LEARNED GRIDIFICATION FOR EFFICIENT PROCESSING OF POINT CLOUDS

Anonymous authors
Paper under double-blind review

ABSTRACT

We provide a method for point cloud classification using `Conv3D` by learning a point cloud representation that lives on a low-resolution dense 3D grid. The gridification step consists of a bipartite graph convolution, which connects the input point cloud to a target grid in 3D. We show that the method achieves remarkable performance given the network size and grid resolution on which the `Conv3D` blocks operate.

1 INTRODUCTION

Point clouds often represent a particular geometric structure together with a signal over this structure. E.g., a point cloud could serve as a sparse representation of a surface in 3D or as a representation of a conformation of a given molecule with points being the atom locations and a vector representation of the atoms as signal. The sparse signal viewpoint allows for an intuitive generalization of convolution type operators, which process the signal via feature value transformations conditioned on the geometry, i.e., through kernels sampled on pair-wise geometric attributes such as relative position.

Whereas convolutions on images, which are signals on regular grids, allow for highly optimized parallel processing by reusing a fixed set of kernel values, this is not possible for the irregularly sampled signals that the point clouds represent. The lack of a regular grid structure for point clouds severely limits parallelization. A potential solution lies in treating the point cloud as a density on $\mathbb{R}^3$, that can be sampled/discretized on a dense regular grid, a process called voxelization. Voxelization methods thus create regular lattice representations on which Conv3D methods can act. However, such methods are expensive, since capturing fine details requires high resolution grids, which scales poorly. Furthermore, due to the sparse nature of point clouds, occupancy of the voxel grid is generally low, which makes such methods unnecessarily expensive and unscalable.

In this paper we explore a solution to the computational inefficiency of point cloud methods, whilst still being able to faithfully pick up on geometric detail. Our simple recipe consists of a *bipartite graph convolution* step that maps the sparse point cloud representation to a dense grid in $\mathbb{R}^3$ through k-nearest neighbour connectivity. Consequently, this grid representation can now be processed using efficient dense convolution methods such as Pytorch's `Conv3D`. We show that the 3D grid resulting from the bipartite message passing step can be of very low resolution, circumventing high cost of capturing fine details in voxelization methods.

2 RELATED WORKS

Point cloud classification has widely been studied in machine learning literature (Sun et al., 2009; Bronstein & Kokkinos, 2010; Aubry et al., 2011). Where classical approaches often relied heavily on (task-specific) feature engineering methods, deep learning-based approaches were introduced only recently, due to the complexity of defining classical deep learning operators (i.e. convolution, pooling) for the irregular data domain of pointclouds (Qi et al., 2017a).

Most of these deep learning methods for pointcloud processing can be classified into two approaches: 1) methods that perform voxelization of the original pointcloud to obtain a regular structure on which classical CNN architectures for regular data can then be applied (Maturana & Scherer, 2015; Qi et al., 2016; Le & Duan, 2018; Wang & Lu, 2019) or 2) methods that operate directly on the irregular point cloud (Qi et al., 2017a;b; Wu et al., 2019; Engel et al., 2021).

Whereas voxel-based approaches allow for straightforward application of convolutional operations and architectures, in practice the voxelization step leads to loss of detail in fine geometry. Since the voxel resolution required to faithfully capture the sparse but detailed nature of pointcloud is high in most settings, additional engineering of the voxel grid is needed. For instance, Le & Duan (2018) propose a voxelization method that is able to capture finer geometrical details by replacing the conventional geometry-invariant aggregation step in voxelization by a concatenation operation of the features that occupy a given voxel. However, as the number of points per voxel can vary, this requires a quantization or subsampling method which still results in information loss. Other methods such as Klokov & Lempitsky (2017); Riegler et al. (2017); Liu et al. (2022a) construct hierarchical voxel representations that adapt resolution to pointcloud density and encode a pointcloud at different resolutions, but these methods are hard to implement for arbitrary pointclouds since they make use of complicated data structures.

Methods that operate on point-clouds directly often consist of operators that are locally invariant to specific geometrical properties of the point cloud. For instance, Qi et al. (2017a;b) apply feature transformations to individual points, aggregating information from different points mainly through geometry-invariant max pooling operations, discarding fine-grained structure. On the other hand, architectures that incorporate local structures, e.g. through (graph) convolutions (Wu et al., 2019; Zhao et al., 2019; Thomas et al., 2019; Mao et al., 2019; Wang & Solomon, 2021) respect local geometry but are only able to apply local operations due to the computational overhead required for evaluating convolutions on irregular domains.

Most related to the proposed approach are methods like Liu et al. (2019); Zhang et al. (2021), falling somewhere in between the two aforementioned classifications. In Liu et al. (2019) authors propose a hybrid architecture that contains both a coarse voxel branch for capturing high-level structures with convolution operations and a pointwise multi-layer perceptron for modelling fine-grained structure. Zhang et al. (2021) leverage the same two-branch structure, but replace the convolution by a windowed self-attention operation to improve performance. In order to solve the problem of low occupancy in voxel grids, a rule book stores the entries of non-empty voxels. Although these methods incorporate both local and global information, their explicit use of voxelization operations still leads to a trade-off in information loss and computational efficiency. Instead, we propose a single-branch architecture similar to classical CNNs, which replaces voxelization by a bipartite graph convolution to obtain a regular feature grid. Since this operation does not make any use of geometry-invariant aggregation or quantization, this should reduce fine-grained information loss.

3 METHOD

Consider a set of N points sampled from the surface of a 3D object, consisting of positions $\mathbf{x}_i \in \mathbb{R}^3$, possibly with some corresponding feature vectors $\mathbf{f}_i \in \mathbb{R}^{C_f}$ (such as surface normals, RGB-values, etc.) associated to every position. A point cloud then forms a set $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^N$ or $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{f}_i)\}_{i=1}^N$ if feature vectors are available. Such a representation is sparsely located in $\mathbb{R}^3$. We seek a representation of this shape that lives on a regular grid in $\mathbb{R}^3$, so that we may further process it using efficient convolution methods like Conv3D.

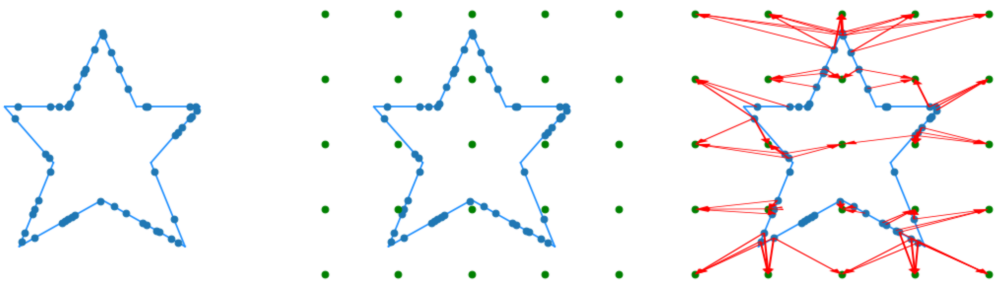


Figure 1: Example in 2D: 1) a point cloud is a set of locations sampled $\mathbf{x}^s$ from the surface of a shape. 2) Define target grid $\mathbf{x}_j^t$ in $\mathbb{R}^2$. 3) Connect point cloud to target grid via k-nearest neighbours.

In order to do so, we construct a graph that associates each position on a 3D regular grid with k positions in the source point cloud through k-nearest neighbour connectivity. In essence, if we consider the *source* point cloud set $\mathcal{S} = \{(\mathbf{x}_i^s, \mathbf{f}_i^s)\}_{i=1}^{N_s}$ and the *target* positions on the grid the set $\mathcal{T} = \{\mathbf{x}_j^t\}_{j=1}^{N_t}$, this constitutes a *bipartite graph* where the edges are directed from $\mathcal{S}$ource to $\mathcal{T}$arget.

In order to generate feature vectors $\mathbf{f}_i^t$ on the target grid positions $\mathbf{x}_i^t$, we employ a **bipartite graph convolution** as in Nassar (2018):

$$\mathbf{f}_i^t = \square_{j \in \mathcal{N}(i)} \phi_e(\mathbf{f}_j^s, \mathbf{a}_{ij}) \qquad \text{with } \mathbf{a}_{ij} = \phi_a(\mathbf{x}_i^t - \mathbf{x}_j^s)$$

Where ϕ_e is some differentiable message function that is conditioned on an embedding of the relative offset of the input point to the target point and, if available, an input feature $\mathbf{f}_j^s$, and $\square$ is some aggregation method, such as mean or max. In the absence of $\mathbf{f}_j^s$, the aggregation yields a purely geometric neighborhood embedding. The grid positions, along with the target feature vectors $\mathbf{f}_i^t$ together induce a representation of the point cloud that lives on a grid, $\mathcal{T} = \{(\mathbf{x}_j^t, \mathbf{f}_j^t)\}_{j=1}^{N_t}$.

4 Experiments

We test the model on the ModelNet10 and ModelNet40 point cloud classification datasets and empirically study several open design choices.

4.1 Point Cloud Classification

Point cloud classification for ModelNet10 and ModelNet40 results can be found in Table 1 and comparison with a PointNet++ baseline, a seminal model in point cloud classification, using the same optimisation setup. We apply a subsampling of 1000 points for each object and use the Adam optimizer Kingma & Ba (2014) with a learning rate of 1e-4. Surface normals are used as feature vectors, and all objects are normalized between [-1, 1]. All models were trained for 100 epochs and results were averaged over three seeds. Our model approaches the baseline with significantly less parameters, and for the smallest model, the `Conv3D` blocks operate on a 3D grid of *only* $3 \times 3 \times 3$ on the interval $[-1, 1]^3$. Each target grid position is connected to its 9 nearest input neighbours. The pipeline can be found in Fig. 2, where we used $L = 7$ convolution blocks and $k = 3$ and $k = 5$ for the small and best model, respectively. Further implementation details can be found in Appendix A.

Model	#Params	Accuracy	Runtime
PointNet++	1.5M	$87.11 \pm .005$	1h1m
PointNetNeXt++	1.5M	$87.67 \pm .412$	1h3m
Ours 5x5x5	351K	$86.02 \pm .282$	46m
Ours 3x3x3	263K	$84.27 \pm .738$	36m
ModelNet40			
PointNet++	1.5M	$88.39 \pm .014$	21m
PointNetNeXt++	1.5M	$90.44 \pm .230$	23m
Ours 5x5x5	351K	$89.62 \pm .367$	12m
Ours 3x3x3	263K	$86.94 \pm .183$	12m
ModelNet10			

Table 1: Shape classification results. Two versions of the proposed model were tested, one that operates on a resolution 3 grid and one on a resolution 5 grid.

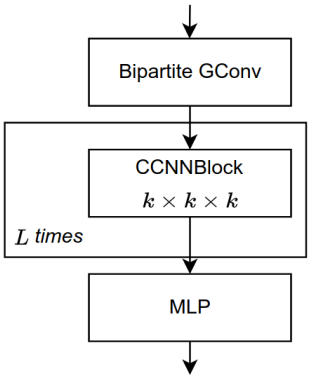
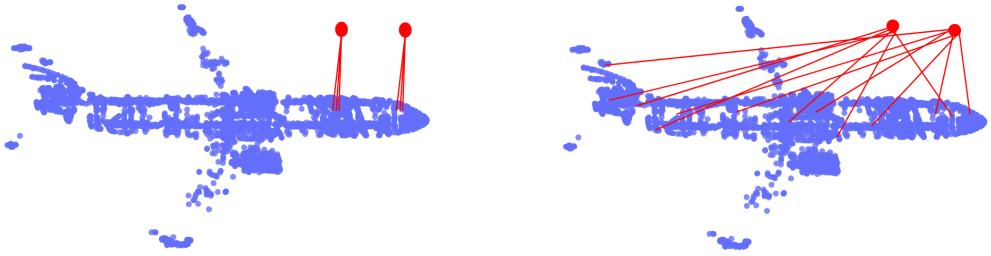


Figure 2: Architecture

4.1.1 Connectivity & Receptive Field

A free architectural choice is the connectivity of the input pointcloud to each target grid point: using radius connectivity would result in a method that is akin to voxelization methods. In our

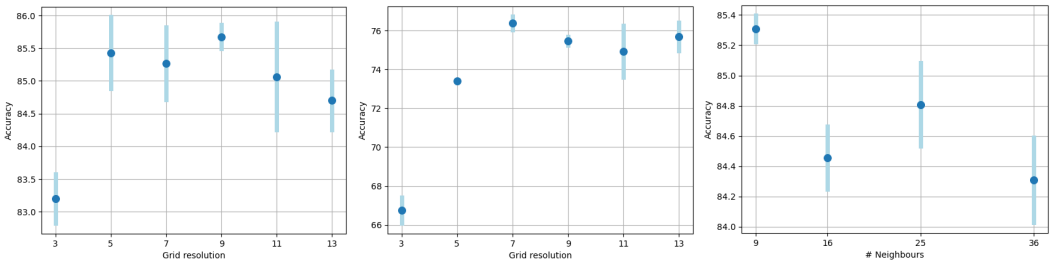
experiments, we use a K-nearest neighbour approach to ensure that each target grid position is populated with feature values. Choosing the amount of neighbours then corresponds to the effective receptive field of each target grid position. We experiment with two models differing in size of the receptive field, namely a **local model** in which each target grid position is connected to its n neighbours, and a **global model**, which has a global receptive field: each target grid point is connected to n randomly sampled input points. In our experiments, for the global model we used $n = 100$. Results can be found in Fig. 3.



(a) Local model (b) Global model

4.1.2 Grid Resolution

We also study the effect of increasing the grid resolution. In practice we see that we can operate at extremely low resolution. Results can be found in Fig. 3 a, b.



(c) Local model (d) Global model (e) Local model

Figure 3: a & b: Effect of increasing the grid resolution for the local and global models. c: Effect of increasing the receptive field for the local model.

5 Discussion & Conclusion

We provide a method for point cloud processing that achieves remarkable performance given its size and simplicity. Although the method does not yet outperform their PointNet++ counterpart, our experiments show the potential of a learnable voxelization module that improves overall efficiency whilst maintaining expressive capacity. As such, we believe the method provides a promising direction combining the expressivity of message passing graph neural networks for irregular domains and efficient and highly optimized lattice convolution methods.

References

Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pp. 1626–1633. IEEE, 2011.

Michael M Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 1704–1711. IEEE, 2010.

Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. *IEEE Access*, 9: 134826–134840, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE international conference on computer vision*, pp. 863–872, 2017.

David M. Knigge, David W. Romero, Albert Gu, Efstratios Gavves, Erik J. Bekkers, Jakub M. Tomczak, Mark Hoogendoorn, and Jan-Jakob Sonke. Modelling long range dependencies in n-d: From task-specific to a general purpose cnn, 2023. URL https://arxiv.org/abs/2301.10540.

Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9204–9214, 2018.

Juncheng Liu, Steven Mills, and Brendan McCane. Rocnet: Recursive octree network for efficient 3d processing. *Computer Vision and Image Understanding*, 224:103555, 2022a.

Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022b. URL https://arxiv.org/abs/2201.03545.

Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1578–1587, 2019.

Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 922–928. IEEE, 2015.

Marcel Nassar. Hierarchical bipartite graph convolution networks, 2018.

Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2016.

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*, pp. 652–660, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017b.

Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3577–3586, 2017.

Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pp. 1383–1392. Wiley Online Library, 2009.

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6411–6420, 2019.

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

Zongji Wang and Feng Lu. Voxsegnet: Volumetric cnns for semantic part segmentation of 3d shapes. *IEEE transactions on visualization and computer graphics*, 26(9):2919–2930, 2019.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 9621–9630, 2019.

Cheng Zhang, Haocheng Wan, Shengqiang Liu, Xinyi Shen, and Zizhao Wu. Point-voxel transformer: An efficient approach to 3d deep learning. *CoRR*, abs/2108.06076, 2021. URL `https://arxiv.org/abs/2108.06076`.

Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5565–5573, 2019.

A Appendix: Architectural details

For the convolutional blocks, we used 3D CCNN blocks as in Knigge et al. (2023). For the edge attribute $\mathbf{a}_{ij}$ we use a Random Fourier Feature embedding on the relative offset of neighbor and target node, since this embedding provides explicit control over the kernel smoothness through an initial frequency parameter σ. We applied sweeps on the kernel smoothness σ. Finally, we used PyTorch's ReduceLROnPlateau learning rate scheduler.

For the baseline models, we used a standard PointNet++ and a PointNet++ where the graph MLP components are replaced by graph ConvNeXt modules Liu et al. (2022b).