulty: LEVEL 1 for boxes with more than five LiDAR points, and LEVEL 2 for boxes with at least one LiDAR point. In this paper, we utilize the officially prescribed mAPH on LEVEL 2 by default.

nuScenes. There are 1000 driving sequences in nuScenes [2], with 700, 150, and 150 for training, validation, and testing, respectively. Each sequence lasts about 20 seconds and has a LiDAR frequency of 20 frames per second. The primary metrics for 3D detection are mean Average Precision (mAP) and nuScenes Detection Score (NDS). NDS is a weighted average of mAP and other attribute measurements such as translation, scale, orientation, velocity, and other box properties. In this study, we employ mAP and NDS as experimental results.

4.2 Experimental Settings

We employ the same network designs and training schedules as CenterPoint [39] and keep the positive and negative sample strategies, post-processing settings, loss functions, etc., unchanged.

Backbone settings. VoxelNet [36,42] and PointPillars [15] are two 3D encoders used by CenterPoint, dubbed CenterPoint-Voxel and CenterPoint-Pillar, respectively. Our INT also experiments with these two backbones, which correspond to INT-Voxel and INT-Pillar, respectively.

Frame settings. Although INT can be trained and inferred on an infinite number of frames, the sequence length of the actual dataset is finite. To facilitate comparison with previous work and demonstrate the benefits of the INT framework, we select a few specific frames on nuScenes and Waymo Open Dataset. We use 10, 20 and 100 training frames on nuScenes, and 10 training frames on Waymo Open Dataset.

Fusion settings. On INT, we choose three kinds of data added to Memory Bank: point-style foreground pointcloud, image-style intermediate feature map,

and image-style final prediction map. For the foreground pointcloud, we fuse the historical points with the current points during the input phase and then update them based on the predictions at the end of network. For the intermediate feature map, we fuse and update its historical information at the same position before the Region Proposal Network. For the final prediction map, we fuse and update historical information simultaneously before the detection header. Appendix A.4 takes INT-Voxel as a typical example to provide a more specific explanation.

Latency settings. To test the network's actual latency, we remove redundant parts of the data processing in CenterPoint [39] and just maintain the data IO and memory transfer (to the GPU) operations. We shift the essential voxelization component to the GPU to limit the CPU's influence. We also build up data prefetching in the dataloader to lessen IO effect and run latency tests when it is stabilized. Finally, the following test circumstances are used: CUDA Version 10.2, cudnn 7.6.5, GeForce RTX 2070 SUPER, Driver Version 460.91.03.

4.3 Effectiveness and Efficiency

As shown in Table 1 and 2, we first compare to the baseline CenterPoint to demonstrate the paper's main point, i.e., the effectiveness and efficiency of INT. In these two tables, we try two types of backbone, termed E-PointPillars and E-VoxelNet in the columns, and the unit of latency is milliseconds. The settings of INT can be referred to Table 4 and 5. CenterPoint with multiple frames refers to concatenating multi-frame pointclouds at the input level, which introduces repetitive computation and additional memory burden as analyzed in Sec. 2.2. For example, two-frames CenterPoint in Table 1 increases the latency by around 20 ms when compared to its single-frame counterpart (more details in Appendix A.6). In contrast, the latency of INT is unaffected by the number of frames used, and its performance is much better than that of multi-frame CenterPoint as the number of frames grows. As can be obviously observed in Table 1 and 2, INT shows significant improvements in both latency and performance, gaining around 7% mAPH (Waymo) and 15% NDS (nuScenes) boost while only adding 2˜4 ms delay when compared to single-frame CenterPoint.

4.4 Comparison with SOTAs

In this section, we compare the results of INT on the Waymo *test* set with those of other SOTAs schemes, as shown in Table 3. The approaches are divided into two groups in the table: single-frame and multi-frame. It is seen that the multi-frame scheme is generally superior to the single-frame scheme. Most multi-frame approaches employ the original pointclouds concatenation [28,39,10], and we can see that the number of frames is used fewer due to computational and memory constraints. Finally, our suggested INT scheme outperforms other SOTA schemes by a large margin. As far as we know, INT is the best non-ensemble approach on Waymo Open Dataset leaderboard[1].

[1] https://waymo.com/open/challenges/2020/3d-detection/

Table 1. Effectiveness and efficiency of INT on Waymo Open Dataset *val* set. The APH of L2 difficulty is reported. The "-2s" suffix in the rows means two-stage model. CenterPoint's mAPH results are obtained from official website, except for those with a *, which are missing from the official results and were reproduced by us.

methods	frames	E-PointPillars					E-VoxelNet				
		VEH↑	PED↑	CYC↑	mAPH↑	latency↓	VEH↑	PED↑	CYC↑	mAPH↑	latency↓
CenterPoint	1	65.5	55.1	60.2	60.3	**57.7**	66.2	62.6	67.6	65.5	**71.7**
CenterPoint	2	66.6*	61.9*	62.3*	63.6*	77.8	67.3	67.5	69.9	68.2	90.9
INT (ours)	2	66.2	60.4	64.4	63.7	61.6	69.4	69.1	72.6	70.3	74.0
INT (ours)	10	**69.6**	**66.3**	**65.7**	**67.2**	61.6	**72.2**	**72.1**	**75.3**	**73.2**	74.0
CenterPoint-2s	1	66.7	55.9	61.7	61.4	**61.7**	67.9	65.6	68.6	67.4	**76.6**
CenterPoint-2s	2	68.4*	63.0*	64.3*	65.2*	82.9	69.7	70.3	70.9	70.3	95.8
INT-2s (ours)	2	67.9	61.7	66.0	65.2	65.9	70.8	68.7	73.1	70.8	78.9
INT-2s (ours)	10	**70.8**	**67.0**	**68.1**	**68.6**	65.9	**73.3**	**71.9**	**75.6**	**73.6**	78.9

Table 2. Effectiveness and efficiency of INT on nuScenes *val* set. CenterPoint's mAP and NDS results are obtained from official website, except for those with a *, which are missing from the official results and were reproduced by us.

methods	frames	E-PointPillars			E-VoxelNet		
		mAP↑	NDS↑	latency↓	mAP↑	NDS↑	latency↓
CenterPoint	1	42.5*	46.4*	**39.2**	49.7*	50.7*	**81.1**
CenterPoint	10	50.3	60.2	49.4	59.6	66.8	117.2
INT (ours)	10	49.3	59.9	43.0	58.5	65.5	84.1
INT (ours)	20	50.7	61.0	43.0	60.9	66.9	84.1
INT (ours)	100	**52.3**	**61.8**	43.0	**61.8**	**67.3**	84.1

Table 3. Comparison with SOTAs on Waymo Open Dataset *test* set. We only present the non-emsemble approaches, and INT is currently the best non-emsemble solution on the Waymo Open Dataset leaderboard[1], to the best of our knowledge. Accessed on 2 March 2022.

Methods	Frames	VEH-APH↑		PED-APH↑		CYC-APH↑		mAPH↑	
		L1	L2	L1	L2	L1	L2	L1	L2
StarNet [21]	1	61.0	54.5	59.9	54.0	-	-	-	-
PointPillars [15]	1	68.1	60.1	55.5	50.1	-	-	-	-
RCD [1]	1	71.6	64.7	-	-	-	-	-	-
M3DeTR [8]	1	77.2	70.1	58.9	52.4	65.7	63.8	67.1	61.9
HIK-LiDAR [34]	1	78.1	70.6	69.9	64.1	69.7	67.2	72.6	67.3
CenterPoint [39]	1	79.7	71.8	72.1	66.4	-	-	-	-
3D-MAN [37]	15	78.3	70.0	66.0	60.3	-	-	-	-
RSN [28]	3	80.3	71.6	75.6	67.8	-	-	-	-
CenterPoint [39]	2	80.6	73.0	77.3	71.5	73.7	71.3	77.2	71.9
CenterPoint++ [39]	3	82.3	75.1	78.2	72.4	73.3	71.1	78.0	72.8
AFDetV2 [10]	2	81.2	73.9	78.1	72.4	75.4	73.0	78.2	73.1
INT (ours)	10	83.1	76.2	78.5	72.8	74.8	72.7	78.8	73.9
INT (ours)	100	**84.3**	**77.6**	**79.7**	**74.0**	**76.3**	**74.1**	**80.1**	**75.2**

4.5 Ablation Studies

Impact of different fusion data. This section investigates the impact of various fusion data used in INT. We use one kind of point-style data, the foreground pointcloud, and two kinds of image-style data, the intermediate feature map before RPN and the final prediction map in this paper. The fusion method of foreground pointcloud is termed as *PC Fusion* which is explained in Sec. 3.2. The fusion method of the intermediate feature map and the final prediction map is *Concat*, as described in Sec. 3.2, named as *FM Fusion* and *PM Fusion*, respectively. The fusion results of these three kinds of data on Waymo Open Dataset and nuScenes are shown in Table 4 and 5. First, the tables' performance columns show that all the three fusion data have considerable performance boosts, with *PC Fusion* having the highest effect gain. Then according to the latency columns, the increase in time of different fusion data is relatively small, which is very cost-effective given the performance benefit.

Table 4. Impact of different fusion data on Waymo Open Dataset *val* set. By default, the training sequence length was set to 10 frames. In order to indicate how the final result comes in Table 1, we also add a column called "Two Stage".

PC	FM	PM	Two	E-PointPillars					E-VoxelNet				
Fusion	Fusion	Fusion	Stage	VEH↑	PED↑	CYC↑	mAPH↑	latency↓	VEH↑	PED↑	CYC↑	mAPH↑	latency↓
				65.5	55.1	60.2	60.3	**57.4**	66.2	62.6	67.6	65.5	**71.5**
√				68.1	65.8	65.4	66.4	59.5	71.7	70.8	74.2	72.3	72.7
	√			63.6	63.3	64.7	63.8	59.0	66.1	67.3	73.8	69.1	72.2
		√		66.4	64.0	64.2	64.5	58.2	67.7	68.1	74.1	70.0	72.0
√	√			69.5	66.8	64.8	67.0	60.9	72.0	71.8	76.5	73.5	73.3
√	√	√		69.6	66.3	65.7	67.2	61.6	72.2	**72.1**	**76.1**	73.5	74.0
√	√	√	√	**70.8**	**67.0**	**68.1**	**68.6**	65.9	**73.3**	71.9	75.6	**73.6**	78.9

Table 5. Impact of different fusion data on nuScenes *val* set. By default, the training sequence length was set to 10 frames. In order to indicate how the final result comes in Table 2, we also add a column called "100 frames".

PC	FM	PM	100	E-PointPillars			E-VoxelNet		
Fusion	Fusion	Fusion	frames	mAP	NDS	latency	mAP	NDS	latency
				42.5	46.4	**39.2**	49.7	50.7	**81.1**
√				47.1	58.5	41.2	56.6	64.5	82.4
	√			48.4	57.4	40.5	55.9	56.6	82.1
		√		45.3	56.0	39.7	53.9	62.7	82.0
√	√			48.7	59.8	42.4	58.4	65.2	83.3
√	√	√		49.3	59.9	43.0	58.5	65.5	84.1
√	√	√	√	**52.3**	**61.8**	43.0	**61.8**	**67.3**	84.1

Impact of training sequence length. The length indicated here actually refer to the maximum length since Dynamic Training Sequence Length (DTSL) is used

in training. Fig. 1 depicts the relationship between frames and performance for the 2-stage INT-Voxel model on the Waymo Open Dataset. We also plot the 2-stage CenterPoint-Voxel results together to make comparisons clearer. As shown in Fig. 1(b), INT improves as the number of frames increases, although there is saturation after a certain point (See Appendix A.5 for more explanation); as for Fig. 1(c), the time consumed by INT is slightly higher than that of single-frame CenterPoint, but it does not increase with the number of frames.

Impact of sequence augmentation. Sec. 3.4 introduces SeqAug, a data augmentation technique for on-stream training, and this section examines the role of Point Transformation and GtAug in SeqAug. As seen in Table 6, both augmentation strategies result in significant performance improvements, making data augmentation essential for INT training just as regular detectors do.

Table 6. Impact of SeqAug on Waymo Open Dataset *val* set. One-stage INT-Pillar and INT-Voxel are used.

Sequence Point Trans.	Sequence GtAug	E-PointPillars				E-VoxelNet			
		VEL-APH↑	PED-APH↑	CYC-APH↑	mAPH↑	VEL-APH↑	PED-APH↑	CYC-APH↑	mAPH↑
		59.6	56.7	56.2	57.5	65.0	62.4	61.9	63.1
√		69.3	65.0	62.4	65.6	71.8	71.4	73.1	72.1
√	√	**69.6**	**66.3**	**65.7**	**67.2**	**72.2**	**72.1**	**76.1**	**73.5**

More ablation studies in appendix. The impact of sequence length on nuScenes is shown in Appendix A.1. The performance and latency of temporal fusion methods for image-style data (proposed in Sec. 3.2) are shown in Appendix A.2. The impact of DTSL proposed in Sec. 3.1 can be found in Appendix A.3.

5 Conclusion

In this paper, we present INT, a novel on-stream training and prediction framework that, in theory, can employ an infinite number of frames while using about the same amount of computational and memory cost as a single-frame detector. To make INT feasible, we propose three key modules, i.e., SeqFusion, SeqSampler, and SeqAug. We utilize INT on the popular CenterPoint, with significant latency reductions and performance improvements, and rank **1st** currently on Waymo Open Dataset 3D Detection leaderboard among the non-ensemble SOTA methods. Moreover, the INT is a general multi-frame system, which may be used for tasks like segmentation and motion as well as detection.

Acknowledgement: This work was supported by Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba Research Intern Program.

References

1. Bewley, A., Sun, P., Mensink, T., Anguelov, D., Sminchisescu, C.: Range conditioned dilated convolutions for scale invariant 3d object detection. arXiv preprint arXiv:2005.09927 (2020) 12
2. 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. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11621–11631 (2020) 4, 8, 10
3. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1907–1915 (2017) 3
4. Chong, Z., Ma, X., Zhang, H., Yue, Y., Li, H., Wang, Z., Ouyang, W.: Monodistill: Learning spatial features for monocular 3d object detection. arXiv preprint arXiv:2201.10830 (2022) 3
5. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3075–3084 (2019) 1, 4
6. Fan, L., Xiong, X., Wang, F., Wang, N., Zhang, Z.: Rangedet:in defense of range view for lidar-based 3d object detection. In: Proceedings of the IEEE International Conference on Computer Vision (2021) 3
7. Feng, Y., Ma, L., Liu, W., Luo, J.: Spatio-temporal video re-localization by warp lstm. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1288–1297 (2019) 4
8. Guan, T., Wang, J., Lan, S., Chandra, R., Wu, Z., Davis, L., Manocha, D.: M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers. In: Proceedings of the IEEE Winter Conference on Applications of Computer Vision (2021) 12
9. He, L., Zhou, Q., Li, X., Niu, L., Cheng, G., Li, X., Liu, W., Tong, Y., Ma, L., Zhang, L.: End-to-end video object detection with spatial-temporal transformers. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 1507–1516 (2021) 4
10. Hu, Y., Ding, Z., Ge, R., Shao, W., Huang, L., Li, K., Liu, Q.: Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds. In: Proceedings of the AAAI Conference on Artificial Intelligence (2021) 1, 3, 11, 12
11. Huang, R., Zhang, W., Kundu, A., Pantofaru, C., Ross, D.A., Funkhouser, T., Fathi, A.: An lstm approach to temporal 3d object detection in lidar point clouds. In: European Conference on Computer Vision (2020) 1, 4
12. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in neural information processing systems. vol. 28 (2015) 8
13. Kang, K., Li, H., Xiao, T., Ouyang, W., Yan, J., Liu, X., Wang, X.: Object detection in videos with tubelet proposal networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 727–735 (2017) 4
14. Ku, J., Pon, A.D., Waslander, S.L.: Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11867–11876 (2019) 3
15. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019) 1, 3, 9, 10, 12

16. Li, B., Zhang, T., Xia, T.: Vehicle detection from 3d lidar using fully convolutional network. arXiv preprint arXiv:1608.07916 (2016) 3
17. Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X.: Gs3d: An efficient 3d object detection framework for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1019–1028 (2019) 3
18. Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7345–7353 (2019) 3
19. Meyer, G.P., Laddha, A., Kee, E., Vallespi-Gonzalez, C., Wellington, C.K.: Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12677–12686 (2019) 3
20. Miao, Z., Chen, J., Pan, H., Zhang, R., Liu, K., Hao, P., Zhu, J., Wang, Y., Zhan, X.: Pvgnet: A bottom-up one-stage 3d object detector with integrated multi-level features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3279–3288 (2021) 1, 3
21. Ngiam, J., Caine, B., Han, W., Yang, B., Vasudevan, V.: Starnet: Targeted computation for object detection in point clouds. arXiv preprint arXiv:1908.11069 (2019) 12
22. Piergiovanni, A., Casser, V., Ryoo, M.S., Angelova, A.: 4d-net for learned multimodal alignment. In: Proceedings of the IEEE International Conference on Computer Vision (2021) 1, 3, 4
23. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE International Conference on Computer Vision (2019) 1, 3
24. 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) 3
25. 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. vol. 30 (2017) 3
26. Shi, S., Guo, C., Jiang, L., Wang, Z., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020) 1, 3
27. 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 (2019) 1, 3
28. Sun, P., Wang, W., Chai, Y., Elsayed, G., Bewley, A., Zhang, X., Sminchisescu, C., Anguelov, D.: Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021) 11, 12
29. 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 the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2446–2454 (2020) 2, 4, 8, 10
30. Vora, S., Lang, A.H., Helou, B., Beijbom, O.: Pointpainting: Sequential fusion for 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4604–4612 (2020) 3
31. Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.Q.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection

for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8445–8453 (2019) 3
32. Wu, P., Chen, S., Metaxas, D.N.: Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11385–11395 (2020) 1, 4
33. Xiao, F., Lee, Y.J.: Video object detection with an aligned spatial-temporal memory. In: European Conference on Computer Vision. pp. 485–501 (2018) 4
34. Xu, J., Tang, X., Dou, J., Shu, X., Zhu, Y.: Centeratt: Fast 2-stage center attention network. arXiv preprint arXiv:2106.10493 (2021) 12
35. Xu, J., Zhang, R., Dou, J., Zhu, Y., Sun, J., Pu, S.: Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 16024–16033 (2021) 3
36. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10) (2018) 1, 3, 4, 9, 10
37. Yang, Z., Zhou, Y., Chen, Z., Ngiam, J.: 3d-man: 3d multi-frame attention network for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1863–1872 (2021) 1, 5, 12
38. Yin, J., Shen, J., Guan, C., Zhou, D., Yang, R.: Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020) 1, 4
39. Yin, T., Zhou, X., Krahenbuhl, P.: Center-based 3d object detection and tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021) 1, 3, 4, 9, 10, 11, 12
40. Yoo, J.H., Kim, Y., Kim, J., Choi, J.W.: 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In: European Conference on Computer Vision. pp. 720–736. Springer (2020) 3
41. Zeng, Y., Zhang, D., Wang, C., Miao, Z., Liu, T., Zhan, X., Hao, D., Ma, C.: Lift: Learning 4d lidar image fusion transformer for 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 17172–17181 (2022) 3
42. 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 (2018) 1, 3, 10
43. 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. pp. 923–932. PMLR (2020) 3