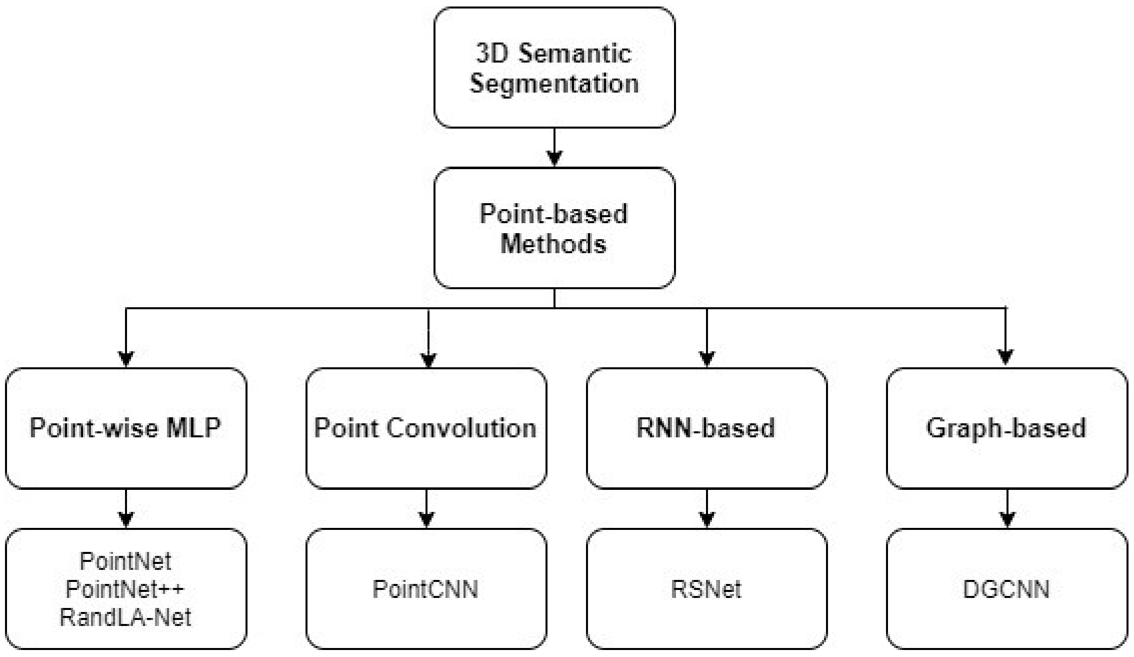
3D Point Cloud Semantic Segmentation Using Deep Learning Techniques

Rucha Apte
Dec 9, 2020 · 12 min read

Photo by Paul Green on Unsplash

Introduction

Point cloud learning has recently attracted attention due to the development of Augmented Reality / Virtual Reality and its wide applications in areas of computer vision, autonomous driving, and robotics. Deep Learning has been successfully used to solve 2D vision problems, however, the use of deep learning techniques on point clouds is still in its infancy because of the unique challenges faced for its processing. Earlier approaches in Deep Learning overcome this challenge by pre-processing the point cloud into a structured grid format at the cost of increased computational cost or loss of depth information. 3D point cloud segmentation is the process of classifying point clouds into different homogeneous regions such that the points in the same isolated and meaningful region have similar properties. 3D segmentation is a challenging task because of high redundancy, uneven sampling density, and lack of explicit structure of point cloud data. The segmentation of point clouds into foreground and background is a fundamental step in processing 3D point clouds. One can precisely determine the shape, size, and other properties of the objects in 3D data. However, segmenting objects in 3D point clouds is not a trivial task. The point cloud data is usually noisy, sparse, and unorganized. Apart from that, the sampling density of points is uneven and the surface shape can be arbitrary with no statistical distribution pattern in data. Moreover, due to limitations in 3D sensors, the background is entangled with the foreground. Additionally, it is difficult to have a deep learning model that is computationally efficient and has a low memory footprint to perform segmentation. The segmentation process is helpful for analyzing the scene in various applications like locating and recognizing objects, classification, and feature extraction. 3D point cloud segmentation can be deployed at scene level (semantic segmentation), object-level (instance segmentation), and part level (part segmentation). Semantic segmentation is a technique that detects for each pixel, the object category that it belongs to and also treats multiple objects of the same class as a single entity.

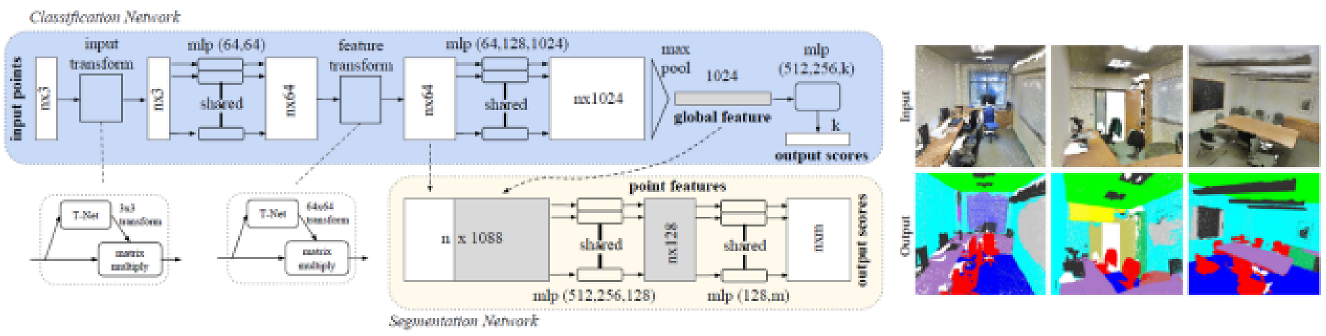


Taxonomy for 3D semantic segmentation using Point-based Methods

Given a point cloud, the goal of semantic segmentation is to separate it into several subsets according to the semantic meanings of points. The current article focuses on studying state-of-the-art semantic segmentation techniques under the point-based methods. The taxonomy for various point-based 3D semantic segmentation techniques can be given by 4 paradigms as (a) Point-wise MLP, (b)Point Convolution, (c)RNN-based, and (d) Graph-based.

PointNet

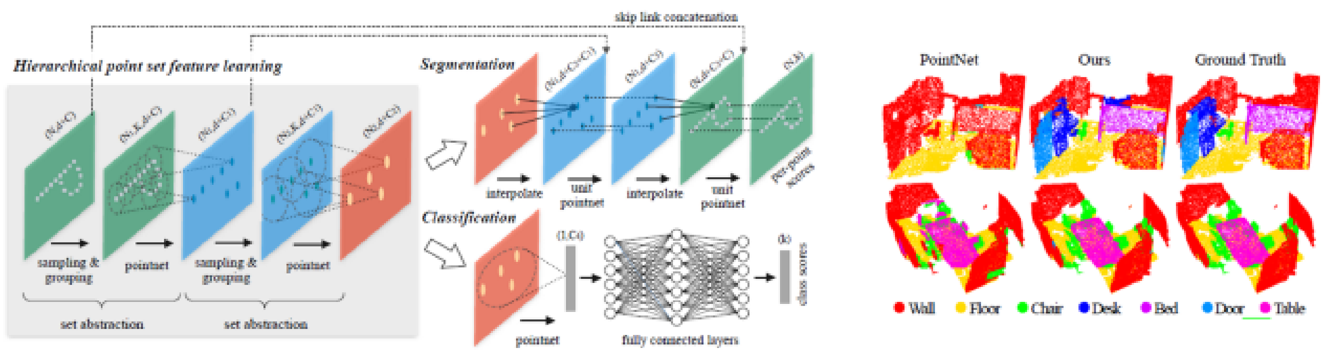
Convolutional architectures require highly regular input data formats in order to perform weight sharing and other kernel optimizations. Since point clouds and meshes are not in regular format, most methods typically transform the data into regular 3D voxel grids or collection of images before feeding them to a deep net architecture. However, this transformation renders the resulting data unnecessarily voluminous and also introduces quantization artifacts that can obscure the natural invariances of the data. PointNet directly consumes point clouds which respect the permutation invariance in the input. PointNet architecture contains three key modules: the max-pooling layer as a symmetric function to aggregate the information from all points, a local and global information combination structure, and two joint alignment networks that align input points and point features. In order to find the symmetry function for unordered input, a general function defined on the point set is approximated by applying a symmetric function on transformed elements. PointNet approximates a function by a multi-layer perceptron network and transformed function by a composition of single-variable function and a max-pooling function. The output of the function forms a vector which is considered as the global signature of the input set and is fed to per point features by concatenating the global features with each of the point features. Then, new per point features are extracted based on combined point features, as the per point will be aware of both the local as well as global information. The joint alignment network that forms the third module is inspired by the fact that the semantic labeling of the point cloud has to be invariant if the point cloud undergoes geometric transformations. PointNet predicts the affine transformation matrix by T-net architecture and directly applies this transformation to co-ordinates of input points. The T-net is composed of point independent feature extraction, max pooling, and fully connected layers. The transformation matrix in the feature space has a higher dimension. Hence, in order to optimize, a regularization term that constrains the feature transformation matrix to be close to the orthogonal matrix is added to softmax training loss. The figure gives the detailed architecture of the PointNet and the output of the semantic segmentation.



PointNet Architecture and Output of Semantic Segmentation

PointNet++

PointNet does not capture local structures induced by the metric space in which points are present, limiting its ability to recognize fine-grained patterns and generalizability to complex scenes. PointNet++ introduces a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set. By exploiting metric space distances, PointNet++ is able to learn local features with increasing contextual scales. PointNet++ partitions a set of points into overlapping local regions by distance metric of the underlying space. Similar to CNNs, it extracts local features capturing fine geometric structures from small neighborhoods, and these local features are further grouped into larger units and processed to produce higher-level features. This process is repeated until features of the whole point set are obtained. The design of PointNet++ addresses two issues: how to generate the partitioning of the point set, and how to abstract sets of points or local features through a local feature learner. While PointNet uses a single max-pooling operation to aggregate the whole point set, PointNet++ builds a hierarchical grouping of points and progressively abstract larger local regions along the hierarchy. This hierarchical structure is composed of a number of abstraction levels and at each level, a set of points is processed and abstracted to produce a new set with fewer elements. The abstraction layer is made of three layers: Sampling layer, Grouping layer, and PointNet layer. The Sampling layer selects a set of points from input points, which defines the centroids of local regions. The grouping layer then constructs local region sets by finding “neighboring” points around the centroids. PointNet layer uses a mini-PointNet to encode local region patterns into feature vectors. Details of architecture used for PointNet++ and its comparison with PointNet is illustrated in the figure.



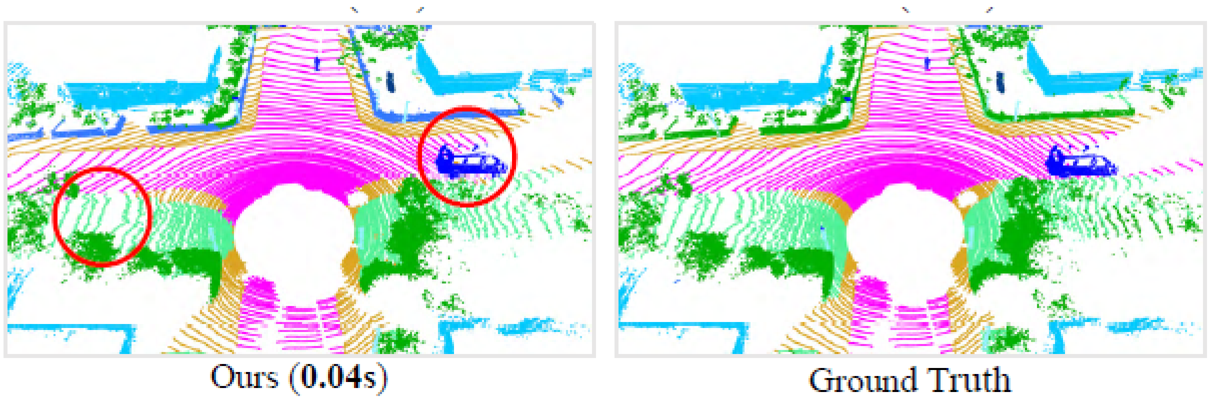
PointNet++ Architecture and Output of Semantic Segmentation

RandLA-Net

RandLA-Net introduces a lightweight neural architecture that can process large-scale point clouds which is 200 times faster than other architecture since most of the existing architecture is using expensive sampling techniques and computationally expensive post and pre-processing methods. PointNet is computationally efficient but fails to capture the context information of each point. RandLA-Net processes large-scale 3D point clouds in a single pass, without requiring any pre/post-processing steps such as voxelization, block partitioning, or graph construction. RandLA-Net only relies on random sampling within the network and hence requires much lesser memory and computation. The local feature aggregator obtains successively larger receptive fields by considering local spatial relations and point features. The entire network contains shared Multi-layered Perceptrons without relying on graph construction and kernelization and hence is efficient. Different sampling methods like Farthest Point Sampling, Inverse Density Importance Sampling, Generator-based Sampling are remarkably computationally efficient. However, they may result in the dropping of significant features. Hence, RandLA-Net proposes a local aggregation module. This module is applied to each 3D point in parallel and it consists of three neural units.

- LocSE: In this module given a point cloud, all the features are explicitly used to encode the three-dimensional coordinate information. It uses the K-Nearest Neighbours algorithm for gathering neighboring points and then the step of relative point position encoding is performed which tends to aid the network to learn local features. Finally, in the point feature augmentation, the encoded relative point position is concatenated with the corresponding point features and an augmented feature vector is obtained. This vector encodes local geometric structures.

- Attentive Pooling: For a given set of local features, a shared function is designed to learn a unique attention score and aggregate usage information in the neighborhood point feature set. The shared Multi-Layered Perceptron is then followed by a softmax function. These learned attention scores are then summed.
- Dilated Residual Block: Since Random Sampling continuously downsamples the input point cloud, it is necessary to increase the receptive field of each point. Using Resnet architecture and skip connections multiple LocSE and Attentive Pooling Blocks are connected to form Dilated Residual Block. This block is responsible for dilating the receptive field and expanding the effective neighborhood. Figure 4 gives a comparison of the outputs of RandLA-Net with PointNet++.



RandLA-NET Output of Semantic Segmentation in comparison with PointNet++

PointCNN

PointCNN is capable of leveraging spatially-local correlation in data represented densely in grids and provides a framework for feature learning from point clouds. PointCNN learns a χ-transformation from the input points to promote weighting of the input features associated with points and the permutation of points into canonical order. The architecture of PointCNN contains two designs:

1. Hierarchical Convolution: In regular grids, convolutions are recursively applied on local grid patches, which often reduces the grid resolution, while increasing the channel number. Similarly, in point clouds, χ-Conv is recursively applied to “project”,

or “aggregate”, information from neighborhoods into fewer representative points, but each with richer information.

2. χ-Conv Operator: χ-Conv operator operates in the local region and takes the associated points and neighborhood points as inputs and convolves with them. The neighboring points are transformed into local coordinate systems of representative points and later, these local coordinates are then individually lifted and combined with associated features.

A PointCNN with two χ-Conv layers transforms the input points into fewer feature representations but each with richer features. However, the number of training samples for the top χ-Conv layers drops rapidly, making it inefficient to train. To address this problem, PointCNN uses a denser connection, where more representative points are kept in the χ-Conv layers. Also, in order to maintain the depth of the network while keeping the receptive field growth rate, dilated convolutions are employed. For segmentation tasks, the high-resolution point-wise output is required, and hence Conv-DeConv architecture is used and follows the U-Net design.

Recurrent Slice Networks

Most of the other semantic segmentation networks do not model required dependencies between point clouds. The key component of the RSNet is a lightweight local dependency module. The local dependency module is highly efficient and has the time complexity of the slice pooling/unpooling layer as O(n) w.r.t the number of input points and O(1) w.r.t the local context resolutions. The RSNet takes as inputs raw point clouds and outputs semantic labels for each of them. Given a set of unordered points and a candidate label set, the task of the RSNets is to assign each of the points with a semantic label. The input and output feature extraction blocks are used for independent feature generation. In the middle is the local dependency module. The input feature block consumes input points and produces features and the output feature blocks take processed features as inputs and produces final predictions for each point. Both blocks use a sequence of multiple 1 x 1 convolution layers to produce independent feature representations for each point. The local dependency module is a combination of a novel slice pooling layer, bidirectional Recurrent Neural Network (RNN) layers, and a slice unpooling layer. The local context problem is solved by first projecting unordered points into ordered features and then applying traditional end-to-end learning algorithms. The projection is achieved by a novel slice pooling layer. In this layer, the inputs are features of unordered points and the output is an ordered sequence of aggregated features. Next, RNNs are applied to model dependencies in this sequence. Finally, a slice unpooling layer assigns features in the sequence back to points. The figure gives a detailed architecture of RSNets. RSNets are shown to surpass previous state-of-the-art methods on three widely used benchmarks while requiring less inference time and memory.

RSNet Architecture

Dynamic Graph CNN

DGCNN is an EdgeConv suitable for CNN-based high-level tasks on point clouds including classification and segmentation. EdgeConv acts on graphs dynamically computed in each layer of the network. It captures local geometric structure while maintaining permutation invariance. Instead of generating point features directly from their embeddings, EdgeConv generates edge features that describe the relationships between a point and its neighbors. EdgeConv is designed to be invariant to the ordering of neighbors, and thus is permutation invariant. Because EdgeConv constructs a local graph and learns the embeddings for the edges, the model is capable of grouping points both in Euclidean space and in semantic space. Instead of working on individual points like in PointNet, DGCNN exploits local geometric structures by constructing a local neighborhood graph and applying convolution-like operations on the edges connecting neighboring pairs of points. edge convolution (EdgeConv), has properties lying between translation-invariance and non-locality. Unlike graph CNNs, DGCNN's graphs are not fixed but rather dynamically updated after each layer of the network. That is, the set of k-nearest neighbors of a point changes from layer to layer of the network and is computed from the sequence of embeddings. The DGCNN can perform classification as well as segmentation tasks. The classification model takes as input n points, calculates an edge feature set of size k for each point at an EdgeConv layer, and aggregates features within each set to compute EdgeConv responses for corresponding points. The output features of the last EdgeConv layer are aggregated globally to form a 1D global descriptor, which is used to generate classification scores for cc classes. The segmentation model extends the classification model by concatenating the 1D global descriptor and all the EdgeConv outputs (serving as local descriptors) for each point. It outputs per-point classification scores for p semantic labels. The network contains two blocks:

1) Point cloud transform block: This block is designed to align an input point set to a canonical space by applying an estimated 3 × 3 matrices. To estimate the 3 × 3 matrices, a tensor concatenating the coordinates of each point and the coordinate differences between its k neighboring points is used.

2) EdgeConv block: This block takes as input a tensor of shape n × f, computes edge features for each point by applying a multi-layer perceptron (MLP) with the number of layer neurons, and generates a tensor of shape n × an after pooling among neighboring edge features. DGCNN architecture can be easily incorporated as-is into existing pipelines for point cloud-based graphics, learning, and vision.

DGCNN Architecture

Summary and Quantitative Results

The following table presents a quantitative analysis of results on 3 publicly available datasets S3DIS, Semantic3D, ScanNet(v2), and Sem. KITTI using evaluation metrics OA, mIOU.

Quantitative Results

References

[1] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," IEEE transactions on pattern analysis and machine intelligence, 2020.

[2] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11 108–11 117.

[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.

[4] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in Advances in neural information processing systems, 2017, pp. 5099–5108.

[5] Y. Li, R. Bu, M. Sun, W.Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in Advances in neural information processing systems, 2018, pp. 820–830.

[6] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3d segmentation of point clouds," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2626–2635.

[7] A. V. Phan, M. Le Nguyen, Y. L. H. Nguyen, and L. T. Bui, "Dgcnn: A convolutional neural network over largescale labeled graphs," Neural Networks, vol. 108, pp. 533–543, 2018.

Deep Learning Point Cloud Segmentation Self Driving Cars Pointnet