2023 졸업프로젝트 진행 보고서

주제: 3D Object Detection

2023. 03. 07. ~ 2023. 04. 04.

신상윤, 윤성우

진행 상황

* VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection 논문 1회독

3DSSD 논문 정리

LiDAR point cloud는 드문드문 공간이 많다(point에 비해 빈공간이 많다)

따라서 RPN(region proposal network 구역제안네트워크)에 넣기위해 노력을했다.

(point cloud에서 객체를 감지하고 지역화하기 위해 효율적인 표현을 개발했다)

대부분 RPN에 넣기 위해 hand-crafted feature(수제 특징)에 집중했다.

-> 머신러닝

풍부하고 상세한 3D 정보를 이용할수있을때에는 만족스러웠지만

더 복잡한 모양과 장면에 적응하지 못하고 data에서 필요한 변화를 학습하지 못했다.

본 연구에서는 end-to-end VoxelNet을 제안한다.

-> 딥러닝

1. point cloud를 동일한 간격의 voxel로 나누고 VFE layer을 누적하여 각 voxel을

인코딩한다. (end-to-end 방식으로 정확한 3D 경계 상자를 예측)

VFE : Voxel Feature Encoding

2. 3D convolution으로 point cloud를 고차원 volumetric representation으로 변환한다.

3. RPN으로 결과를 산출한다.

<VOXEL NET>

1. Feature Learning network

Voxel Partition: point cloud가 주어지면 같은 크기를 가지는 Voxel들로 나눈다.

Grouping: 점들이 위치한 voxel에 따라 그룹화 한다.

그룹화 후에는 복셀에 다양한 개수의 점들이 포함된다. 그림2 참고

Random Sampling: 점이 너무 많아 모든 포인트를 처리하기는 불가능

이를 위해서 T개 이상의 점을 포함하는 Voxel에서 T개의 점만 sampling

Voxel Feature Encoding : 그림2에서 한 Voxel에 대해 계층적으로 VFE Layer를 적용하는 것을 볼 수 있다.

일반성을 잃지 않고 VFE Layer - 1을 사용하여 설명한다. 그림3 참고

V: point t개를 가진 nonempty Voxel, t <= T

point는 xyz 좌표와 r(received reflectance)을 가짐

먼저 V의 local 평균을 구한다. (Vx, Vy, Vz)

다음 각 point를 다음과 같이 증가시켜 P\_hat = (Xi, Yi, Zi, Ri, Xi - Vx, Yi - Vy, Zi - Vz)

input feature를 얻는다. V\_in = {P\_hat\_i} i = 1~t

각 P\_hat은 FCN에 통과시켜 feature space, point-wise feature를 얻는다. f\_i

point-wise feature를 element-wise MaxPooling을 통해 locally aggregated feature를 얻는다. f\_틸다

마지막으로 f\_i, f\_틸다를 연접하여 output feature를 얻는다. f\_out = {f\_i, f\_틸다}

V\_out = {f\_out\_i} i = 1~t

모든 non empty Voxel에 대해 적용하고 FCN은 parameter를 공유한다.

output feature는 point-wise feature, locally aggregated feature를 모두 결합하므로 VFE layer를 쌓으면

shape information을 학습할 수 있다.

최종적으로 voxel-wise feature는 n번째 point-wise feature를 FCN 통과후 element-wise MaxPooling을 통과하면 얻을 수 있다.

Sparse Tensor Representation: 거의 90% Voxel이 비어있으므로 non-empty voxel만 sparse tensor로 표현한다.

2. Convolutional Middle Layers (이미지 처리에서 필터 연산과 비슷)

ConvMD(cin, cout, k, s, p)로 M차원 Convolution operator를 나타낸다.

cin, cout : input, output channels 개수 ex) RGB이미지에서 R,G,B가 각각 채널임

k: kernel size 커널 = 필터

s: stride size 필터 이동 횟수(다음 필터가 몇칸 이동하는지)

p: padding size 크기 조정을 위해 특정 값으로 채워줌

3D Conv -> BN layer -> ReLU layer를 순차적으로 적용

ConvMD는 voxel-wise features를 합치고 점진적으로 receptive field를 확장시키고,

shape description에 context를 추가한다.

3. Region Proposal Network

기존 RPN을 end-to-end 학습이 가능하도록 수정하였다.

RPN의 입력은 ConvMD에서 제공하는 feature map이다.

그림 4 참고

3개의 fully convolutional layers가 있다.

첫 layer는 stride 2로 downsampling, 나머지는 stride 1로 유지 이후 각각 BN, ReLU 연산 적용

그런 다음 모든 블록을 fixed size로 upsampling후 연접하여 high resolution feature map을 구성한다.

마지막으로 이 feature map은 probability score map, regression map에 매핑된다.

<Loss Function>

anchor box -> bounding box의 후보들, 최종 예측 박스를 선정해야됨

positive anchors -> 원하는 object가 있는 구역, 1로 학습

negative anchors -> 원하는 object가 없는 구역, 0으로 학습

bounding box의 parameter (x,y,z ,l,w,h, theta)

x,y,z : center location

l,w,h : box의 length, width, height

theta: z축에 대한 회전각

실제: 3D ground truth box -> 윗첨자 g

예측: anchor box -> 윗첨자 a

두가지 parameter로 loss function을 정의한다.

먼저 residual vector(잔차 벡터) U\* = [dx,dy,dz,dl,dw,dh,dtheta]를 정의한다.

loss function을 정의한다.

p\_pos, p\_neg: a\_pos, a\_neg에 softmax 적용

L\_cls: classification loss, binary cross entropy

L\_reg: regression loss, SMOOTHL1LOSS

alpha, beta: loss 비율을 조절하는 hyper parameter

<Efficient Implementation>

그림 5 참고

앞에서 설명한것 처럼 point cloud는 빈공간이 많아 빈 voxel도 많으므로 모든

voxel을 연산하면 효율적이지 않다.

따라서 stacked VFE로 nonempty voxel에 대해서만 voxel-wise feature를 얻었고,

voxel coordinate buffer를 통해 nonempty voxel의 좌표를 저장해놓고 마지막에

다시 mapping 해주었다.

K: nonempty voxel 수

T: 각 voxel이 가질 수 있는 최대 point수

7: input feature dimension

이 논문에서는 end-to-end로 학습 가능한 구조, sparse한 point cloud와 voxel을 병렬적으로 처리하는 효율적인 방법을 제안하였다. KITTI benchmark에서도 최상위의 성능을 보였다. 그러나 여전히 pedestrian과 cyclist에 대해서는 성능 개선의 여지가 남아 있다.