



3d object detection

자율주행 LiDAR



2조 Team. 벼리



팀원 소개



CEO 조영수

Chief Executive Officer
가장 높은 의사결정 총 책임자

CSO 기은서

Chief Strategy Officer
전략 수립/실행 총괄 책임자

CTO 김현규

Chief Technology Officer
기술 활용/관리 총괄 책임자

목차

- 01 Introduction
- 02 3D Object Detection
- 03 Datasets
- 04 Lidar-based models
- 05 Results
- 06 Conclusion & Future works

Introduction

☆☆ 주제 선정 배경



자율주행 알고리즘 개발 공모전

- 알고리즘 개발 지원 부문
 - 2D 객체 인식
 - 3D 객체 인식
 - 2D Semantic segmentation
- 3D 객체 인식 선정 이유
 - 2D Object Detection의 프로젝트는 이론 공부 와 프로젝트를 진행해본 경험이 있음
 - 3D Object Detection에 대한 심화적인 공부와 성능 개선 전략의 새로운 시도를 해보고자 함

Introduction

★ 주행 환경 객체 인식 point cloud

주행 환경에서 3D 객체를 인식하는 3가지 방법

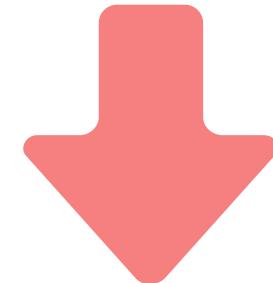
- 이미지 데이터만을 활용하는 방식
- Point cloud 데이터만을 활용하는 방식
- 이미지와 데이터와 point cloud 데이터를 결합해 활용하는 방식

.....

Point cloud란?

- 3차원 공간 상에서 XYZ 좌표 정보이며 추가적으로 intensity 정보 사용
- point들이 구름처럼 이루어져 있다고 하여 이름이 붙어짐

공모전 제공 데이터: Point Cloud , JSON File

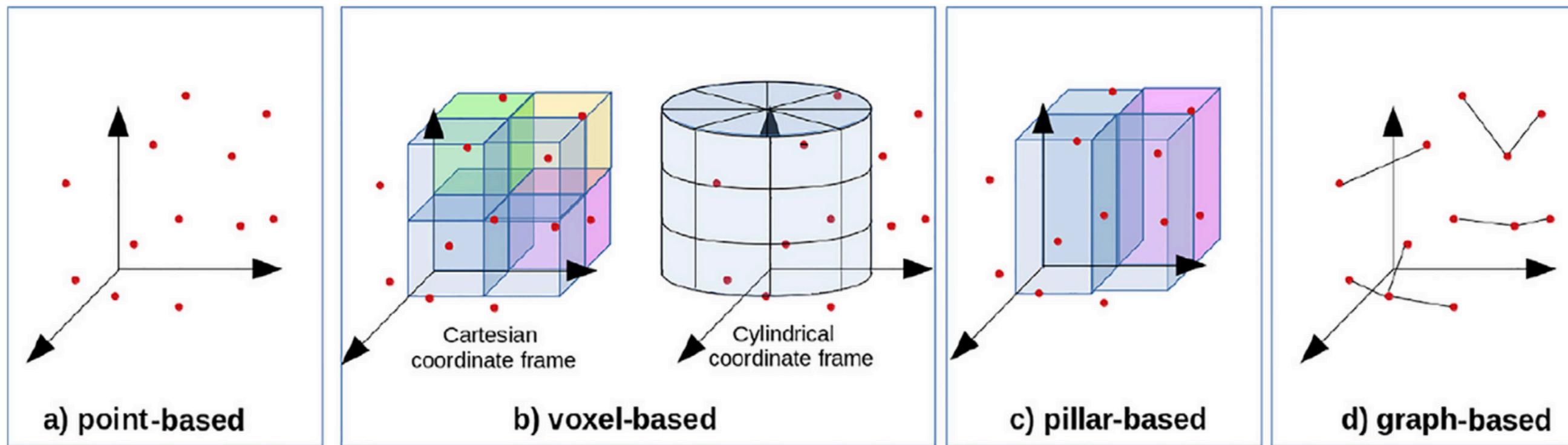


문제 정의

Point cloud만 사용 하였을 시
객체인식 성능을 향상 시킬 수
있는 방법은 무엇일까?

3D Object Detection

Point cloud data 인코딩 방식



출처: <https://www.sciencedirect.com/science/article/abs/pii/S0097849321001321?via%3Dihub>

순열 불변성, 비정형, 무순서적
특성이 있다는 것을 확인

일정한 격자 형태를
사용할 수 없음

객체 인식을 위한 특징 추출을
위해서 다양한 인코더를 사용

3D Object Detection

☆☆☆ 3D 객체 인식 Flowchart

• 기본적인 구성 (VoxelNet 예시)

a. 3D Backbone

i. Voxel Feature Extractor

- 입력으로 point cloud 좌표

ii. 3D Sparse Convolution

b. 2D Backbone

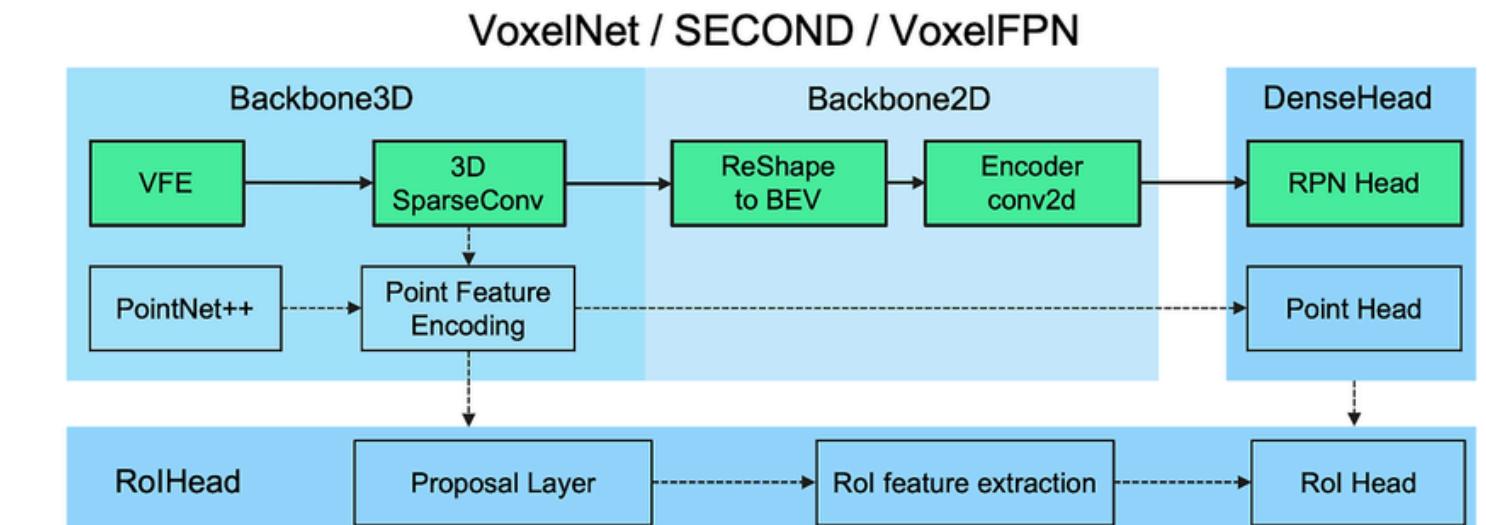
i. Map to BEV (Bird Eye's View)

ii. Encoder convolution 2d

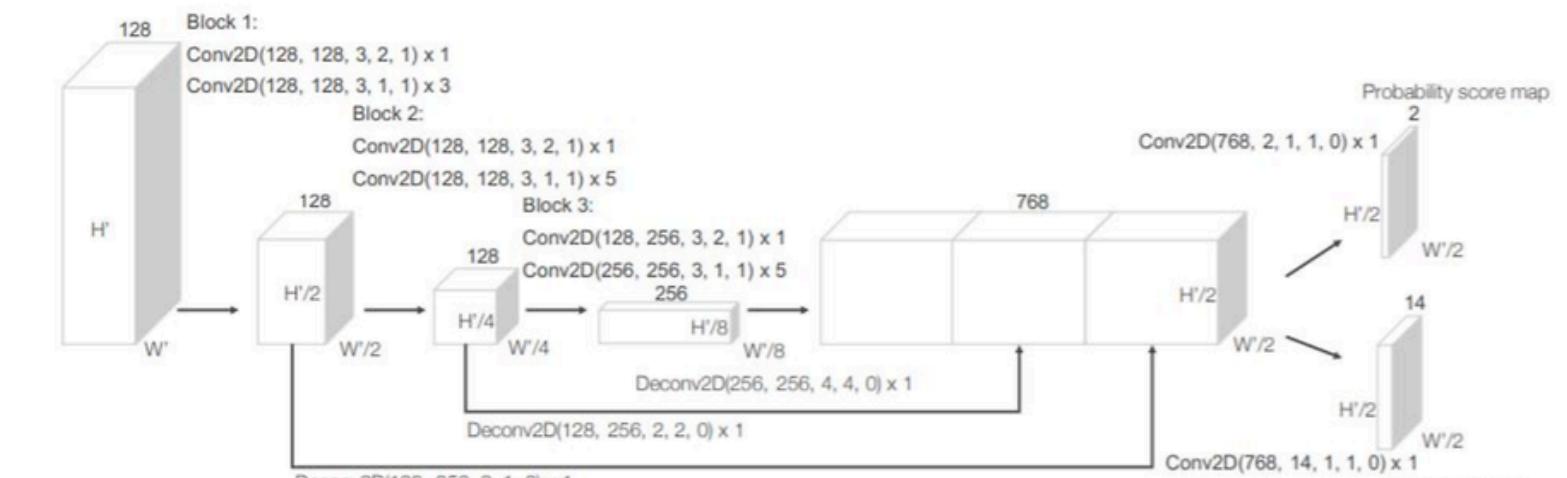
c. Dense Head

i. RPN Head

- 감지된 Object별 3D box 추출
- class score와 bounding box regression 결과 산출



OpenPCDet 아키텍처



VoxelNet의 RPN 아키텍처

출처: <https://github.com/open-mmlab/OpenPCDet/tree/master>

Datasets

Lidar 기반 데이터셋

KITTI

- a. 2012년도 데이터셋
- b. 3 classes의 객체 존재
- c. 3 difficulty levels 존재
- d. mAP, mAOS metrics을 사용함

nuScenes

- a. 2019년도 데이터셋
- b. 10 classes의 객체 존재
- c. mAP, NDS score metrics를 사용함

Waymo

- a. 2019년도 데이터셋
- b. 3 classes의 객체 존재
- c. 2 difficulty levels 존재
- d. mAP, mAPH metrics를 사용함

	KITTI	nuScenes	Waymo open dataset
Lidar sensor	1x Velodyne HDL-64 (64 Channels Lidar)	1x Velodyne HDL-32 (32-Channels Lidar)	1x 75m range, 4x HoneyComb 20m range
Additional sensors	2x Stereo Camera, GPS, IMU	6x Camera, 5x Radar, GPS, IMU	5x Camera
Annotated frames	15K	40K	230K
Scenes amount	22	1000	1150
Hours	1.5	5.5	6.4
Object class	8	23	4
3D Boxes amount	200K	1.4M	12M
Location	One City (Karlsruhe)	Two Cities (Boston, Singapore)	3 Regions (USA)
Scenes weather	Sunny, Cloudy	Various Weather	Sunny, Cloudy
Scenes time	Day	Day, Night	Day, Night, Dusk, Dawn
Published year	2012	2019	2019

출처: <https://www.sciencedirect.com/science/article/abs/pii/S0097849321001321?via%3Dihub>

Lidar-based models

★ 기존 모델

PointPillar (2019)

a. Contributions

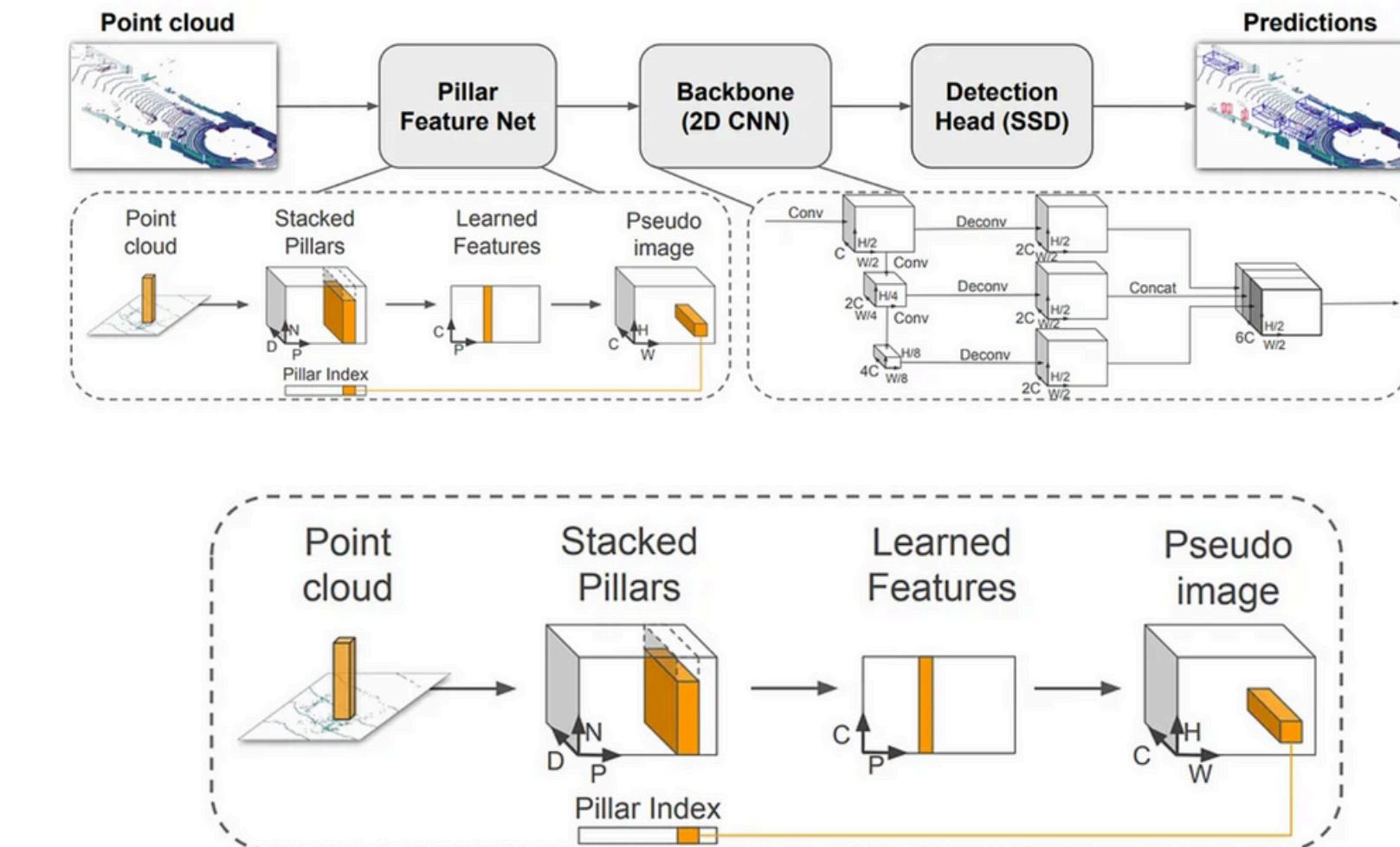
- i. Novel point cloud 인코더를 제안하여 end-to-end 학습 가능
- ii. 3D object detection을 2D convolution 연산만 가지고 수행하였으며 62 fps 속도 작동

a. 구조

- i. Feature 인코더 네트워크
 - point cloud는 pseudo-image로 변환
- ii. 2D convolution backbone을 가짐
- iii. Detection head에서 탐지를 수행하고 3D bounding box 예측

a. 결과

- i. Pillar-wise로 end-to-end하게 학습한 3D Object Detection 모델



출처: <https://arxiv.org/abs/1812.05784>

Lidar-based models

★ 기존 모델

VoxelNeXT (2023)

a. Contributions

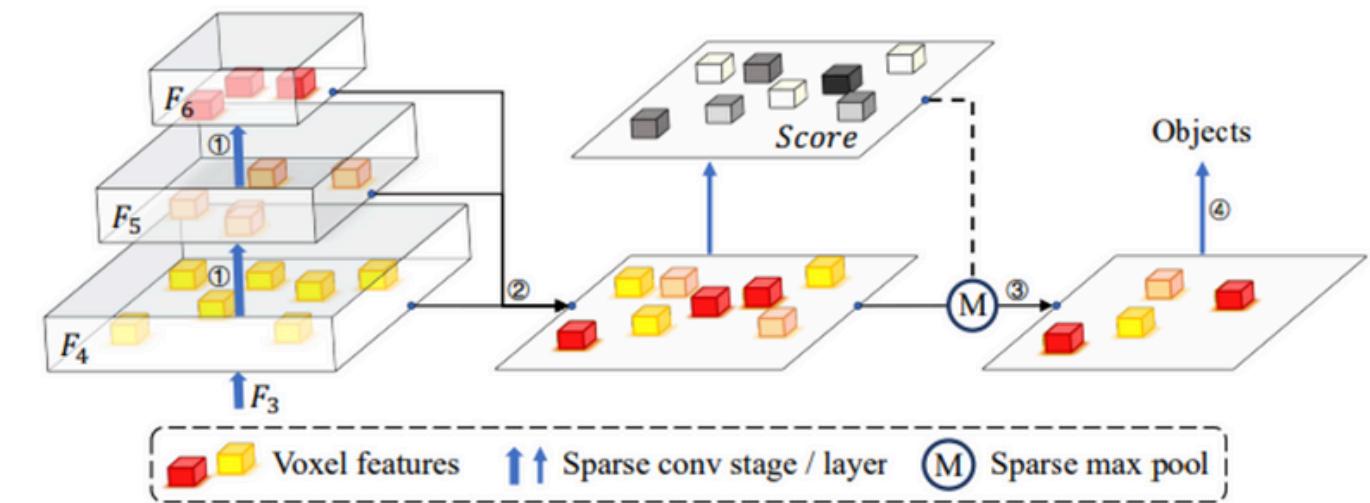
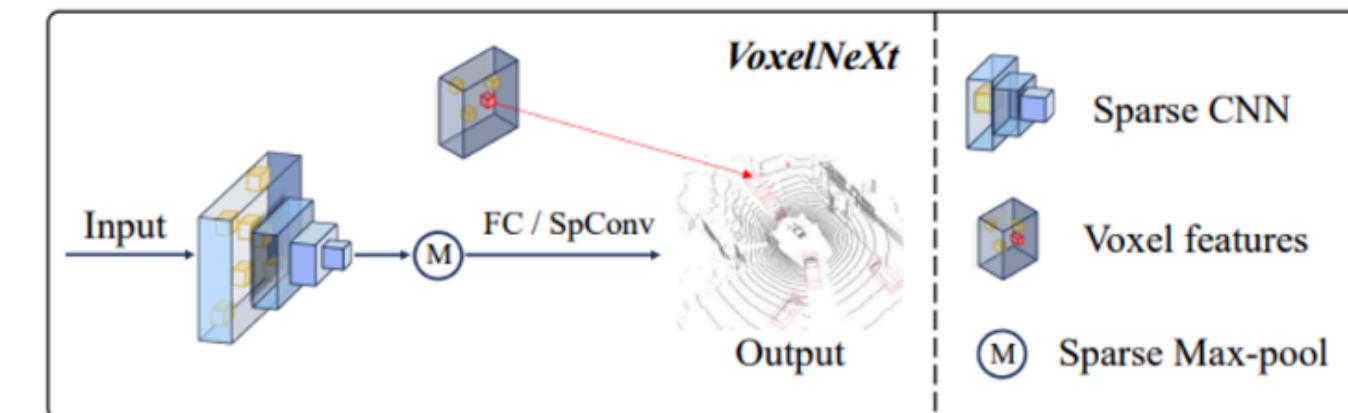
- i. 효과적인 sparse convolution network 사용함
- ii. Voxel feature들로만 3D 객체 인식 및 Tracking 가능
- iii. Densify 와 NMS 제거함) 제거함

a. 구조

- i. Sparse CNN backbone
 - 1. 추가적인 Down-sampling & Sparse 높이 compression
 - 2. 밀집하지 않은 공간의 Voxel Pruning
- ii. Sparse 예측 head
 - 1. Voxel 선별
 - 2. Box Regression

a. 결과

- i. Voxel 기반 예측이 가능하고 효과적인 것을 처음으로 보여줌
 - Anchor나 center가 불필요함



출처: <https://arxiv.org/abs/1711.06396>

Lidar-based models

★ 기존 모델

CenterPoint (2021)

a. Contributions

- i. 탐색 공간 축소
- ii. 후속 작업의 간소화
- iii. 효율적인 포인트 기반 특징 추출

a. 구조

i. 2 단계 3D Object Detection 모델

ii. First stage

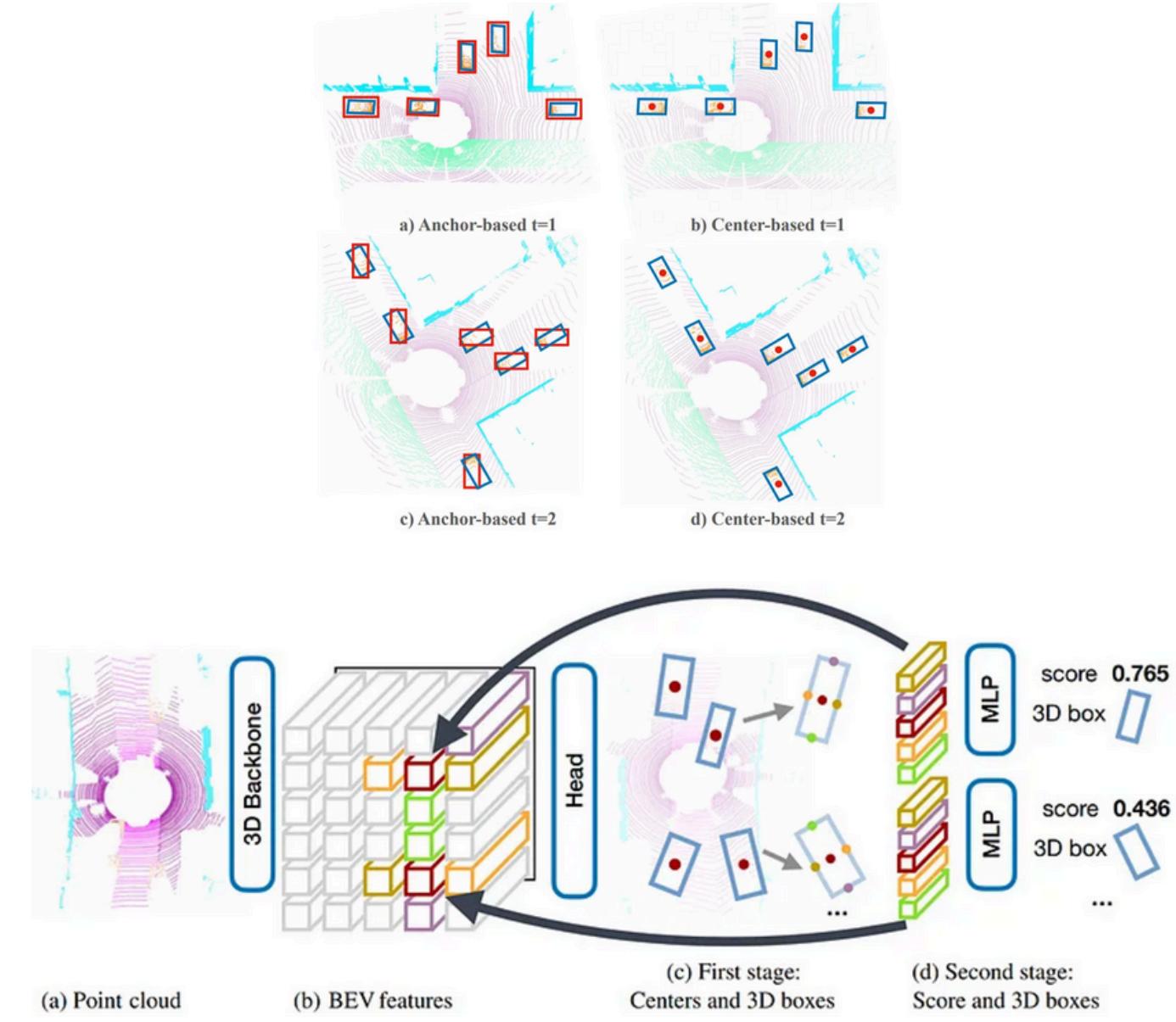
- 1. Object의 중심 찾음
- 2. 3D Bbox 찾음

iii. Second stage

- 1. class- agnostic confidence score 찾음
- 2. 3D Bbox 찾음

a. 결과

- 단순하고, 효율적이면, 효과적인 3D Object Detection 및 Object Tracking 모델



출처: <https://arxiv.org/abs/2006.11275>

☆☆ 제안한 model

Voxel Feature Extractor (전처리)

- Mean VFE (기존 VFE)
 - 각 voxel마다 모든 점들의 평균 features를 계산하는 함수
- Dynamic Voxel Feature Extractor
 - 역활
 - 동일한 voxel에 속한 포인트들의 피쳐 중에서 최대값, 중심값 등을 사용하여 새로운 voxel 생성
 - 필요성
 - Point cloud는 간단하지 않은 데이터라 더 customized된 전처리 필요

Map to BEV (2차원으로 만드는 과정)

- HeightCompression (기존 Map to BEV)
 - Depth와 channel을 곱해서 5차원을 4차원으로 만들어줌
- PointPillarsscatter
 - 가능
 - 여러 배치에 대해 피쳐 맵을 생성할 수 있도록 설계
 - 실제로 포인트 클라우드가 있는 부분에만 연산 집중
 - 필요성
 - 배치마다 Point cloud의 희소성을 더 고려해야 함

Results

Metrics

mAP

- 2D축의 중심광의 거리 d로 threshold를 사용하여 매칭

$$mAP = \frac{1}{|\mathbb{C}||\mathbb{D}|} \sum_{c \in \mathbb{C}} \sum_{d \in \mathbb{D}} AP_{c,d}$$

mTP

- 중심값의 거리를 2m를 기준으로 class가 맞은 것의 비율

$$mTP = \frac{1}{|\mathbb{C}|} \sum_{c \in \mathbb{C}} TP_c$$

NDS

$$NDS = \frac{1}{10} [5 mAP + \sum_{mTP \in \mathbb{TP}} (1 - \min(1, mTP))]$$

- 절반은 객체 인식에 달려 있고 절반은 객체 인식 했을 때 품질에 달려 있음

Results

☆ Quantitative results

Model 이름	mAP	NDS
PointPillars (2019)	0.3860	0.5309
CenterPoint (2020)	0.5222	0.6079
VoxelNeXT (2023)	0.5543	0.6265
제안한 모델	0.5352	0.6170

데이터셋

- nuScene 데이터셋을 사용하여 비교

결과값 비교

- CenterPoint와 제안한 모델

◦ mAP와 NDS 성능 15% 향상

Conclusion

☆☆ 3D Object Detection 분석

1. Point Cloud Data

- 입력값
 - x,y,z,r (instensity)로 구성됨
- 특성
 - 순열 불변성, 비정형, 무순서적

2. 3D Object Detection

- 3D Backbone
 - VFE & 3D CNN
- 2D Backbone
 - Map to BEV & 2D CNN
- Dense Head

2. 제안한 모델

- VFE (Voxel Feature Extractor)
 - 기존 meanVFE를 DynamicVFE 교체
- Map to BEV
 - 기존 HeightCompression Pointpillarsscatter로 교체
- 성능 향상
 - mAP와 NDS가 각각 약 15%
 - 아직 VoxelNeXT 성능 따라가지 못함

Future work

☆☆ 3D backbone & 경량화

1. 3D Backbone

a. DVST (CVPR 2023)

- 효율적이고 deployment-freindly 3D backbone

b. LION

- RNN 기반의 간단하고 효과적인 창 기반 3D backbone인 LION을 제안

2. AI Model 경량화

a. TensorRT

- NVIDIA GPU에서 모델을 더 빠르게 실행하기 위한 최적화 된 런타임 엔진
 - 특히 딥러닝 모델을 배포 환경에서 더 효율적으로 실행하고 추론(inference) 성능을 향상



3D Backbone을 교체하고 경량화를 통해 성능 개선 가능

LiDAR 3D Object Detection

감사합니다

Team. 벼리

2조