Dynamic Graph CNN for Learning on Point Clouds in Master Seminar 3D-Machine Learning Proceedings

Finn Rasmus Schäfer
Technische Universität Munich
Boltzmannstrasse 15
85748 Garching
finn.schaefer@tum.de

Abstract

The present work deals with a learning-based approach for learning on point clouds. The approach presented is an extension of existing works and shows how to apply deep learning on point clouds while keeping local neighborhood information. The work shows why it is important and how to gain local neighborhood information. The outcome of the paper is a usable CNN layer that is capable of dealing with high-level classification and segmentation tasks. The novel approach is compared to state-of-the-art networks, and it is explained why they perform significantly better.

1. Introduction and Motivation

In Computer Vision, especially in camera-based computer vision, deep learning has become one of the most important features in recent years. When working on images, deep learning has completely replaced handcrafted or non-learning-based approaches. Due to the good results and the fast-developing field of learning-based approaches, scientists are aiming to also perform deep learning on point clouds, which can be extracted from laser scanners and other sensors. However, there are some difficulties: point clouds are not permutation invariant, continuously distributed, and completely irregular. Point clouds need to be processed differently compared to images [1], [2], [7]. Some approaches like PointNet [6] are processing every point individually, while others prefer different techniques [5], [3], [4]. The PointNet approach, which basically computes every point individually, is not capable of keeping local neighborhood information. When processing every point individually, you lose spatial relationships and local context. This directly leads to the loss of important geometric information. Because this information matters - which was a fundamental outcome of the presented work and is introduced later - further research like the provided work was done to preserve this information. By setting up a graph and using convolution-like approaches, the authors were capable of making use of the local geometry and improving tasks like 3D object segmentation and classification. The upcoming sections will explain what EdgeConv is and how it basically works.

2. Background & Related Work

In this section, related work and other approaches are presented. Before deep learning was introduced for images and point clouds, there were many non-learning-based approaches used to perform tasks on point clouds. These approaches required handcrafted feature extractors mostly based on edge and corner detection. The handcrafted approaches often extracted low-level features that are not really needed for modern approaches like autonomous driving. Since processing the point cloud by performing mesh reconstruction or other approaches is very expensive, the basic idea of deep learning on point clouds is to directly process the point cloud and extract high-level features. For these high-level segmentation and classification tasks, one approach that sought to be extended by the dynamic graph CNN (DGCNN) for learning on point clouds paper is the PointNet approach. The weakness of this approach is that the point cloud is processed directly, but every point on its own. This leads to information loss on local geometric data. The PointNet++ approach is more capable of capturing spatial relationships. The PointNet network was the main base work and benchmark model for the authors of the paper. That is why the PointNet and PointNet++ approaches are shortly explained in the following.

2.1. PointNet

The PointNet approach revolutionized the field of 3D point cloud analysis by providing a powerful framework that is capable of directly learning from raw point cloud data. Directly processing a point cloud enables end-to-

end learning without requiring pre-segmentation or other pre-processing steps. PointNet is able to learn local point-wise features that can be aggregated and summarized for the whole point cloud. The approach is permutation invariant because of processing each point individually. It is a common approach for real-world scenarios, for example, in classification and segmentation.

2.2. PointNet++

PointNet++ shares similarities with PointNet while introducing crucial differences. Instead of processing every point individually, PointNet++ introduces a hierarchical architecture for the points. This architecture organizes the points into local regions, which basically enables the network to learn local and global contextual information. These regions are named "point set hierarchy".

3. EdgeConv Approach and Results

EdgeConv makes use of graph CNNs. Instead of processing every point of the point cloud individually, the whole point cloud is processed at the same time. Also, the basis of this approach, the graph that represents the point cloud, is updated after each layer of the network. Although different types of graphs can be used, the most common graph is a k-nearest neighbor (kNN) graph. The relevant features are extracted from the edges, which are designed to capture local geometric structures. The edge features enable the extraction of meaningful information from the unordered point set. Edges are defined by the kNN graph, where a single edge is a point and one of its k neighboring points. For every edge of the graph, an edge feature is computed. Relevant features can be properties like distances or angles between the two neighboring points. To effectively exploit the edge features, EdgeConv employs a learnable convolutional operation. This operation aggregates information from neighboring edges and updates the features of each point. Specifically, it applies a convolutional filter to the edge features, allowing the network to learn complex patterns and relationships among adjacent points. As mentioned in the Introduction, permutation invariance is a highly wanted feature in this case. The permutation invariance in this approach is ensured by using symmetric functions for the aggregation of neighborhood information. These symmetric functions can be max or average pooling. The aggregation of edge features results in a single feature representation for every single point. Due to the symmetric functions, it is guaranteed that the output is independent of the ordering of the input points, which basically ensures permutation invariance. When repeatedly applying EdgeConv, it is progressively able to capture local geometric structures. This essentially enables a deep neural network to learn hierarchical representations of point clouds. In a network, each layer can encode different complex and abstract features. In the following, the EdgeConv and the dynamic graph update are investigated more precisely.

Method	Linear Function	Nonlinear Function
PointNet	None	$h_\Theta(x_i, x_j) = h_\Theta(x_i)$
PointNet++	max pooling	$h_\Theta(x_i, x_j) = h_\Theta(x_i)$
PCNN	$\sum$ pooling	$h_{\theta_m}(x_i, x_j) = \theta_m \cdot x_j) g(u(x_i, x_j)$

Table 1. This table shows how to choose the linear function (Aggregation) and nonlinear function (Edge function) to make the dynamic graph CNN approach act like existing approaches.

3.1. EdgeConv

Although EdgeConv is capable of working on higher-dimensional point clouds, in this example, a simple 3D point cloud is assumed. Higher orders of point clouds could also hold information like color, depth, or surface normals. If we assume a 3D point cloud with a specific number of points (n), we can describe the point cloud as $X = x_1, x_2, ..., x_n$. Based on this data, the directed graph is generated. This graph ($\mathcal{G}$) has edges ($\mathcal{E}$) and vertices ($\mathcal{V}$). Because each point of the point cloud is a vertex, we also get n vertices. Every edge feature (e_{ij}) is calculated via a nonlinear function (h_θ) that has a set of learnable parameters (θ). The input to this nonlinear function is the current point (x_i) and one of its neighboring points (x_j), mathematically resulting in $e_{ij} = h_\theta(x_j, x_i)$. After applying the nonlinear function to all edges of the source point, the computed edge features are aggregated to the point using a symmetric function (Placeholder for a symmetric function: $\square$). The aggregated feature for a point x'_i can be described as $x'_i = \underset{j:(i,j)\in\mathcal{E}}{\square} h_\theta(x_i, x_j)$. This calculation is visually displayed in figure 1. By choosing the symmetric and the nonlinear function, we can influence the output and what the EdgeConv basically describes. Under specific circumstances, we can let EdgeConv work like other approaches such as PointNet, PointNet++, MoNet, and PCNN. The needed symmetric function and the needed nonlinear function for these special use cases can be seen in table 1. If there is no aggregation, EdgeConv can process every point individually. Although this is a valid option, this is not very functional because the same problems as in the PointNet approach occur.

3.2. Classification Results

One of the main tasks that can be processed with the EdgeConv Layer is the classification approach. The classification takes the point cloud as input and outputs a classification score. The specific structure for classification can be seen in the upper branch in figure 3a. Each EdgeConv Layer is used to extract local geometry. An EdgeConv Layer consists of a multi-layer perceptron that has the kNN tree as input and an aggregation function. For classification tasks,

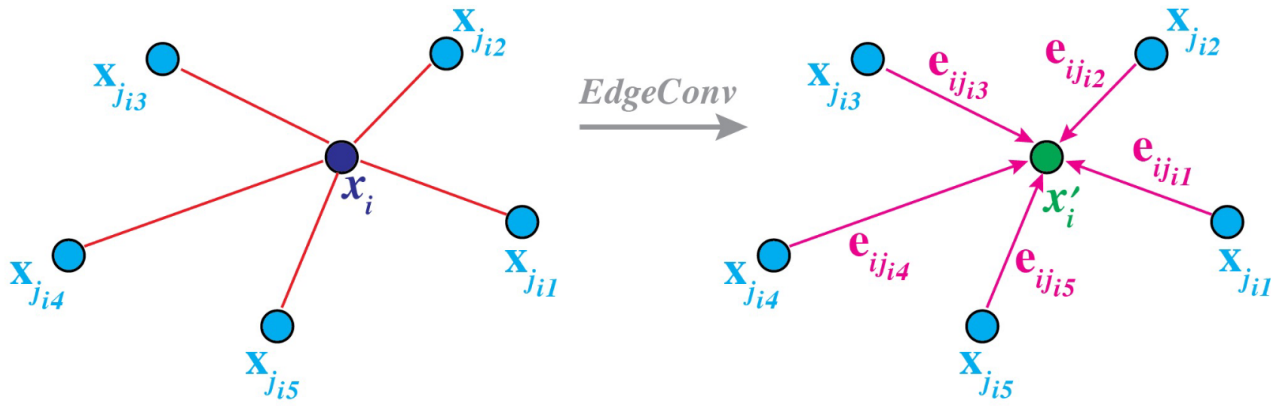


Figure 1. This figure shows how EdgeConv changes the graph structure and visually displays all the parameters described in section 3.1. The source point x_i', its neighboring points x_j, and its edge features e_{ij} are visible. The aggregation of the edge features (the result of the symmetric function applied to the edge features) is displayed in green.

the k of the kNN tree is equal to 20. The architecture also uses skipping connections. After concatenating the features of the previous layers, the global point cloud feature is extracted via a max pooling function. The last two fully connected layers, which use a dropout probability of 0.5, are used to transform the global point cloud feature. For the whole network, a Leaky ReLU function and batch normalization are applied. The classification results can be compared to other approaches. In table 2, mean class accuracy and overall accuracy are compared to other existing models. A conclusion is provided in section 4.

3.3. Segmentation Results

For semantic and part segmentation a different architecture is used. The input is the same as for the classification, but the output differs for scene and part segmentation. For scene segmentation a probability distribution over the classes is assigned. This means every point is assigned with a probability value for each class that represents the likelihood of the element belonging to the class. By having a probability distribution, it becomes possible to consider multiple plausible class assignments for each element, which can be valuable in cases where there is ambiguity or overlapping regions in the scene. For part segmentation there is typically point or region labeling. This depends on the granularity of the task. In part segmentation, each point or region is assigned a class label (categorical vector) that indicates the specific part or object it corresponds to. The architecture for segmentation tasks can be seen in the bottom branch of figure 3a.

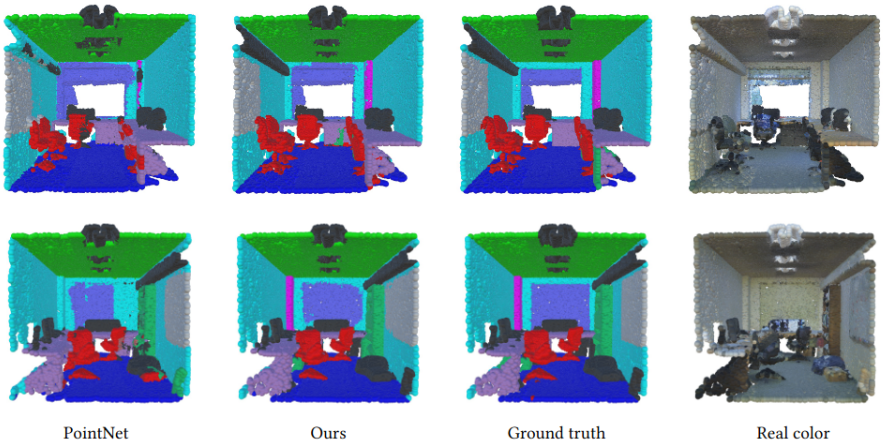
The provided branch efficiently captures spatial relationships and contextual information among the point cloud points.This leads to accurate and detailed segmentation results. The approach employs hierarchical architectures to refine the initial segmentation results. The network leverages the graph information to propagate and refine predictions across different levels of the hierarchy. This contextual refinement process enables the model to incorporate contextual cues and global information to improve the segmentation accuracy. The results for scene segmentation (displayed in figure 2a) and part segmentation (displayed in figure 2b) are benchmarked on tables 3 and 4.

	Mean Cass Accuracy	Overall Accuracy
3DShapeNets	77.3	84.7
VoxNet	83.0	85.9
SUBVOLUME	86.0	89.2
VRN (SINGLE VIEW)	88.98	-
VRN (MULTIPLE VIEWS)	91.33	-
ECC	83.2	87.4
PointNet	86.0	89.2
PointNet++	-	90.7
KD-Net	-	90.6
PointCNN	88.1	92.2
PCNN	-	92.3
DGCNN (BASELINE)	88.9	91.7
DGCNN	**90.2**	**92.9**
DGCNN (2048 POINTS)	**90.7**	**93.5**

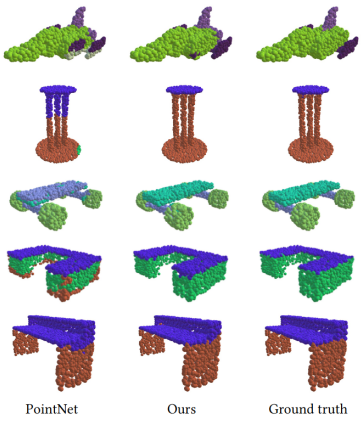
Table 2. Comparison of classification results for 3D point cloud data shows that the dynamic graph CNN approach using EdgeConv achieves the best results in both mean class accuracy and overall accuracy. The dataset used for comparison was the ModelNet40.

4. Conclusion & Discussion

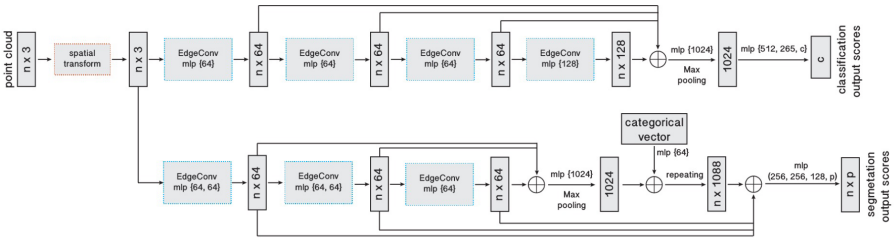
The DGCNN, especially the results, show that local geometric information is relevant when working with 3D point clouds. By introducing a way of capturing local geometric features, it outperforms the PointNet approach, which is not capable of doing this, in every task. This basically shows



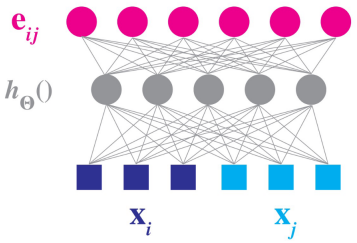
(a) This figure displayes the two branches for 3D classification (upper branch) and 3D segmentation (lower branch).



(b) In this picture, the computation of the graph feature is displayed in a CNN layer-like structure. Using the nonlinear activation function h_θ, the input data (points x_i, x_j) and the output (edge features e_{ij}) are shown.



(a) This figure displays the two branches for 3D classification (upper branch) and 3D segmentation (lower branch).



(b) In this picture, the computation of the graph feature is displayed in a CNN layer-like structure. Using the nonlinear activation function h_θ, the input data (points x_i, x_j) and the output (edge features e_{ij}) are shown.

	Mean Cass Accuracy	Overall Accuracy
PointNet (BASELINE)	20.1	53.2
PointNet	47.6	78.5
MS + CU(2)	47.8	79.2
G + RCU	49.7	81.1
PointCNN	**65.39**	-
DGCNN	56.1	**84.1**

Table 3. Comparison of scene segmentation results using the 3SDIS Dataset shows that the DGCNN using EdgeConv has the best overall accuracy. Although it does not have the best Mean IoU, it outperforms the main benchmark model PointNet. Results are visualized in figure 2a.

	MEAN	CHAIR	BAG	CAP	CAR	MUG
# Shapes		3758	76	55	898	184
PointNet	83.7	89.6	78.7	82.5	74.9	93.0
PointNet++	85.1	**90.8**	79.0	**87.7**	77.3	94.1
PCNN	85.1	**90.8**	80.1	85.5	79.5	94.8
PointCNN	**86.1**	90.6	**86.4**	86.0	**80.8**	**95.3**
DGCNN	85.2	90.6	83.4	86.7	77.8	94.9

Table 4. This table shows the comparison between state of the art part segmentation models and the DGCNN. Allthough the presented approach gets outperformed by PointCNN, it delivers better mean results than the PointNet and PointNet++ networks, which was the real benchmark for the approach. Results are visualized in figure 2b.

that by capturing the local geometric features of individual points and their spatial relationships, the model gains a more comprehensive understanding of the underlying structure and geometry of the point cloud data. Especially for segmentation, we can derive that local geometric information plays a crucial role in distinguishing and characterizing different object classes in point clouds. It shows that geometric features, such as point positions, normals, curvatures, or local surface descriptors, carry discriminative information that aids in accurate segmentation and classification. The local information enables the model to consider local structures and surface properties, allowing better

decisions. Additional studies in the original paper, which were not part of this report, also showed that using local neighborhood information leads to more robustness to noise and incomplete data. In conclusion, the paper's findings advanced the field of 3D point cloud analysis by introducing the EdgeConv layer, which is capable of learning local geometric features that lead to the points mentioned in the beginning. The approach can be advanced by changing the graph used for capturing spatial relations between points or by looking at more than two points at a time. The paper delivers a good base for further research.

References

[1] R. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. Learning continuous semantic representations of symbolic expressions. *arXiv preprint arXiv:1702.08398*, 2019. 1

[2] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. 1

[3] Y. Li, R. Bu, M. Sun, and B. Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018. 1

[4] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointconv: Deep convolutional networks on 3d point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 1

[5] C. Qi, L. Yi, H. Su, and L. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5099–5108, 2017. 1

[6] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pages 652–660, 2017. 1

[7] X. Zhou, Y. Wu, S. Zhu, and Z. Zhang. Permutation invariant neural networks. *arXiv preprint arXiv:1511.01844*, 2015. 1