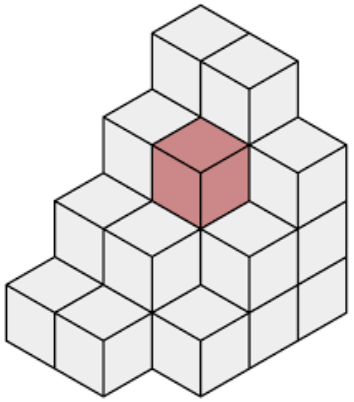
세미나

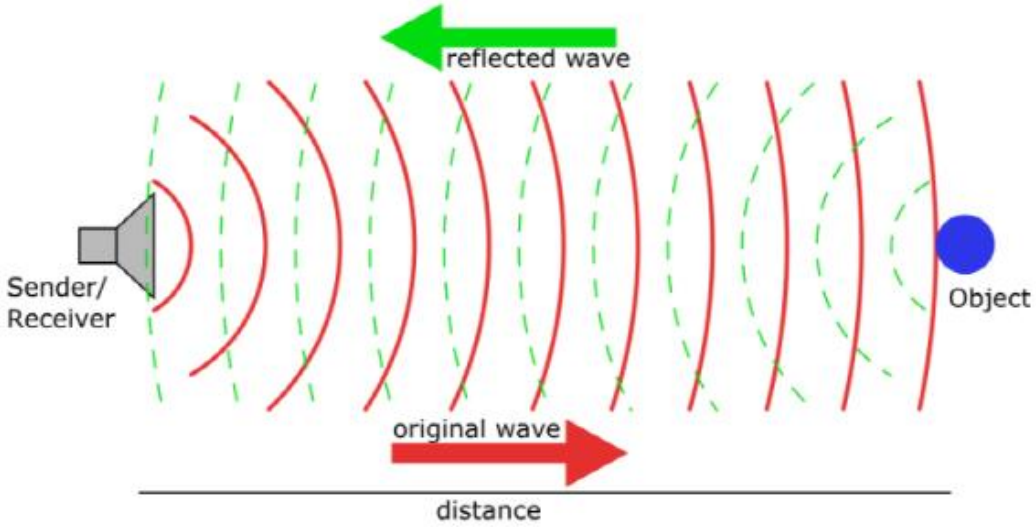
이제희

VoxelNet

Voxel = Volume + Pixel (부피를 가진 픽셀)



Point Cloud : 3차원 공간상에 퍼져 있는 포인트들의 집합



VoxelNet

3D Object Detection을 목적으로 하는 네트워크

3D point cloud (x, y, z, ri)

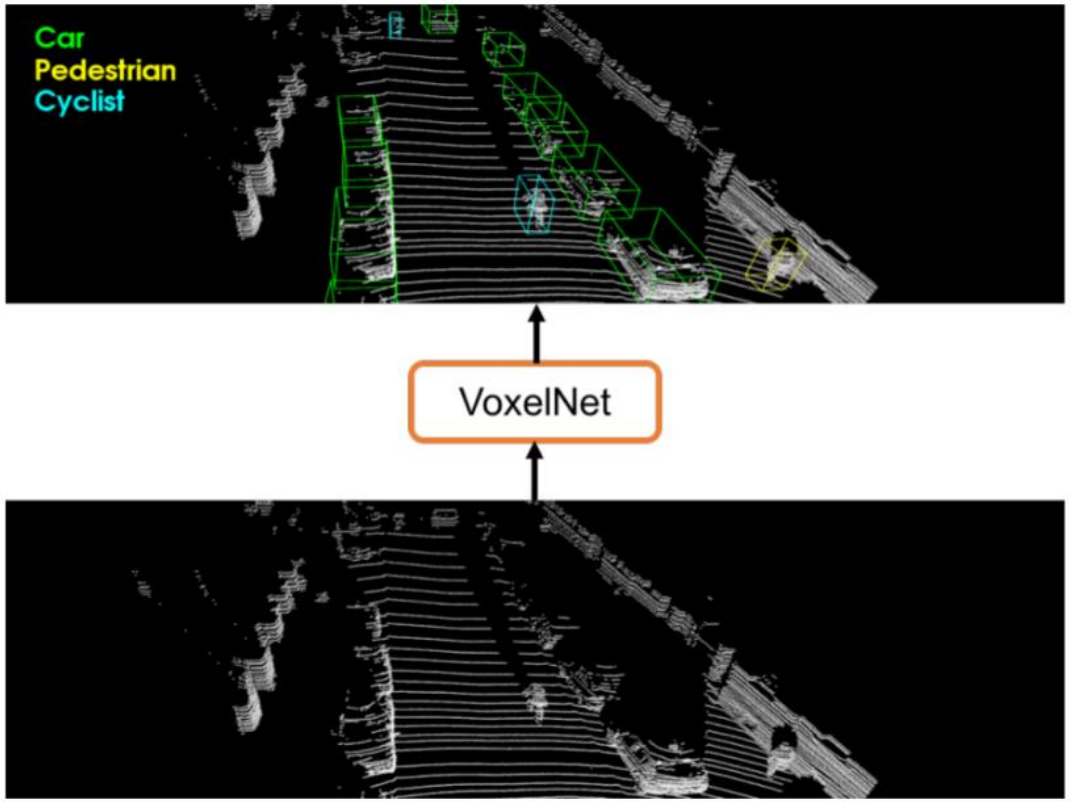
↓

Probablity score map

각 앵커에 대해 객체가 존재할 확률

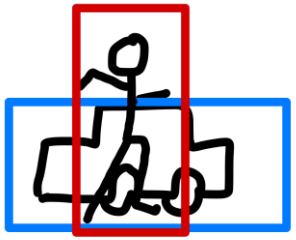
Regression map

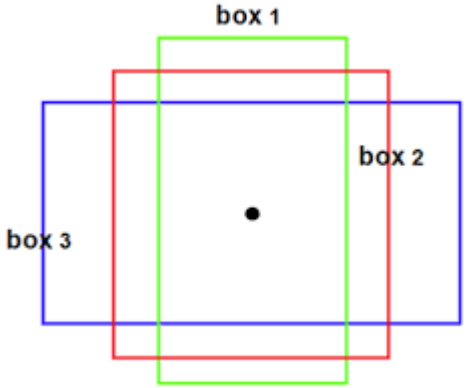
각 앵커에 대해 객체의 경계 상자를 예측



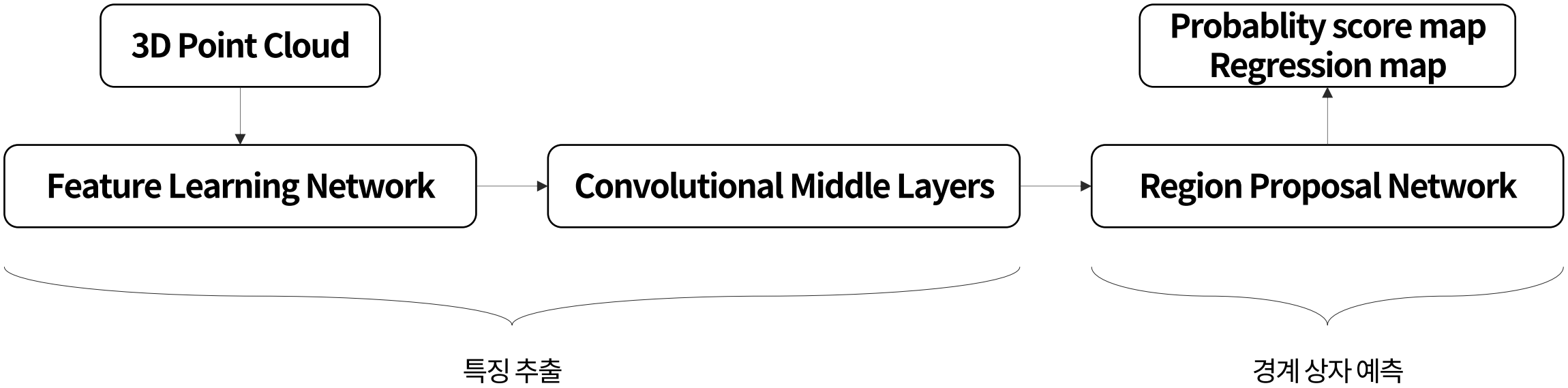
Anchor box: 미리 정해진 object가 있음직한 box

2가지 이상의 object가 겹쳐있을 때 이를 따로 검출하기 위해서 고안





VoxelNet

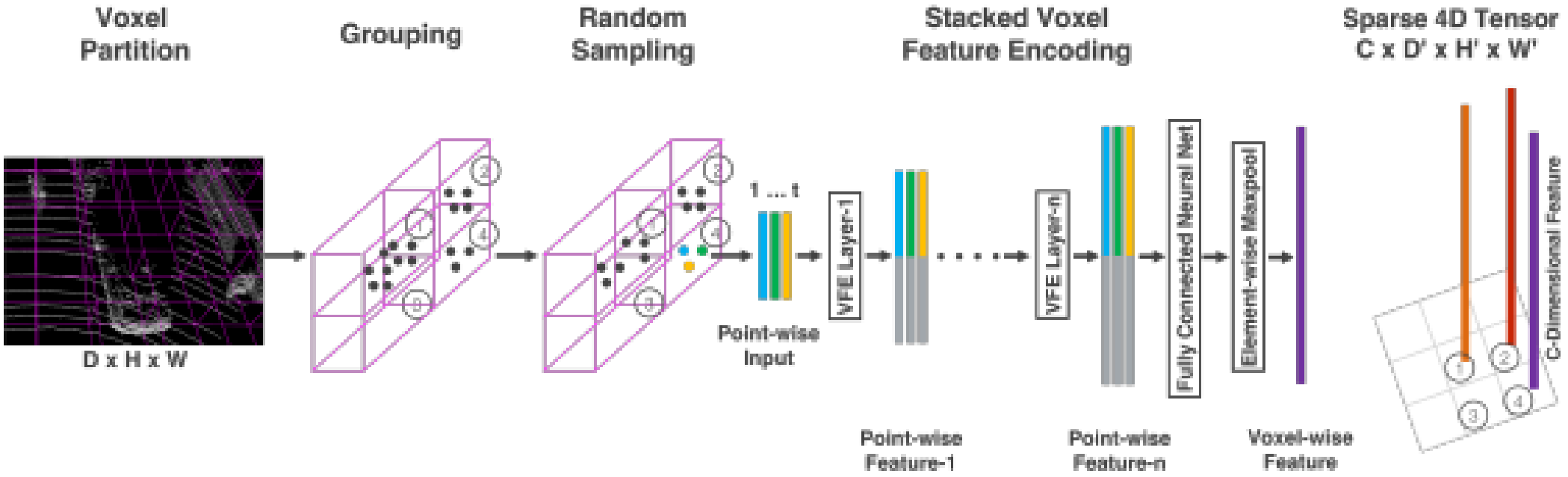


1. 여러 겹의 VFE layer를 통해 point cloud를 동일 크기의 3D Voxel로 나누어 encoding

2. Convolution layers를 사용해 이전 네트워크에서 생성된 특징을 더 고차원적으로 변환

3. RPN을 통해 확률 점수 맵과 회귀 맵을 얻음

Feature Learning Network

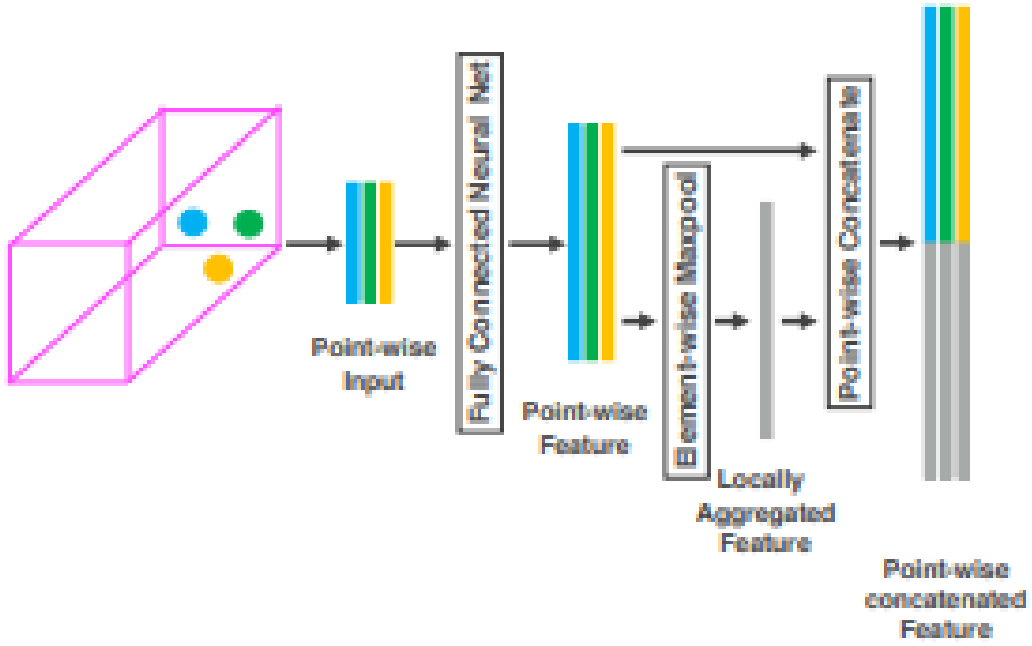


1. 3D point cloud를 일정한 voxel로 나누고 그룹화

2. Voxel의 내부 point에 대해서 VFE Layers에 통과시켜 특징 vector 추출

3. 비어있지 않은 voxel들만 처리해 각 voxel들의 특징을 sparse 4D tensor로 표현
   (tensor에는 voxel의 위치와 특징 모두 포함)

Voxel Feature Encoding(VFE)



Voxel 내부의 각 point의 값을 voxel 중심과의 상대적 오프셋을 포함하도록 변경

$$(x, y, z, ri) \rightarrow Vin(x, y, z, ri, x - vx, y - vy, z - vz)$$

Vin → FCN → point-wise feature
(FCN : 고차원 특징 벡터 생성 (linear layer, BN, ReLU))

Point-wise feature → Maxpooling → Locally aggregated feature
(Maxpooling : 특징 통합)

Output :
Point-wise feature + Locally aggregated feature

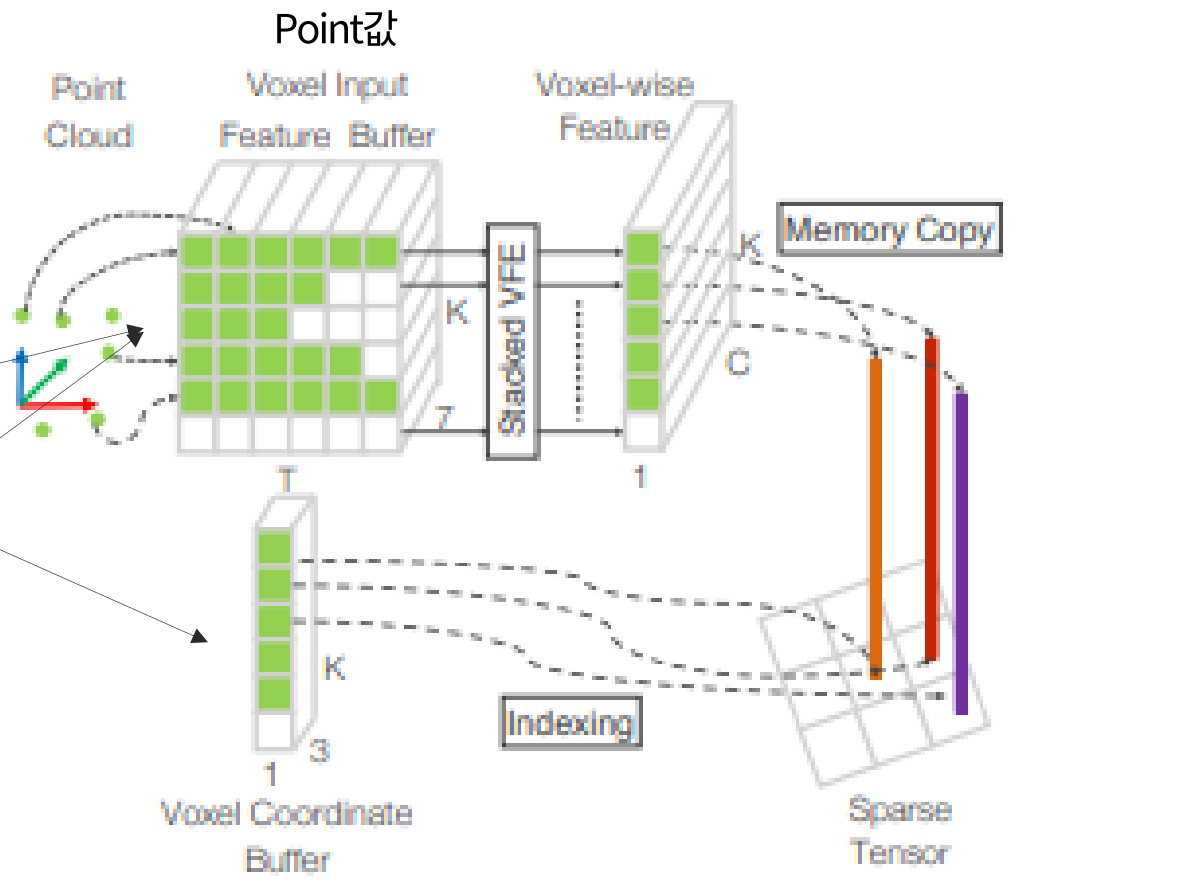
Efficient Implementation

Point cloud는 희소하게 분포되어 있어 GPU가 처리하기 힘듦
이를 해결하기 위해 밀집 텐서 구조로 변환

모든 point를 순회

Point가 들어갈 voxel이 비었으면

Point가 들어갈 voxel에
다른 point가 이미 존재한다면



Point가 존재하는 Voxel의 좌표 저장

3D voxel grid로 재구성

Convolutional Middle Layers

VFE layer에서 나온 sparse 4D tensor를 변환

Input : sparse 4D tensor (128 x 10 x 400 x 352)

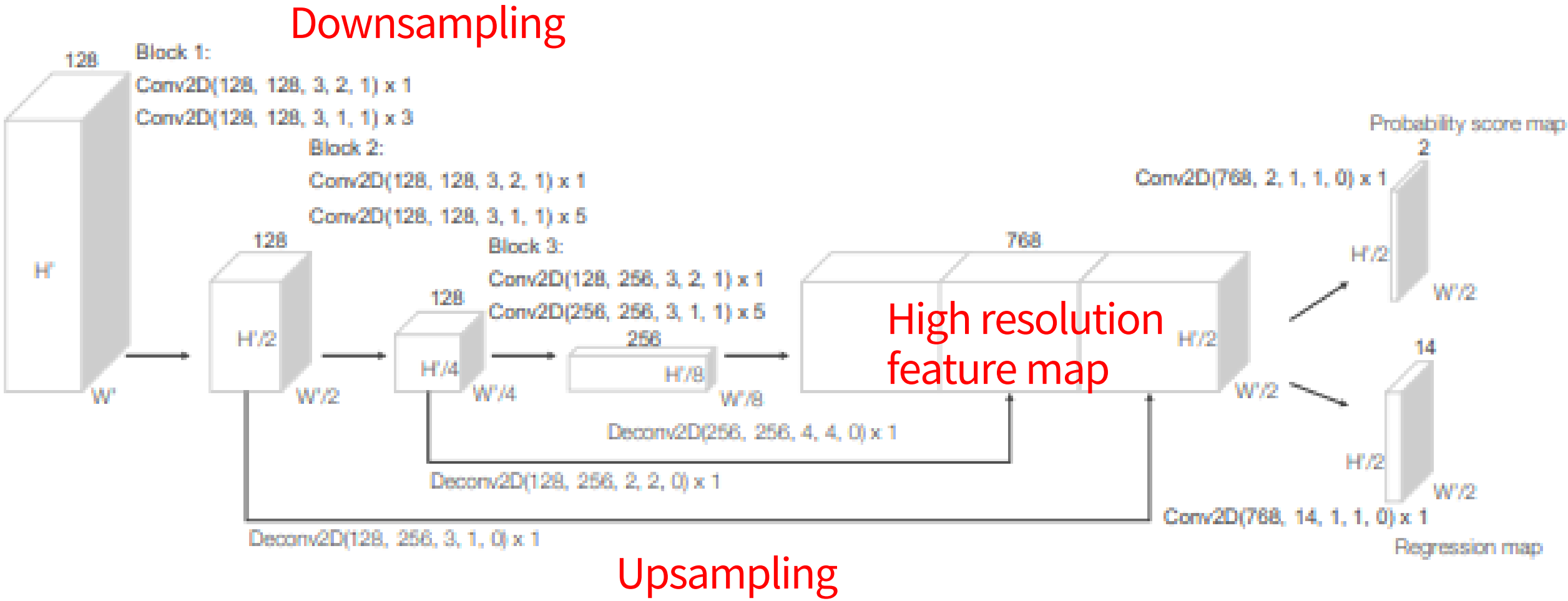
Conv3D(128,64,3,(2,1,1),(1,1,1))

Conv3D(64,64,3,(1,1,1),(0,1,1))

Conv3D(64,64,3,(2,1,1),(1,1,1))

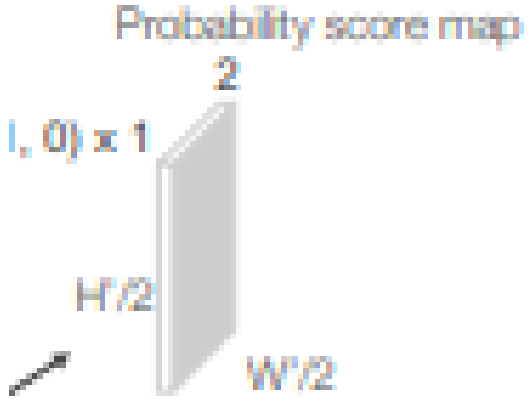
Output : sparse 4D tensor (64 x 2 x 400 x 352)
저해상도로 변경(깊이 정보 줄어듬)

Region Proposal Network



Region Proposal Network

Probability score map
차원 : 2
앵커에 객체가 존재할 확률, 객체가 존재하지 않을 확률
가장 높은 확률을 가진 값을 기반으로 양성 앵커 결정
크로스 엔트로피 손실 함수에 사용하기 위해 양성, 음성 앵커 사용



Regression map
차원 : 14
(중심위치 x, y, z 길이 l 너비 w, 높이 h, 회전 각도 θ) x 2
한 anchor에 대해 7개의 파라미터 2세트 예측
→ 두 가지 회전 각도에 대한 예측을 수행 (0도, 90도)

손실 함수의 값이 최소가 되는 회전 각도를 선택해 적용

Result

난이도: 객체 크기, 가림 상태, 절단 수준에 근거

상위 성능 알고리즘

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [3]	Mono	5.22	5.19	4.13	N/A	N/A	N/A	N/A	N/A	N/A
3DOP [4]	Stereo	12.63	9.49	7.59	N/A	N/A	N/A	N/A	N/A	N/A
VeloFCN [22]	LiDAR	40.14	32.08	30.47	N/A	N/A	N/A	N/A	N/A	N/A
MV (BV+FV) [5]	LiDAR	86.18	77.32	76.33	N/A	N/A	N/A	N/A	N/A	N/A
MV (BV+FV+RGB) [5]	LiDAR+Mono	86.55	78.10	76.67	N/A	N/A	N/A	N/A	N/A	N/A
HC-baseline	LiDAR	88.26	78.42	77.66	58.96	53.79	51.47	63.63	42.75	41.06
VoxelNet	LiDAR	**89.60**	**84.81**	**78.57**	**65.95**	**61.05**	**56.98**	**74.41**	**52.18**	**50.49**

2D 평면

Table 1. Performance comparison in bird's eye view detection: average precision (in %) on KITTI validation set.

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [3]	Mono	2.53	2.31	2.31	N/A	N/A	N/A	N/A	N/A	N/A
3DOP [4]	Stereo	6.55	5.07	4.10	N/A	N/A	N/A	N/A	N/A	N/A
VeloFCN [22]	LiDAR	15.20	13.66	15.98	N/A	N/A	N/A	N/A	N/A	N/A
MV (BV+FV) [5]	LiDAR	71.19	56.60	55.30	N/A	N/A	N/A	N/A	N/A	N/A
MV (BV+FV+RGB) [5]	LiDAR+Mono	71.29	62.68	56.56	N/A	N/A	N/A	N/A	N/A	N/A
HC-baseline	LiDAR	71.73	59.75	55.69	43.95	40.18	37.48	55.35	36.07	34.15
VoxelNet	LiDAR	**81.97**	**65.46**	**62.85**	**57.86**	**53.42**	**48.87**	**67.17**	**47.65**	**45.11**

3D 공간

Table 2. Performance comparison in 3D detection: average precision (in %) on KITTI validation set.

HC-baseline : 손으로 만든 특징 (voxelnet에서 사용하는 특징 학습 네트워크의 성능을 테스트하기 위함)

Result

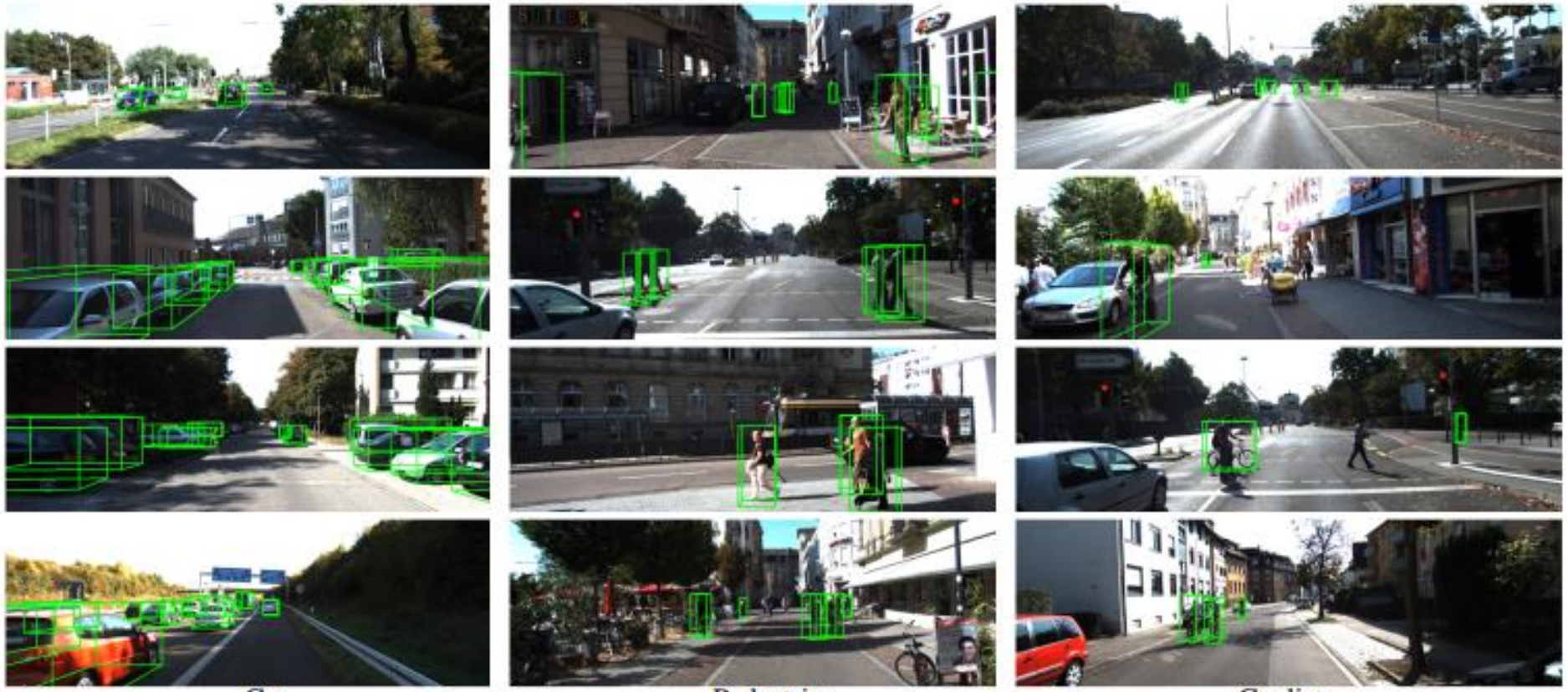


Figure 6. Qualitative results. For better visualization 3D boxes detected using LiDAR are projected on to the RGB images.