PAPER ANALYSIS

Presented by Yannis He

-

Paper: **VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection**

Published Date: 17 Nov, 2017

Authors: Yin Zhou, Oncel Tuzel | Apple.Inc

State-of-the-art LiDAR based 3D detection methods on the KITTI car detection benchmark

https://arxiv.org/abs/1711.06396

ABSTRACT

- Background
  - To interface a highly sparse LiDAR point cloud with a region proposal network (RPN), most existing efforts have focused on hand-crafted feature representations (Ex. Bird's eye view projection)
  - Hand-crafted features yield satisfactory results when rich and detailed 3D shape info is available.
    - But unable to adapt to more complex shapes and scenes, and learn require invariances from data
  - Image-based (2D input) 3D detection approaches are bounded by the accuracy of depth estimation
- Contribution:
  - Remove the need of manual feature engineering for 3D point clouds: VoxelNet, a generic 3D detection network that unifies feature extraction and bounding box prediction into a single stage, end-to-end trainable deep network.
    - Divides a point cloud into equally spaced 3D voxels
    - Transforms a group of points within each voxel into a unified feature representation through the newly introduced voxel feature encoding (VFE) layer.
      - I.e. point cloud is encoded as descriptive volumetric representation, which is then connected to a RPN to generate detections
  - State-of-the-art LiDAR based 3D detection methods on the KITTI car detection benchmark

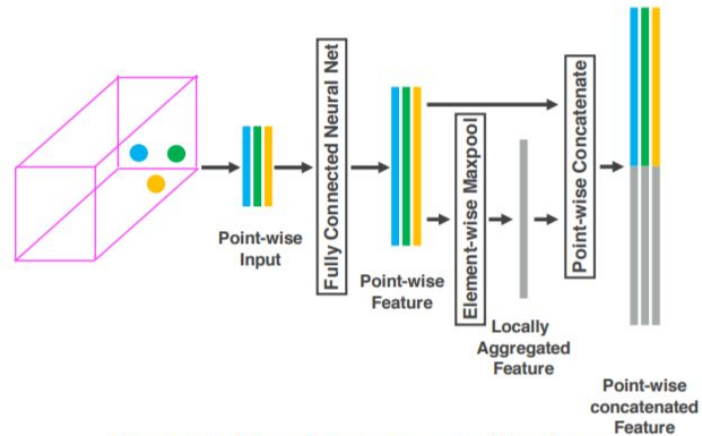


Figure 3. Voxel feature encoding layer.

- Background:
  - Comparing with images, LiDAR provides reliable depth information that can be used to accurately localize objects and characterize their shape
  - But LiDAR points are sparse and have highly variable point density due to factors as non-uniform sampling of the 3D space, effective range of sensors, occlusion, relative pose.
  - This results that many approaches manually crafted feature representations for point clouds that are tuned for 3D object detection.
    - Ex. project point clouds into a perspective view and apply image-based feature extraction.
    - However, manual design choices introduce an information bottleneck, which prevents effectively exploiting 3D shape information and the required invariance for detection task.
- Contribution:
  - 3D detection framework that simultaneously learns a discriminative feature representation from point clouds and predicts accurate 3D bounding boxes, in an end-to-end fashion.
    - Voxel feature encoding (VFE) layer, which enables inter-point interaction within a voxel by combining point-wise features with a locally aggregated feature
      - Stacking VFE layers allows learning complex features for characterizing local 3D shape information
  - VoxelNet divides point cloud into equally spaced 3D voxels, encodes each voxel via stacked VFE layers, and then 3D convolution further aggregates local voxel features, transforming the point cloud into a high-dimensional volumetric representation. Finally a RPN consumes the volumetric representation and yields the detection result.
    - Benefits both from sparse point structure and efficient parallel processing on voxel grid

ARCHITECTURE - VOXELNET

- 3 functional blocks:
  - Feature learning network
  - Convolutional middle layers
  - Region proposal network

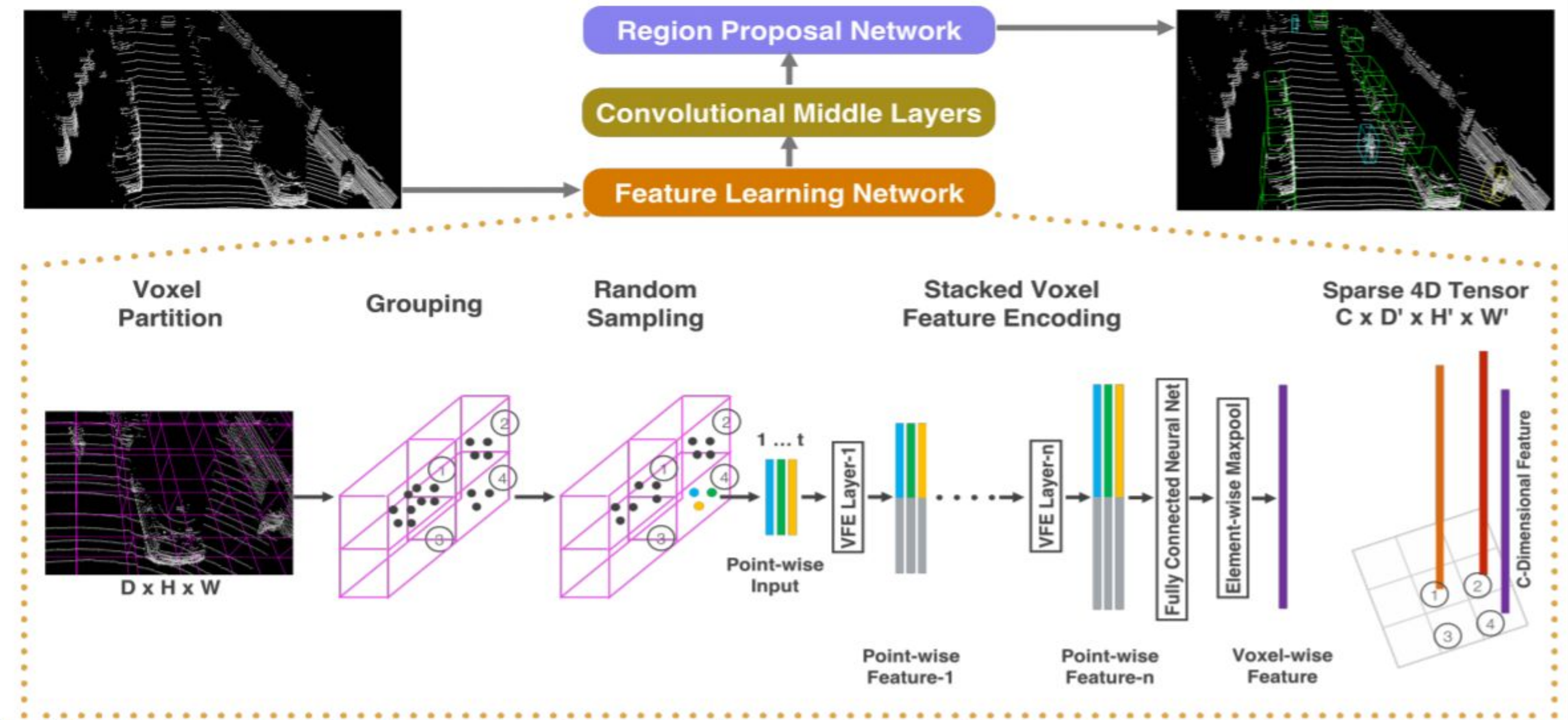
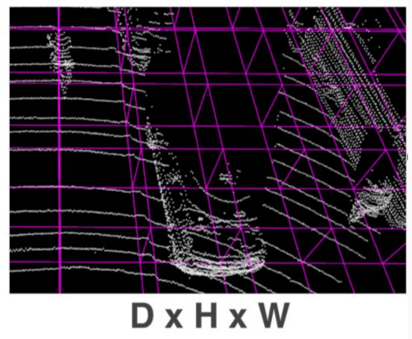


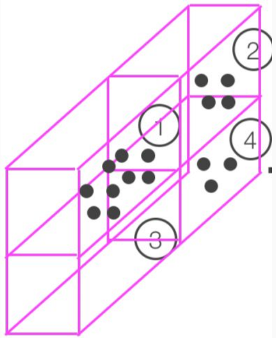
Figure 2. VoxelNet architecture. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. The convolutional middle layers processes the 4D tensor to aggregate spatial context. Finally, a RPN generates the 3D detection.

Architecture - VoxelNet: Feature Learning Network

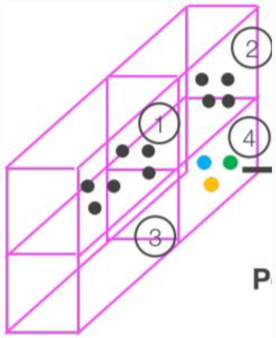
- 1st block: Feature learning network:
  - **Voxel Partition:**
    - Given a point cloud, subdivide 3D space into equalled spaced voxels.
  - **Grouping:**
    - Group points according to the voxel they reside in.
    - Since LiDAR point cloud is sparse and has highly variable point density, voxels will contain a variable number of points after grouping.
  - **Random Sampling:**
    - Randomly sample a fixed number, T, of points from voxels containing more than T points
      - 1) computational saving &
      - 2) decreases the imbalance (sampling bias) of points between voxels and add variation



Voxel Partition



Grouping



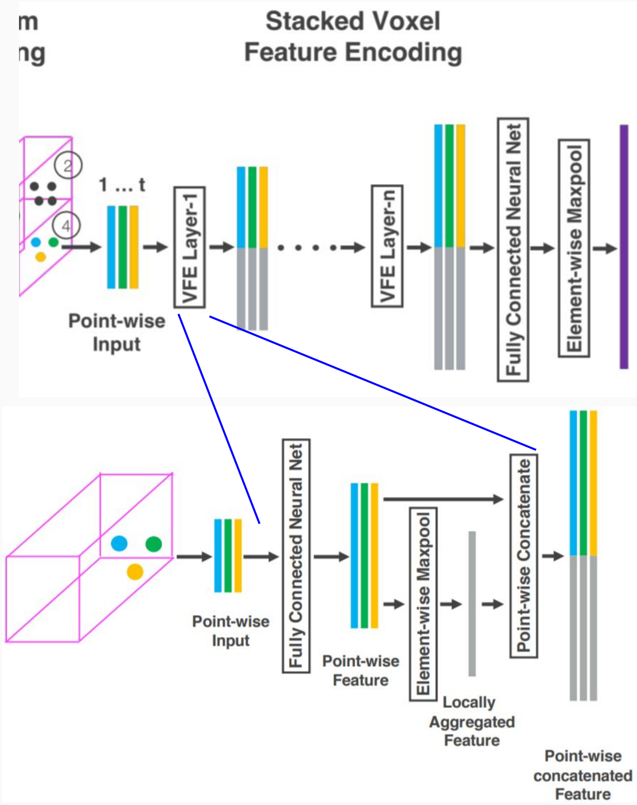
Random Sampling

ARCHITECTURE - VOXELNET: FEATURE LEARNING NETWORK (CONT')

- 1st block: Feature learning network (Cont'):
  - **Stacked Voxel Feature Encoding:**
    - The key innovation: the chain of VFE layers

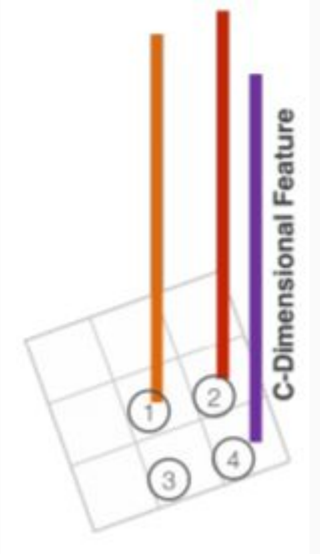
1. Compute local mean as the **centroid** (v_x, v_y, v_z) of all points in a voxel, $V = \{p_i = [x_i, y_i, z_i, r_i]^T \in R^4\}_{i = 1,\dots t}$
2. When augment each point with the relative offset w.r.t. to the centroid and **obtain input features** Vin = $\{\hat{p}_i = [x_i, y_i, z_i, r_i, x_i - v_x, y_i - v_y, z_i - v_z]^T \in R^7\}$i=1...t.
3. Each $\hat{p}_i$ is transformed through the fully connected network(FCN) into a feature space, where we can aggregate information from the point features $f_i \in R^m$ to encode the shape of the surface contained within the voxel. → **point-wise feature representation**
   a. FCN: linear layer, batch normalization (BN), ReLU
4. **Element-wise MaxPooling** across all f_i associate to V to get the locally **aggregated feature** $\tilde{f} \in R^m$ for V
5. Augment each f_i with $\tilde{f}$ to form the point-wise concatenated feature as $f_i^{out} = [f_i^T, \tilde{f}_i^T]^T \in R^{2m}$ → output feature: $V_{out} = \{f_i^{out}\}_{i = 1,\dots t}$

- Output feature combines both point-wise features and locally aggregated feature, stacking VFE layers encondes point interactions with a voxel and enables the final feature representation to learn descriptive shape information.



ARCHITECTURE - VOXELNET: FEATURE LEARNING NETWORK (CONT')

- 1st block: Feature learning network (Cont'):
  - **Sparse Tensor Representation::**
    - By processing only non-empty voxels, we obtain a list of voxel features
      - Each uniquely associated to the spatial coordinates of a particular non-empty voxel.
      - 90% of voxels typically are empty, out of ~100k points
    - The obtained list of voxel-wise features can be represented as a sparse 4D tensor, of size C x D' x H' x W' as shown in the figure
    - Representing non-empty voxel features as a sparse tensor greatly reduces the memory usage and computation cost during backpropagation..



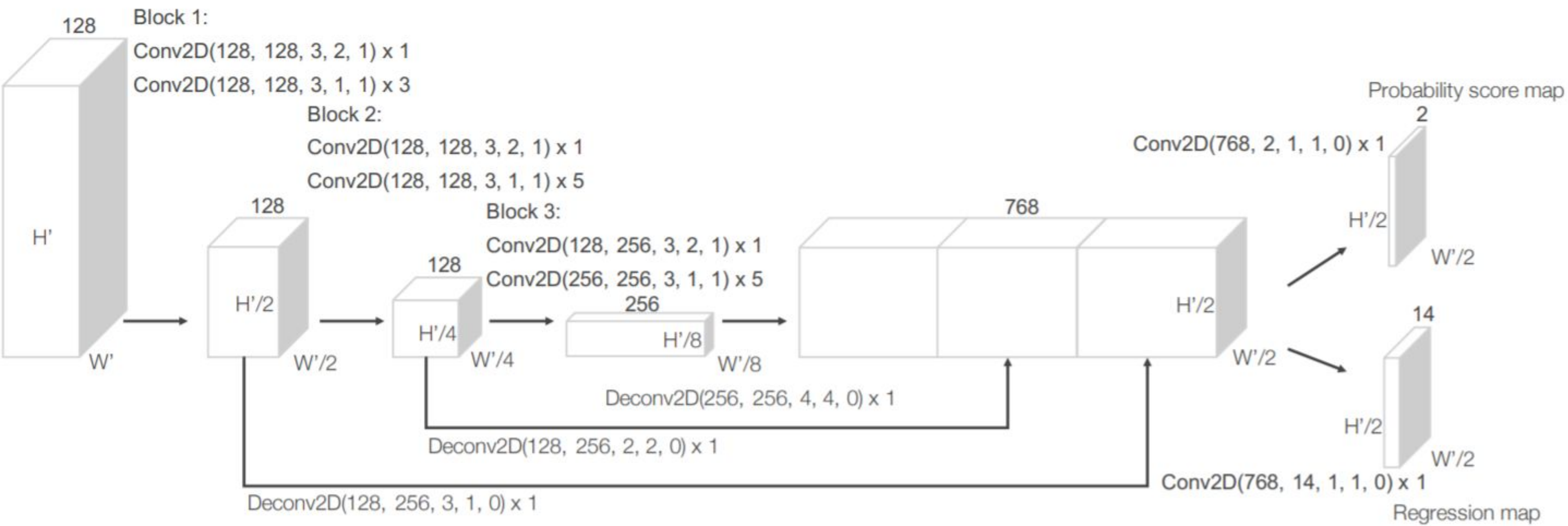
Sparse 4D tensor
C x D' X H' x W'

ARCHITECTURE - VOXELNET: CONVOLUTIONAL MIDDLE LAYERS

- 2nd block: Convolutional Middle Layers
  - ConvMD(c_{in}, c_{out}, k, s, p) is used to represent and M-dimensional convolution operator, where c_{in} and c_{out} are the number of input and output channels. k, s, p are the M-dimensional vectors corresponding to kernel size, stride size, and padding size respectively.
  - Each convolutional middle layer applies 3D convolution, BN layer, and ReLU layer sequentially.
    - Convolutional middle layers aggregate voxel-wise features within a progressively expanding receptive field, adding more context to the shape description.

ARCHITECTURE - VOXELNET: REGION PROPOSAL NETWORK

- 3rd block: Region Proposal Network
  - An important building block of top-performing object detection framework
  - In this work, RPN has several key modification and combined with the feature learning network and convolutional middle layers to form an end-to-end trainable pipeline
  - 3 blocks of fully convolutional layers:
  - Feature maps is mapped to the desired learning targets: 1) a probability score map & 2) a regression map



LOSS FUNCTION

- Notation:
  - Let $\{a_i^{pos}\}i=1...N_{pos}$ be the set of N_{pos} positive anchors & $\{a_i^{neg}\}i=1...N_{neg}$ be the set of N_{neg} positive anchors
  - 3D ground truth box, $(x_c^g, y_c^g, x_c^g, l^g, w^g, h^g, \theta^g)$
    - x_c^g, y_c^g, z_c^g represent center location
    - l^g, w^g, h^g are length, width, height of the box
    - θ^g is the yaw rotation around Z-axis
  - Positive anchor, $(x_c^a, y_c^a, x_c^a, l^a, w^a, h^a, \theta^a)$
  - Residual vector, $\mathbf{u}^* \in R^7$, containing 7 regression targets corresponding to
    - center location $\Delta x, \Delta y, \Delta z$
    - 3 dimension: $\Delta l, \Delta w, \Delta h$
    - Rotation $\Delta\theta$
  - Diagonal of the base of the anchor box: $d^a = \sqrt{(l^a)^2 + (w^a)^2}$
  - Loss function:

$$\Delta x = \frac{x_c^g - x_c^a}{d^a}, \Delta y = \frac{y_c^g - y_c^a}{d^a}, \Delta z = \frac{z_c^g - z_c^a}{h^a},$$
$$\Delta l = \log(\frac{l^g}{l^a}), \Delta w = \log(\frac{w^g}{w^a}), \Delta h = \log(\frac{h^g}{h^a}), \quad (1)$$
$$\Delta\theta = \theta^g - \theta^a$$

$$L = \alpha \frac{1}{N_{\text{pos}}} \sum_i L_{\text{cls}}(p_i^{\text{pos}}, 1) + \beta \frac{1}{N_{\text{neg}}} \sum_j L_{\text{cls}}(p_j^{\text{neg}}, 0) + \frac{1}{N_{\text{pos}}} \sum_i L_{\text{reg}}(\mathbf{u}_i, \mathbf{u}_i^*) \quad (2)$$

Where α, β are positive constants

    - p_i^{pos} & p_j^{neg} represent the softmax output for positive anchor a_i^{pos} and negative anchor a_j^{neg} respectively
    - $\mathbf{u}_i \in R^7$ and $\mathbf{u}^* \in R^7$ are the regression output and ground truth for postive anchor a_i^{pos}.
    - L_{cls}: binary cross entropy loss & L_{reg} is the regression loss (where SmoothL1 function is used)

RESULT

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**

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**

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

RESULT (CONT')

Car Pedestrian Cyclist

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

Benchmark	Easy	Moderate	Hard
Car (3D Detection)	77.47	65.11	57.73
Car (Bird's Eye View)	89.35	79.26	77.39
Pedestrian (3D Detection)	39.48	33.69	31.51
Pedestrian (Bird's Eye View)	46.13	40.74	38.11
Cyclist (3D Detection)	61.22	48.36	44.37
Cyclist (Bird's Eye View)	66.70	54.76	50.55

Table 3. Performance evaluation on KITTI test set.