MASTER IASD

POINT CLOUD AND 3D MODELING

Object DGCNN: 3D Object Detection using Dynamic Graphs

Realized by:

BENCHEIKH LEHOCINE Mohammed Amine

mohammed-amine.bencheikh-lehocine@dauphine.eu

Supervised by:

Mr. François Goulette

Mr. Louis Soum-Fontez

2022/2023

Table of Contents

List of Figures

1 Introduction

3D object detection is a fundamental challenging problem in computer vision that has gained widespread interest in both the research community and industry due to its diverse applications [QLW^{+}18] in self-driving cars, robotics and augmented reality. the main objective of this task is to find bounding boxes from which it is possible to identify the precise location, size, and orientation of 3D objects in a scene which is generally represented as a point cloud captured by a LIDAR Sensor. However, achieving accurate and efficient 3D object detection is highly challenging and difficult since it requires significant domain knowledge, strongly dependent on the dataset and should be very efficient and effective in the time since most applications require real-time inference like self-driving cars.

Modern 3D object detectors typically use multi-stage pipelines that involve several sequential steps for training and testing. These pipelines can be complex and involve hand-crafted post-processing steps to achieve high performance. Post-processing steps may include object confidence aggregation or NMS, which are crucial for removing redundant boxes. However, these steps can be tedious, require expert knowledge to design and optimize and are difficult to parallelize. Recent works such as PointPillars [LVC^{+}19], PillarOD [WFK^{+}20], and CenterPoint [YZK21] simplify the detection pipeline by introducing one-stage anchor-based methods, but they still predict redundant boxes.

Furthermore, In the field of 2D object detection, recent works based on transformers, such as the DETR method [CMS^{+}20], employ a set of bounding box predictions, which eliminates the need for post-processing steps like non-maximum suppression (NMS). However, these approaches have not yet been widely adopted in the field of 3D object detection.

Yue Wang and Justin Solomon's research [WS21] focuses on developing a 3D object detector model that can achieve a high inference frequency with state-of-the-art performance. They aim to eliminate the need for post-processing techniques such as object confidence aggregation or non-maximum suppression. Object DGCNN seeks to address these challenges by combining ideas from one-stage anchor-based models, DETR, and DGCNN on sparse point clouds. Specifically, Object DGCNN projects sparse point clouds on BEV maps and then uses a pre-trained backbone (like PointPillars [LVC^{+}19]) to derive useful 2D feature maps then a dynamic graph to represent the relationships between the points in the point cloud, and performs message passing to update the features of each point. Then use a set-to-set loss to learn bounding boxes without redundancy.

Overall, Object DGCNN represents an important advancement in the field of 3D object detection, as it combines ideas from several approaches to overcome the challenges associated with traditional multi-stage pipelines and post-processing steps.

2 Object detection in 3D setting

2.1 Basic concepts

3D object detection is a computer vision task that involves detecting and localizing objects in 3D scenes constructed from given sensory inputs, such as images or point clouds. The main objective is to predict the attributes of 3D objects [MSWL22], including their location, size, shape, and orientation, in a given scene. It is an important technology for various applications, such as autonomous driving, robotics, and augmented reality, where accurate detection of 3D objects is essential for decision-making and interaction with the environment.

More formally, we can write a general formula for this task as follows :

$$B = F_{det}(I_{inputs})$$

such that F refers to the model, I_{inputs} refers to one or more sensor inputs (points clouds from LIDAR, RADAR data, images ...).

The results are a set of bounding boxes

$$B = \{b_1, b_2, b_3, ..., b_m\}$$

where each b_i contains the following information:

$$b_i = \{x_i, y_i, z_i, l_i, w_i, h_i, \theta_i, class_i\}$$

Where (x_i, y_i, z_i) are the coordinates of the object centre, (l_i, w_i, h_i) refers to the length, width, and high respectively. And we also have the heading angle θ and the object's class.

2.2 Data representation for 3D object detection

point clouds present unique challenges for feature extraction and therefore object detection in a 3D environment, especially in the context of autonomous driving. While convolutional neural networks (CNNs) have proven highly effective for image recognition tasks due to the fact that pixels are regularly distributed on an image plane, they may not be the best fit for these types of data which is sparse, unordered, and irregular. Many previous works in the literature tried to tackle this problem, we can see three main streams :

2.2.1 Point-based approach

Generally, methods based on this approach try first to pass the raw point cloud through a backbone network, which learns features from the points and gradually reduces the dimensionality of the data. The backbone network typically consists of a sequence of point cloud operators, such as PointNet [QSMG17] or PointNet++ [QSMG17]. These operators sample a subset of points from the point cloud and learn useful features. Bounding boxes for 3D objects are then predicted with a prediction head using these features and the downsampled points.

This approach is computationally expensive and requires large amounts of memory to process large point clouds which is prohibitive in the context of real-time applications.

2.2.2 Grid-based approach

The grid-based approach takes a different way to point-based methods by first rasterizing the point cloud data into a discrete grid representation. This grid can take several forms, including voxels, pillars, or bird's-eye view (BEV) feature maps [QLL22]. It is worth mentioning that pillars are a special case of voxels (simply, voxels without limit in the vertical direction) and BEV features maps are generally obtained from the pillars or voxels representation of point clouds. In the work of [WS21], they tested both methods (although they focused in their explanation only on the pillar-based method).

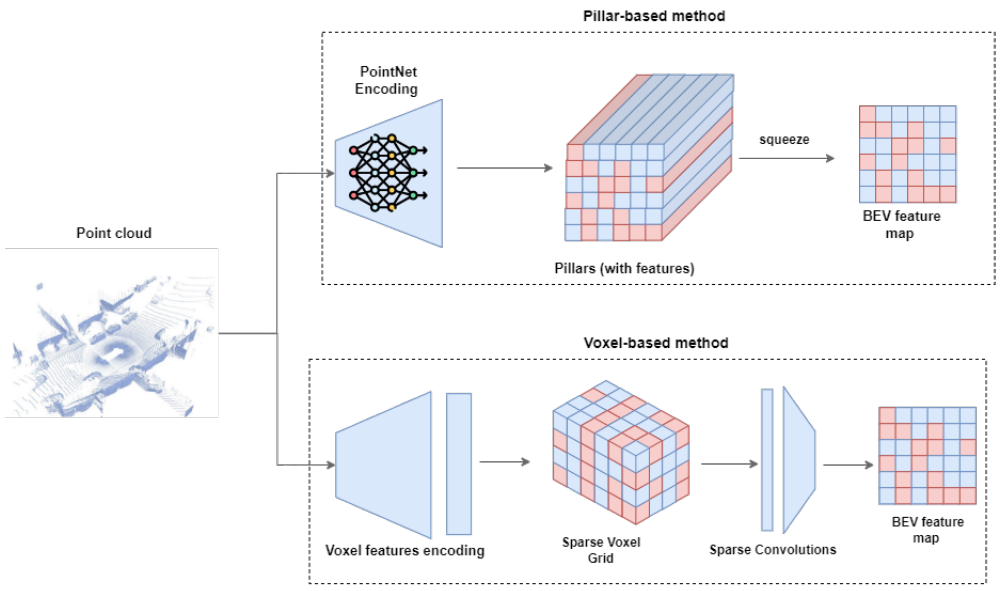

Figure 1: Grid-based approaches

2.2.3 Point-Voxel based approach

A hybrid approach combining both Point and voxels to leverage 3D object detection. methods based on this approach can be devised in two categories or frameworks: the one-stage framework and the two-stage framework. The difference resides in how to combine or use both voxel and point representations; for the one-stage methods, they combine them together in the feature extraction stage. for the two stages framework, they use voxel-based representations to generate regions proposals, and they refine these proposals by key points sampled from the points representations. further explanation can be found in [QLL22].

3 Object DGCNN

3.1 Overview of the method

The authors of object DGCNN [WS21] designed previously another model for recognition, classification, and segmentation from points clouds. They claim that it can not be applied directly for the object detection task for mainly two reasons :

- The point clouds related to 3D object detection contain too many points which makes DGCNN too expensive in terms of computation and memory.
- The output set of bounding boxes is too small compared to the input set of points.

To overcome these two limitations that make the adaptation of DGCNN architecture to object detection impractical, the authors first added a component to learn an intermediate representation of the points cloud: they tested two feature extractions pipelines (Pillar-based and voxel-based pipelines, already explained in the previous section) to construct a Bird's eye view representation of the input.

From the resulting BEV representations, object DGCNN generates a set of bounding boxes by applying local features extraction operators (inspired from DGCNN [WS21]) and k-nearest Neighbors aggregation. They used a permutation invariant set to set loss to optimize only this part of the model (for the construction of the BEV representation, they used directly pre-trained models).

3.2 Bird's eye view representations

The first step as we explained above is to take the point cloud $X = \{x_1, x_2, ..., x_N\}$, which is a set of points in 3D space and scatter them into either pillar Bird's eye view grid or 3D voxels. Next, we apply some 2D convolutions to extract features on this grid. The authors as we said before used two methods:

- Using the same method as in [WFK$^+$20] by Pillars and applying 2D convolution to extract local features on BEV maps.
- Using SparseConv which conducts 3D sparse convolutions to refine the voxel-wise features and then compress them to get BEV feature maps.

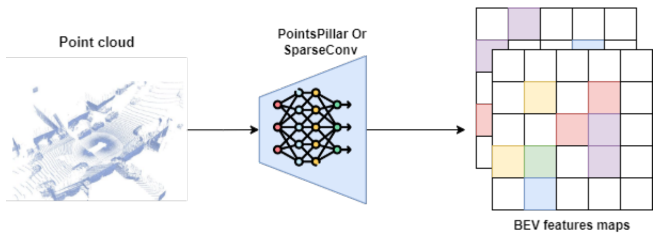

Figure 2: Point clouds scattered on BEV with features maps

3.3 Object DGCNN architecture

Although the name of the method refers to Dynamic graph architecture, it is mainly influenced by the work of [CMS$^+$20] which first introduced the use of an attention mechanism to detect objects in 2D images and used the set-to-set loss to get rid of the need for post-processing NMS or confident aggregations. The model is composed of a number of layers (L) where three steps are performed each forward pass :

- Predict a set of query points (intuitively, they represent objects in the scene) and attention weights to the K neighbours in the BEV feature map.
- collect features from BEV maps (using learnt weights in the previous step for each query).
- model intra objects interactions (represented by queries) using K-nearest neighbours graph.

As mentioned above, in each layer we need to do three predictions: the object's centre point and offsets to its K nearest neighbours and weights for each neighbour feature. They use three neural networks (shared between all queries). Once we have the neighbour's coordinates in The BEV map, a combination of their features is stacked to a vector representing each query's features. Using these vectors, we can construct a graph based on the nearest neighbours to each query (or object, if we consider our previous intuition). We use here EdgeConv operator [WSL+19] to extract features related to interactions between objects and update our queries to pass to the next layer.

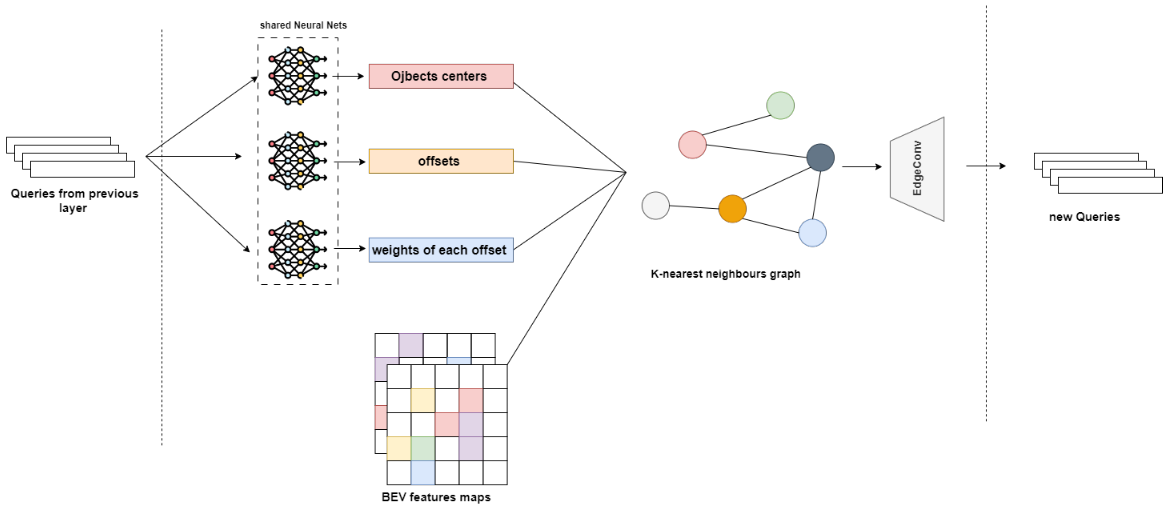

Figure 3: A simplified illustration of one layer from object DGCNN

3.3.1 Detection head

After L layer of object DGCNN, we forward the resulting queries vectors to two networks: a classification network to predict the label class of the object and a regressor network to predict the bounding box parameters (centre, offsets, heading angle . . .).

3.3.2 Set-to-Set loss

let M be the number of objects present in the ground truth of the scene. generally, the number of queries in object DGCNN output is bigger than M, let's note it $\tilde{M}$. We complete the set of ground-truth with a 'no object' special sign to have the same number of elements in both sets. Before we compute 'the famous object detection loss' that most models try to minimize, a matching problem is solved using the Hungarian algorithm to find the best permutation between the two sets to match the predictions [WS21]. we denote this permutation by δ^*.

$$L_{sup} = \sum_{i=1}^{N} -log(\tilde{p}_{\delta^*(i)}(c_i)) + 1\{c_i! = \emptyset\}L_{box}(b_i, \tilde{b_i})$$

the first term represents the loss of classification and the second one relates to the bounding box if the predicted object is not empty.

4 Experiments and possible ameliorations

4.1 Experiments

The authors claimed that Object DGCNN outperforms other state-of-art models like CenterPoint, PointPillar, Free Anchor and PillarOD in the nuscenes benchmark for both metrics: mean average precision mAP and nuscenes score (NDS).

We can observe some drawbacks in their reported results :

- They did not report the inference frequency for each method, although it is crucial to compare the efficiency of removing the NMS post-processing step.
- They did not reported tests on other benchmarks and datasets like KITTI. It is possible that object DGCNN performance is due to some biases present in the nuscenes dataset.

In the ablation study, the authors demonstrated the performance of using dynamic graphs over multi-head attention. same as before, we don't have any comparison in terms of computation cost and time. For distillation, several configurations were conducted depending on the type of backbone used to scatter BEV representation (between the teacher and student network using a voxel-based extractor or Pillar-based one). the results demonstrate the utility of applying knowledge distillation in training time and data.

To test object DGCNN and reproduce part of the results that had been shown in the paper, we used the tool MMdetection3D from the platform OpenMMLab the tool designed to manage, prepare and test 3D object detection on common benchmarks and datasets. In our case, because we have limited time and computing resources, we tried only to test pre-trained models (based on Pillars and based on voxels) on the mini-dataset version from the nuscenes large 3D detection dataset.

Some results from the Mini version of Nuscenes dataset :

Table 1: global metrics results

Metric	Value
mAP	0.4957
mATE	0.4381
mASE	0.7691
mAOE	1.2331
mAVE	0.4206
mAAE	0.3005
NDS	0.4550
Eval time	3.7s

Table 2: per-class Object Detection Results

Object Class	AP	ATE	ASE	AOE	AVE	AAE
car	0.878	0.184	0.756	1.487	0.090	0.060
truck	0.668	0.161	0.781	1.521	0.067	0.000
bus	0.991	0.213	0.871	0.768	0.642	0.153
trailer	0.000	1.000	1.000	1.000	1.000	1.000
construction_vehicle	0.000	1.000	1.000	1.000	1.000	1.000
pedestrian	0.852	0.212	0.317	1.581	0.175	0.191
motorcycle	0.688	0.314	0.825	1.291	0.056	0.000
bicycle	0.398	0.193	0.800	1.449	0.334	0.000
traffic_cone	0.480	0.105	0.341	nan	nan	nan
barrier	0.000	1.000	1.000	1.000	nan	nan

4.2 Future possible ameliorations

Object DGCNN is a promising 3D object detection framework that has been shown to perform well in previous sections. We suggest here some possible future ameliorations that could be added to:

- Multimodel data sources: is it possible to modify the preliminary step of extracting useful features from cloud points to deal with other modalities such as RGB images, LiDAR intensity, or radar data to improve the accuracy of the object detection?
- Use an architecture similar to DGCNN to extract features from points clouds with a few layers (to reduce computation cost) and try to have a DGCNN-only based architecture for object detection.
- The authors did not train the whole model from scratch; they used a pre-trained backbone for feature extraction, a possible way of improving the results is by training the whole model at the same time from scratch.
- Another possible way of improving this model is by using other 3D feature extractors both at the first step and during the construction of the dynamic graph.

5 Conclusion

In conclusion, the Object DGCNN model provides a highly efficient and accurate solution for 3D object detection by eliminating the need for post-processing operations and leveraging dynamic graphs and a set-to-set loss function. To further improve the model, there are several promising future directions that could be explored. One possible direction is to modify the preliminary step of extracting useful features from point clouds to incorporate multimodal data sources such as RGB images, LiDAR intensity, or radar data. By incorporating these additional data sources, the model could potentially improve its accuracy and robustness in detecting objects in complex and diverse environments.

Overall, a lot of future directions present exciting opportunities for advancing the field of 3D object detection using neural network architectures. By incorporating these ideas, it may be possible to further improve the accuracy, efficiency, and robustness of 3D object detection models, leading to more effective applications in areas such as autonomous driving, robotics, and augmented reality.

References

[CMS+20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.

[LVC+19] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[MSWL22] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: a review and new outlooks. *arXiv preprint arXiv:2206.09474*, 2022.

[QLL22] Rui Qian, Xin Lai, and Xirong Li. 3d object detection for autonomous driving: a survey. *Pattern Recognition*, 130:108796, 2022.

[QLW+18] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[QSMG17] 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.

[WFK+20] Yue Wang, Alireza Fathi, Abhijit Kundu, David A Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 18–34. Springer, 2020.

[WS21] Yue Wang and Justin M Solomon. Object dgcnn: 3d object detection using dynamic graphs. *Advances in Neural Information Processing Systems*, 34:20745–20758, 2021.

[WSL+19] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[YZK21] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.