MuGNet: Multi-Resolution Graph Neural Network for Segmenting Large-Scale Pointclouds

Liuyue Xie
Carnegie Mellon University
United States
liuyuex@andrew.cmu.edu

Tomotake Furuhata
Carnegie Mellon University
United States
tomotake@andrew.cmu.edu

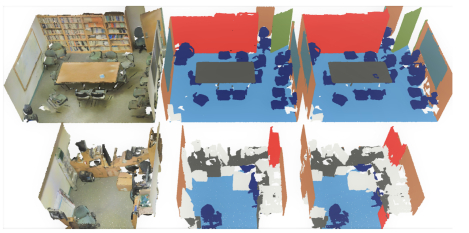
Kenji Shimada
Carnegie Mellon University
United States
shimada@cmu.edu

Abstract: In this paper, we propose a multi-resolution deep-learning architecture to segment dense large-scale pointclouds semantically. Dense pointcloud data require a computationally expensive feature encoding process before semantic segmentation. Previous work has used different approaches to drastically downsample from the original pointcloud to utilize common computing hardware. While these approaches can relieve the computation burden to some extent, they are still limited in their processing capability for multiple scans. We present **MuGNet**, a memory-efficient, end-to-end graph neural network framework to perform semantic segmentation on large-scale pointclouds. We reduce the computation demand by utilizing a graph neural network on the preformed pointcloud graphs and retain the segmentation's precision with a bidirectional network that fuses feature embedding at different resolutions. Our framework has been validated on benchmark datasets, including Stanford Large-Scale 3D Indoor Spaces Dataset(S3DIS) and Virtual KITTI Dataset. We demonstrate that our framework can process up to 45 room scans at once on a single 11 GB GPU while still surpassing other graph-based solutions for segmentation on S3DIS with an 88.5% (+3%) overall accuracy and 69.8% (+7.7%) mIOU accuracy.

Keywords: Machine Learning, Pointcloud, Semantic Segmentation, Bidirectional Neural Network, Graph Neural Network

1 Introduction

Semantic segmentation of large-scale 3D pointcloud data has attracted numerous interests in real-world applications. Automatic sight inspection for quality verification, for example, is one of the emerging applications for pointcloud semantic segmentation. Deployment of a 3D pointcloud semantic segmentation algorithm can avoid delays and reduce costs caused by human errors. Figure 1 shows how accurately our proposed multi-resolution graph neural network can perform and semantic segmentation.



(a) Raw input (b) Ground truth (c) MuGNet result

Figure 1: Sample segmentation results on S3DIS [1] Area 3.

When it comes to dense large-scale pointcloud scans, the current methods either require a significant reduction in the data density [2, 3, 4, 5, 6, 7] or offer only limited processing capability for multiple scans [8, 9, 10, 11, 12]. Two challenges hinder the development of pointcloud semantic segmentation algorithms on dense large-scale pointcloud scans.

The first is the extensive memory usage for processing dense pointclouds with up to billions of data points in a single scan. Prior research has attempted to downsample from the original pointcloud to the state that common computing hardware can be utilized. Performing drastic downsampling on dense pointcloud can, however, negatively impact the segmentation result on dense pointcloud. Intricate details initially present in the dense pointcloud input are removed during the downsampling process. This is undesirable since sparse pointclouds contain less geometric features. The segmentation result obtained with sparse pointcloud would be inaccurate when interpolating back onto the original dense

pointcloud. To address this issue, we convert the large-scale pointclouds into semantically similar point clusters. By doing this, the amount of GPU memory demanded by the semantic segmentation network is drastically reduced. When tested on the benchmarked dataset Stanford Large-Scale 3D Indoor Spaces Dataset, we observe that MuGNet can inference at once up to 45 pointclouds, each containing on average 2.6 million points.

Besides the challenge posed by computation requirements, semantic segmentation is also challenged by the orderless and uneven nature of the raw pointclouds. This leads to the less desirable performance of deep convolutional neural networks, which require evenly arranged and ordered data. Prior research has attempted to convert the orderless pointclouds to mesh or voxels as ways to artificially structure pointclouds before applying deep convolutional neural networks [3, 4, 5, 13, 14]. Such attempts typically result in voluminous rendered data and introduce unnecessary computation overhead to the algorithm. To overcome this challenge, we propose a framework with graph convolutions invariant to permutations of pointcloud orderings. Segmentation can thus be performed without artificially structuring the pointcloud.

In addition to addressing the two major challenges, our proposed framework also features a bidirectional multi-resolution fusion network. The framework reasons the relationship between adjacent clusters at different resolutions with both forward and backward paths. We observe that the concatenated graph features from different resolutions provide richer feature representation than the resultant features from the final convolution layer alone. The backward fusion network then further enriches the representations. With the aforementioned design components, we demonstrate that our proposed MuGNet achieves 88.5% overall accuracy and 69.8% mIOU accuracy on Stanford Large-Scale 3D Indoor Spaces Dataset semantic segmentation, outperforming SPG [15] by 7.7% better mIOU accuracy and 3% better overall accuracy. Figure 1 presents a sample semantic segmentation result for Area 3 in the Stanford Large-Scale 3D Indoor Spaces Dataset.

2 Related Work

Three main categories of learning-based frameworks have been previously proposed for pointcloud semantic segmentation: voxel-based approach, point-based approach, and graph-based approach. We outline the corresponding approaches as follows.

Voxel-based approach: As attempts to tackle the orderless and uneven nature of pointcloud, previous algorithms have ventured to convert pointcloud into structured voxel data such that convolution neural networks can be applied. Volumetric CNN [3] for example, pioneered on voxelized shapes; FPNN [4] and Vote3Deep [5] proposed special methods to deal with the sparsity problem in volumes. Voxelizing a large pointcloud scan, however, imposes computation overhead. Such operation becomes infeasible for processing dense large-scale pointclouds.

Point-based approach: PointNet [6] has inspired many previous works to process pointclouds as direct input. Typical point-based frameworks learn feature encodings of each point. The encoded features are then aggregated with a permutation invariant function to arrive at transformation invariant segmentation results. Spatial and spectral convolutions have been implemented in previous works [8, 16, 9, 13, 2, 11, 10] to improve the efficiency of feature encoding. All of these approaches are demanding in memory usage and thus require either a sliding window approach or data downsampling approach to process dense pointclouds. In contrast, our framework alleviates the memory demand during the segmentation process by preforming point clusters based on geometric similarities. In this way, we can process multiple dense pointclouds at once with a commonly available GPU.

Graph-based approach: While pointclouds are inherently orderless and unevenly distributed, they can be represented as graphs with interdependency between points. Node classification can be performed to distinguish the classes among the nodes in the graph representations [17, 18, 19]. Wang et al.[20] first applied the concept of graph convolutional network on pointcloud data and formulated an approach that dynamically computes the graphs at each layer of the network. Various types of graph convolutional neural networks have since been utilized in the context of pointcloud segmentation [21, 22, 15, 23, 24]. All of the frameworks sequentially infer the graph embeddings with only forward network paths. On the contrary, we exploit a bidirectional framework that retains rich contextual information derived from multiple convolution units. Our network's backward path receives graph embeddings at different resolutions and infer rich contextual relationships to achieve high segmentation accuracy.

3 Proposed Computational Framework

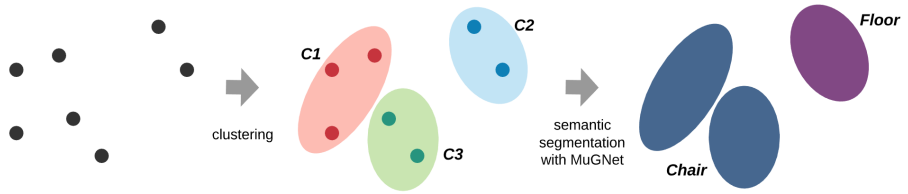


Figure 2: Illustration of the overall workflow for MuGNet. The input pointcloud is first clustered, and then each formed cluster is classified into its respective semantic classes based on cluster-features.

Given a large-scale pointcloud with millions of points spanning up to hundreds of meters, directly processing it with a deep neural network requires an innumerable amount of computation capability. Downsampling points from the original pointcloud by order of magnitude is common in coping with this limitation. However, this approach takes away the intricate details in the original pointcloud and yet still suffers from expensive memory usage. We propose MuGNet, a multi-resolution graph neural network inspired by EfficientDet [25], to effectively translate large-scale pointclouds into directed connectivity graphs efficiently segments the pointclouds from the formed graph with a bidirectional graph convolution network. MuGNet allows the processing of multiple large-scale pointclouds in a single pass while producing excellent segmentation accuracy. Figure 2 showcases the overall workflow of MuGNet. In the subsequent sections, we will further explain the cluster formation process and introduce the three key design features in our segmentation network: cluster feature embedding, feature-fusion backbone, and bidirectional-convolution network.

3.1 Clustering algorithm for graph formation

We preprocess large-scale pointclouds into geometrically homogeneous clusters, a 3D equivalent of superpixels commonly used in image analysis. The existing point clustering approaches can be roughly classified into unsupervised geometric approaches and supervised learning-based approaches. There is currently no standard clustering strategy that is suitable for large-scale pointclouds. We individually analyze their relative merits as follows and indicate our reasoning for choosing the supervised clustering approach.

(1) Unsupervised clustering Given a pointcloud data with millions of points, geometric features based on the neighborhood planarity, linearity, and scattering can be calculated [26, 15]. The unsupervised methods rely on the assumption that geometrically or radio-metrically homogeneous segments are also semantically homogeneous. This assumption is challenged when attempting to form semantically homogeneous clusters for semantic segmentation [27]. Since the clusters are formed without semantic information oversight, the clusters can have semantic impurity among its constituting points. The impurity within each of the clusters negatively impacts the subsequent node classification network's performance and limits the final semantic-segmentation accuracy.

(2) Supervised clustering The supervised-clustering method proposed by Landrieu et al.[27] standardizes a pointcloud with a spatial transformer, embeds local feature with a small MLP-based network, and finally forms clusters by optimizing on the generalized minimal partition problem with the introduction of contrastive loss as a feedback metric for cluster purity. The supervised clustering approach achieves significant improvements compared to the unsupervised methods in terms of reducing semantic impurity within the formed clusters. The semantically pure clusters can provide enhanced geometric and semantic connectivity information for subsequent learning tasks. On average, the supervised clustering technique can effectively encapsulate pointcloud information of a Stanford Large-Scale 3D Indoor Spaces Dataset room scan containing ~ 2.6 million points into $\sim 10^3$ clusters. Compared to downsampling a pointcloud scan randomly, the conversion to point clusters can perform a drastic reduction in data size while carrying richer contextual information from the original pointcloud. Figure 5(c) and 6(c) visualize sample pointcloud scans color-coded by geometric features and point clusters produced by the supervised clustering approach.

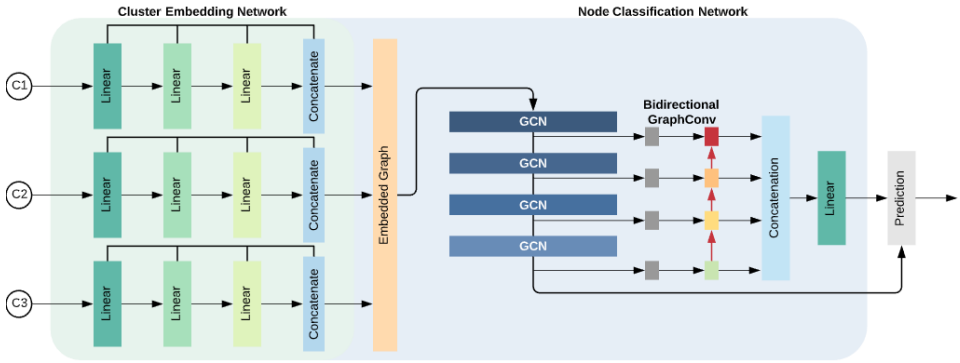


Figure 3: MuGNet includes cluster embedding and bidirectional graph convolution networks. The input pointcloud is clustered based on their geometric similarities and learnable features into cluster set $C = \{C_1, C_2, ...C_K\}$, where K denotes the total number of clusters formed for a pointcloud.

3.2 Cluster-feature Embedding

As shown in Figure 3, our cluster-feature embedding network is applied to each of the formed pointcloud clusters. The points in a cluster go through a multi-resolution feature-fusion network with three sets of multi-layer perceptrons. The input points contain geometric features analogous to [15]. The feature-fusion network receives pointcloud at three different resolutions. The highest resolution channel is defined as the cluster points, from which the other two lower-resolution channels are formed by down-sampling with linear layers. Each of the three-channel inputs then goes through a series of 2D convolutions, and their output features are concatenated together into a single feature vector. Utilization of the feature-fusion network for cluster-feature extraction effectively identifies cluster-features that are not only local representative but also distinctive at a larger receptive field.

3.3 Feature-fusion Backbone

The node-classification network that identifies the semantic class for each formed cluster employs a backbone network to extract feature vectors at different resolutions. The feature vectors are subsequently fed to a bidirectional pyramidal feature-fusion network. Consider a graph $G(V, E)$ containing point clusters as nodes and node features obtained from the embedding network, where $V = 1, 2, ...N$ and $E \subseteq |V| \times |V|$ represent the set of vertices and edges in the formed graph respectively. N denotes the total number of nodes. The set of neighboring nodes of Node i can be denoted as $N(i) = \{j : (i, j) \in E\} \cup \{i\}$. Each node feature $h_i \in \mathbb{R}^F$ is associated with a corresponding graph Node $i \in V$, among the set of node features, $H = \{h_1, h_2, ..., h_N\}$, where F denotes feature dimension at each node.

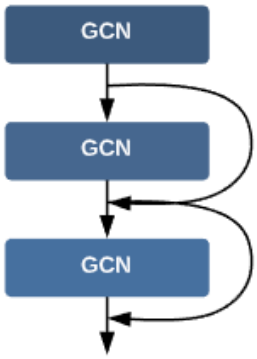


Figure 4: Exploded-view of the backbone layers.

The backbone network consists of four residual blocks, each constructed with a graph convolution, batch normalization, and activation layer:

$$H_i^{bbout} = Activation(BatchNorm(GraphConv(H_i^{bbin}))),$$

where i denotes the level of backbone. It has been observed in previous works that with the message-passing mechanism for graph convolutions, the node features eventually become similar as the number of passing increases. Short-term and long-term residual connections are added to the basic building blocks to increase information density and avoid gradient diminishing as the network depth increases. For the short-term residual connection, the output features from the previous block are aggregated with a simple addition operation. The expression of the residual block with a short-term direct connection from the previous block is formulated as:

$$H_2^{bbout} = \{Activation(BatchNorm(GraphConv(H_2^{bbin}))) + H_1^{bbout}\}.$$

The outputs from the residual blocks are fed into the bidirectional graph convolution network in parallel. At the end of the backbone structure, the features are concatenated together for cluster

classification:

$$H^{bbout} = \{H_1^{bbout}, H_2^{bbout}, ..., H_4^{bbout}\},$$

where H^{bbout} expresses the concatenated final feature output from the backbone network.

3.4 Bidirectioal-graph Convolution Network

Multi-scale feature-fusion aims to aggregate features at different resolutions. Given a list of multi-scale features, $H^{in} = (H_{l1}^{in}, H_{l2}^{in}, ...)$, where each of H_{li}^{in} represents the feature at level l_i, the feature-fusion network acts as a transformation function, f, that aggregates features at different resolution and outputs a list of new features denoted as $H^{out} = f(H^{in})$. The structure of the bidirectional graph convolution network is shown in Figure 3. The network receives graph features, $H^{in} = (H_1^{in}...H_4^{in})$, at different resolutions from backbone levels 1-4. The conventional feature pyramidal network used in image tasks aggregates multi-scale features in a top-down manner and performs resizing operation for resolution matching. Each of the network nodes in the pyramidal network is formulated in a similar fashion to the basic backbone building block, where batchnorm and activation layers are incorporated on top of each of the graph convolution:

$$H_i^{out} = Activation(BatchNorm(GraphConv(H_i^{in}))).$$

For graph convolutions, resizing becomes inefficient and causes information mixing during the message passing operation. The feature pyramidal network is thus redesigned to produce a constant number of features at each level:

$$\begin{aligned} H_4^{out} &= GraphConv(H_4^{in}), \\ H_3^{out} &= GraphConv(H_3^{in} + H_4^{out}), \\ &\cdots \\ H_1^{out} &= GraphConv(H_1^{in} + H_2^{out}). \end{aligned}$$

While propagating the node features through a pyramidal network complements the baseline backbone structure by enriching the aggregated feature information, it is still limited by the one-way information flow. To address this issue, we configure a bidirectional graph propagation network that creates a two-way information flow to aggregate information for both the deep-to-shallow direction and the other way around. A skip connection that travels from the input node to the output node is incorporated for each resolution in addition to the bidirectional passes. In this way, feature information is more densely fused, and the diminishing-gradient issue can be avoided to an extent.

Resultant features from the fusion network for different resolutions contribute to the final output feature unequally, and naively concatenating the features requires unnecessarily heavy memory usage. Instead, an additional set of weights are introduced when aggregating the multi-res output features into a single feature vector. The intuition is to have the network itself learn the importance of each resolution during training. This approach avoids manual assignment of importance based on potentially biased or inaccurate human knowledge and leads to a more elegant memory usage compared to naive concatenation. To give a sample formulation of the final graph bidirectional GraphConv network with learnable weights for different resolutions, the formulation for aggregation at level 3 is shown as follows:

$$\begin{aligned} H_3^{mid} &= GraphConv(\frac{w_1 \cdot H_3^{in} + w_2 \cdot H_4^{in}}{w_1 + w_2 + \epsilon}), \\ H_3^{out} &= GraphConv(\frac{w_1' \cdot H_3^{in} + w_2' \cdot H_3^{mid} + w_3' \cdot H_2^{out}}{w_1' + w_2' + w_3' + \epsilon}), \end{aligned}$$

where H_3^{in} indicates the middle feature-passing node at the third level on the top-to-bottom pathway. Features propagated from channels for the other resolutions are constructed similarly. The pathway and network edges are formulated according to connections shown in Figure 3.

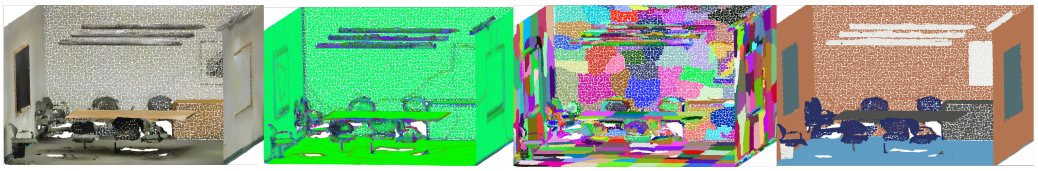
Through experiments, we have found that graph neural network is prone to experiencing information assimilation, which leads to ineffective feature-fusion at deeper convolution levels. This phenomenon is conceptually analogous to the thermodynamic equilibrium state where the temperature gradient between two objects diminishes as the energy from one object is transferred to another. We send the output features from both the backbone network and the pyramidal fusion network to the segmentation module to address this problem.

4 Results

In this section, we evaluate the MuGNet on various 3D pointcloud segmentation benchmarks, including the Stanford Large-Scale 3D Indoor Spaces(S3DIS) dataset [1], and the Virtual KITTI(vKITTI) dataset [28]. The network performance is evaluated by the mean Intersection-over-Union(mIOU) and Overall Accuracy (OA) of all object classes specific to each dataset.

4.1 Segmentation Results on S3DIS

The Stanford Large-Scale 3D Indoor Spaces Dataset provides a comprehensive clean collection of indoor pointcloud scans. There are in total over 695 million points and indoor scans of 270 rooms [1]. Each scan is a dense pointcloud of a medium-sized room ($\sim 20 \times 15 \times 5$ meters). We use the standard 6-fold cross-validation approach in our experiments.



(a) Raw input (b) Geometric features (c) Pointcloud clusters (d) MuGNet segmentation result

Figure 5: Qualitative semantic segmentation results of MuGNet on the validation set of S3DIS [1].

Table 1 shows the network's performance averaged over all six areas. Our network has achieved better performance than state-of-the-art graph-based methods. Compared to the latest state-of-the-art for non-graph-based network RandLA-Net [29], our network also achieves comparable results with 0.5% higher overall accuracy and only 0.2% lower mIOU. It should be noted that our network has been validated to consistently arrive at the reported result for the five times that the network is trained. In contrast, some of the baselines tend to produce inconsistent results due to their random sampling operations.

	OA	mIOU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PointNet [6]	78.5	47.6	88	88.7	69.3	42.4	23.1	47.5	51.6	42	54.1	38.2	9.6	29.4	35.2
Engelmann et al. [23]	81.1	49.7	90.3	92.1	67.9	44.7	24.2	52.3	51.2	47.4	58.1	39	6.9	30.0	41.9
DGCNN [24]	84.1	56.1	-	-	-	-	-	-	-	-	-	-	-	-	-
SPG [15]	85.5	62.1	89.9	95.1	76.4	62.8	47.1	55.3	68.4	73.5	69.2	63.2	45.9	8.7	52.9
RandLA-Net [29]	88.0	**70.0**	-	-	-	-	-	-	-	-	-	-	-	-	-
MuGNet (ours)	**88.5**	69.8	**92.0**	**95.7**	**82.5**	**64.4**	**60.1**	**60.7**	**69.7**	**82.6**	**70.3**	**64.4**	**52.1**	**52.8**	**60.6**

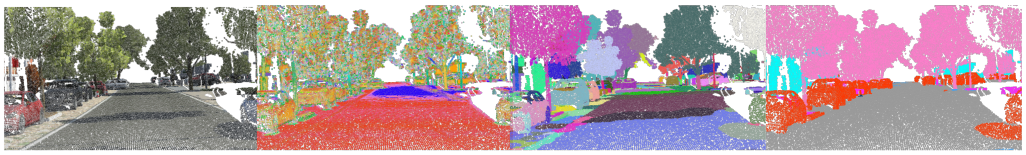
Table 1: Semantic segmentation results measured by overall accuracy, mean intersection over union, and intersection over union of each class for all six areas in S3DIS dataset [1].

	OA	mIOU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PointNet [6]	-	41.1	88.8	97.3	69.8	0.05	3.9	46.3	10.8	52.6	58.9	40.3	5.8	26.4	33.2
Engelmann et al. [23]	-	48.9	90.1	96.0	69.9	0.0	18.4	38.3	23.1	75.9	70.4	58.4	40.9	13.0	41.6
SPG [15]	86.4	58.0	89.3	96.9	78.1	0.0	42.8	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
GACNet [22]	87.8	62.8	**92.3**	**98.3**	81.9	0.0	20.3	**59.1**	40.8	78.5	**85.8**	61.7	**70.7**	**74.7**	52.8
MuGNet (ours)	**88.1**	**63.5**	91.0	96.9	**83.2**	**5.0**	**37.0**	54.3	**62.6**	**85.3**	76.4	**70.1**	55.2	55.2	**53.4**

Table 2: Semantic segmentation results measured by overall accuracy, mean intersection over union, and intersection over union of each class in Area 5 of S3DIS dataset [1].

We would also like to investigate our network's performance compared to GACNet [22], a state-of-the-art graph convolution for pointcloud semantic segmentation. The results for Area 5 are compared since this is the only area reported for the work. As shown in Table 2, our network still achieves better performance than the rest of the baselines for the segmentation task on the Area 5. Besides, most of the compared baseline networks struggle with handling a single pointcloud scan at once. They require drastic down-sampling and dividing of a large pointcloud room into small $1 \times 1 \times 1$ blocks with sparse points. Our network, on the other hand, can effectively process multiple rooms in a single pass.

4.2 Segmentation Results on Virtual KITTI



(a) Raw input (b) Geometric features (c) Pointcloud clusters (d) MuGNet segmentation result

Figure 6: Qualitative semantic segmentation results of MuGNet on the validation set of vKITTI [28].

The Virtual KITTI (vKITTI) dataset [28] contains simulated LiDAR data acquired through 50 annotated 1242×375 resolution monocular videos generated from five different worlds in urban setting. Each of the scans in the dataset is a dense pointcloud with 13 semantic classes. The annotated 2D depth images are projected into 3D space to produce the simulated LiDAR scans that resemble pointclouds obtained by real-life LiDAR scanners. For testing and training, the scans are separated into 6 non-overlapping sets. We obtain the final evaluation following the 6-fold validation protocol.

	OA	mIOU	terrain	tree	vegetation	building	road	g-rail*	t-sign*	t-light*	pole	misc	truck	car	van
PointNet [6]	63.3	17.9	32.9	76.4	11.9	11.7	49.9	3.6	2.8	3.7	3.5	0.7	1.5	25.1	3.4
Engelmann et al. [23]	73.2	26.4	38.9	87.1	14.6	44.0	58.4	12.4	9.4	10.6	5.3	2.2	3.6	43.0	13.3
MuGNet (ours)	**85.1**	**50.0**	**70.0**	**88.6**	**35.2**	**63.0**	**80.2**	**40.8**	**32.0**	**56.3**	**23.4**	**3.92**	**7.1**	**84.3**	**65.4**

Table 3: Overall accuracy, mean intersection over union, and intersection over union of each class for all six splits in vKITTI dataset[28]. *"t-" is short for traffic; "g-" is short for guard.

Table 3 shows a quantitative comparison of MuGNet against other benchmark methods for the dataset, and Figure 6 presents MuGNet's qualitative segmentation result. The reported model is trained with the XYZ coordinates without RGB values to investigate our model's ability to learn geometric features. For comparison purposes, the benchmarked models are also trained with the same configuration.

MuGNet has demonstrated to exceeded previous approaches in all evaluation metrics. It should be noted that during inference, MuGNet can fit all 15 scans in each of the six splits into a single GPU simultaneously, making it more viable for inspection tasks where computation resource is limited.

4.3 Efficiency Analysis

The scalability of MuGNet is investigated by increasing the number of rooms to be processed in a single shot with the Stanford Large-Scale 3D Indoor Spaces(S3DIS) dataset [1]. Table 4 showcases the number of rooms for inference against the respective inference time and GPU memory consumption, and Figure 7 presents the corresponding plot of the trend. With a setup of single NVIDIA RTX 1080Ti GPU, up to 45 rooms (totaled $38,726,591$ points) can be accommodated into the available computation resource. We have observed that with an increase of room number, the increase in memory usage slows down and resembles a pseudo-logistic growth. The inference time also only increases at a relatively low rate. The growth trends indicate that the model scales nicely with an increased number of scans to be processed. Therefore, the model would be desirable for the application scenario where a multitude of dense scans needs to be processed with a few shots.

# of Rooms	Mean Inference Time(sec)	GPU Memory Usage(GB)
1	0.5056	1.042
5	0.5396	4.073
12	0.6124	6.684
34	0.7077	10.365
45	0.8332	10.617

Table 4: Efficiency analysis on S3DIS dataset [1].

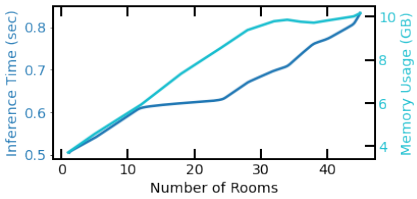


Figure 7: Corresponding efficiency plot.

4.4 Ablation Study

To further investigate the behavior of our network, we conduct the following ablation studies. All ablated networks are trained and tested with the same 6-fold validation setup, as previously mentioned in Section 4.1.

(1∼3) Removing Bidirectional GraphConv: We want to study the backbone's ability to propagate useful information. In conventional convolutions, it has been widely proven that deeper networks often produce features at a higher quality that are more representative of the entire data distribution and enhance network performance. If this phenomenon remains true for graph convolutions, then using features from deeper layers of the backbone as input to Bidirectional GraphConv should, in theory, yield a higher accuracy segmentation result. After removing the bidirectional network entirely, we obtain the final features directly from the backbone network before feeding them to the segmentation network. We also vary the number of backbone blocks to $7, 14$, and 28 to investigate the extent of the backbone structure's capability. From the three different depths that the backbone has been tested, it can be observed that increased backbone depth, in fact, leads to better performance.

(4) Stacking multiple Bidirectional GraphConvs: The Bidirectional GraphConv network is structurally capable of stacking multiple network copies in parallel. Previous works in 2D image processing have shown a positive correlation between model performance and the number of stacked networks. However, graph convolutions are fundamentally different from the conventional convolutions for 2D images, where a message-passing mechanism replaces parts of filtering operation. Naively adopting the structure used in 2D image processing would lead to sub-optimal results.

(5) Increase depth of backbone layers: Previous studies on graph neural networks have alerted on the loss of expressive power as the network complexity increases [30, 31]. It is, therefore, crucial to investigate the impact of backbone depth on the Bidirectional GraphConv network. When the backbone layer is increased to 14, and the last 4-layer outputs are fed into the Bidirectional GraphConv, the resultant segmentation is deemed to be less optimal than having 4 layers of the backbone. Besides, the GPU memory usage increases by $\sim 1/3$ times from our final network. Despite incorporating short-term and long-term residual connections, graph convolutions are still prone to information assimilation among nodes. A deeper backbone network does not necessarily extract higher-quality features.

	mIOU%
(1) Backbone with 7 layers	59.0
(2) Backbone with 14 layers	53.4
(3) Backbone with 28 layers	64.5
(4) Stacking 2 bidirectional GraphConvs	66.3
(5) MuGNet with 14-layer backbone	63.7
(6) MuGNet Baseline	**69.8**

Table 5: The 6-fold validated mean IOU of ablated networks tested on the S3DIS dataset [1]. The results are compared against the final MuGNet configuration.

5 Conclusion

In this paper, we have demonstrated that it is possible to process a large quantity of dense pointcloud scans by reformulating the learning objective to utilize a bidirectional graph convolutional network. Most of the current approaches rely on drastic downsampling or sliding window operations, both of which negatively impact the segmentation result on dense pointclouds. In contrast, we effectively transform the pointclouds into graph representations, which radically reduce the computation demand and facilitate the processing of up to forty-five room scans with a single commercially available GPU. A multi-resolution cluster embedding network is introduced to provide high-quality representations of the point clusters. Finally, a bidirectional-graph convolution network aggregates useful features from different graph resolutions. Extensive experiments on benchmark datasets have proven our network's strong capability in processing a multitude of pointclouds. It has also achieved state-of-the-art accuracy for pointcloud semantic segmentation. Our model would be desirable for application scenarios where many pointcloud scans need to be processed at once, and detailed pointcloud information needs to be retained in the segmented pointcloud. Our algorithm is limited by the observed data distribution as other data-driven approaches. This shortcoming can be potentially solved by introducing a high-level probabilistic framework. Future work will involve investigations on potential industrial applications and robust inference for the model.

Acknowledgments

The authors like to thank YKK AP Inc. for their financial support for this work.

References

[1] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, Feb. 2017.

[2] H. Thomas, C. R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *CoRR*, abs/1904.08889, 2019. URL http://arxiv.org/abs/1904.08889.

[3] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016.

[4] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas. FPNN: field probing neural networks for 3d data. *CoRR*, abs/1605.06240, 2016. URL http://arxiv.org/abs/1605.06240.

[5] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. *CoRR*, abs/1609.06666, 2016. URL http://arxiv.org/abs/1609.06666.

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. URL http://arxiv.org/abs/1612.00593.

[7] Z. Zhang, B.-S. Hua, and S.-K. Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[8] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst. Shapenet: Convolutional neural networks on non-euclidean manifolds. *CoRR*, abs/1501.06297, 2015. URL http://arxiv.org/abs/1501.06297.

[9] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV 2018*, pages 90–105, Cham, 2018. Springer International Publishing.

[10] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. URL http://arxiv.org/abs/1706.02413.

[11] Y. Li, R. Bu, M. Sun, and B. Chen. Pointcnn. *CoRR*, abs/1801.07791, 2018. URL http://arxiv.org/abs/1801.07791.

[12] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] A. Komarichev, Z. Zhong, and J. Hua. A-CNN: annularly convolutional neural networks on point clouds. *CoRR*, abs/1904.08017, 2019. URL http://arxiv.org/abs/1904.08017.

[14] J. Mao, X. Wang, and H. Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[15] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. *CoRR*, abs/1711.09869, 2017. URL http://arxiv.org/abs/1711.09869.

[16] S. Tang, B. Li, and H. Yu. Chebnet: Efficient and stable constructions of deep neural networks with rectified power units using chebyshev approximations. *ArXiv*, abs/1911.05467, 2019.

[17] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

[18] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017. URL http://arxiv.org/abs/1706.02216.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[20] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018. URL http://arxiv.org/abs/1801.07829.

[21] Y. Shen, C. Feng, Y. Yang, and D. Tian. Neighbors do help: Deeply exploiting local structures of point clouds. *CoRR*, abs/1712.06760, 2017. URL http://arxiv.org/abs/1712.06760.

[22] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan. Graph attention convolution for point cloud semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[23] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe. Exploring spatial context for 3d semantic segmentation of point clouds. *CoRR*, abs/1802.01500, 2018. URL http://arxiv.org/abs/1802.01500.

[24] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018. URL http://arxiv.org/abs/1801.07829.

[25] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection, 2019.

[26] J. Demantké, C. Mallet, N. David, and B. Vallet. Dimensionality based scale selection in 3d lidar point clouds. *Proceedings of the ISPRS Workshop Laser Scanning*, 38:97–102, 01 2011. doi:10.5194/isprsarchives-XXXVIII-5-W12-97-2011.

[27] L. Landrieu and M. Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. *CoRR*, abs/1904.02113, 2019. URL http://arxiv.org/abs/1904.02113.

[28] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. *CoRR*, abs/1605.06457, 2016. URL http://arxiv.org/abs/1605.06457.

[29] 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. *arXiv preprint arXiv:1911.11236*, 2019.

[30] K. Oono and T. Suzuki. On asymptotic behaviors of graph cnns from dynamical systems perspective. *CoRR*, abs/1905.10947, 2019. URL http://arxiv.org/abs/1905.10947.

[31] N. Dehmamy, A. Barabási, and R. Yu. Understanding the representation power of graph neural networks in learning graph topology. *CoRR*, abs/1907.05008, 2019. URL http://arxiv.org/abs/1907.05008.