Graph Neural Networks for Computer Vision: Semantic Segmentation

Sana Moin
CV Project Seminar Report 2022

I. Introduction

With the advent of deep neural networks, many complex tasks such as machine translation, object recognition, semantic segmentation, etc. are now becoming a reality. Moreover, the rapid development of technologies and computational resources has enabled faster processing and made the development more feasible to approach by a mass of people, therefore opening up a myriad of paradigms to work with. Graphs are one of the data structures that have been used extensively to solve a variety of computational tasks as they can store the relationships between different elements efficiently. As a lot of data can be represented in graphs, deep learning has also extended its roots to use the strengths of this structure to create even better representations of the data which can provide more meaning to it. Nowadays, graph neural networks(GNN) is used in all modalities of data, such as for scene graph generation, point clouds classification and segmentation, action recognition, text classification, recommender systems, chemistry, and the list goes on [28]. Some common works are done in the direction of object detection [21] and for image and video understanding [17]

The task of semantic segmentation in computer vision is to group the pixels in an image together which belong to the same class. The image data can be represented in a 2D or a 3D form. Several methods are used to segment 2D images using a variety of deep learning techniques. One of the earliest models for this task was using Fully Convolutional Networks [12] which uses a modified version of existing convolutional neural network(CNN) architectures, another model uses CNNs along with fully connected Conditional Random Fields [7]. In [16], transposed convolution was used to achieve this task. Furthermore, many models were developed based on faster R-CNN [20] to perform instance segmentation on 2D images [5]. Recurrent Neural Network(RNN) based architectures are also leveraged by different models, like ReSeg model[25] where CNN features are used by RNNs, another model that is based on Long Short Term Memory(LSTM) on natural scene images [1], which was then further extended to be used on graphs as well [11]. Several attention mechanisms were also used in various literature such as by using multi-scale features of images [6] or using Pyramid attention network where dense features are extracted from spatial pyramid [9]. Adversarial training approaches work well as well where a convolutional semantic network is used for adversarial training [14].

Even though these varieties of networks perform well on 2D images, there are drawbacks to using the 2D images for semantic segmentation itself. 2D images do not contain the exact geometric or shape information, occlusions, illumination patterns, and posture can distort the representations, and little can be understood about the scale of what is visible. Therefore there has been a lot of research done on how 3D data can be represented and used effectively for the task of semantic segmentation.

There are many ways to represent 3D data. One of them is multiview, where for the same images multiple 2D images are taken by changing their view to get a general understanding of how it would appear overall. CNNs can be applied to such images later [22], however generating multiple images of the same view is an expensive approach. Another way to capture a 3D pose is by using a depth map and then applying CNNs over it [27], but it is prone to noisy data. Another common approach is to use Voxels, which is a 3D grid representation of the objects in the scene that can be used for volumetric shape modeling or object detection, like in VoxNet which uses 3D CNN [15]. They are quite efficient in modeling 3D poses of objects, however, there are a few drawbacks in using them because they miss the fine features of the objects, and integration of color information is difficult, which could be needed for many tasks. Then comes Point Cloud, which is simply a cluster of unordered points in a 3D space that can represent the objects visible. They are represented by coordinates in the space, can use the color data, and is what it is like in real-life scenarios, but they do not include the information about surfaces.

Because point clouds give a good geometric representation of the surroundings, they can be leveraged to do the task of semantic segmentation as well, although the aspect of them being unordered and sparse does pose a challenge. A range of approaches are used to perform this task, which involves

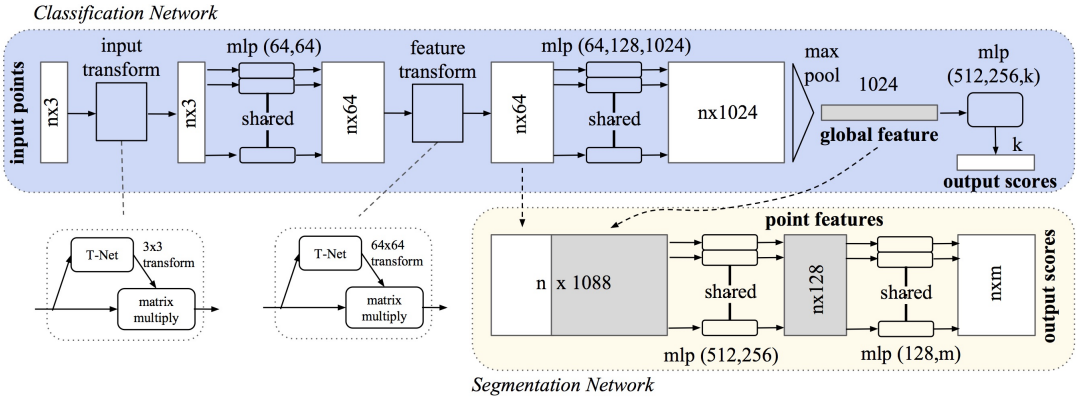

Figure 1. PointNet Architecture [18]

using different architectures of CNNs and transformation networks, however, the focus of this survey will primarily be on the use of GNNs on PointCloud data for semantic segmentation.

In the next sections, the background will define what graph neural networks are and how the semantic segmentation task is approached on point cloud data, furthermore, the next section will go deep into the architectures of a few approaches for the task, followed by a result section where the models will be compared on the accuracy of the task, and finally discussion and conclusion section where pros and cons of the model will be compared, along with future work directions.

II. Background

A. Point Clouds

Point Clouds are a set of N unordered points each having a dimension of D, that simply contains the xyz coordinate information, but can also include the RGB information, laser intensity information, etc. Point clouds can be captured by sensors such as LiDaR, photogrammetry, RGB-D images, etc. Several architectures can process point cloud data for a variety of tasks such as object classification, part segmentation, scene semantic parsing, modeling, scientific computing, etc.

Because of the characteristics of point clouds, raw point cloud data wasn't processed in the earlier works. Point clouds were converted into some other form of representation, such as voxels, and then processed by any neural network. There are a few conditions that neural networks need to fulfill to be able to process the raw point data. As the point cloud is an unordered set and a permutation of the order of the set will not change the set representation, the neural networks that can process them should be permutation invariant. For that, the neural networks should learn symmetric functions to learn such a representation, where all the points can be independently and identically modified. and then the embeddings can be mapped to a higher-dimensional space, and this representation could be then aggregated and processed by the neural networks. Another aspect to be considered is the invariance under geometric transformations, which means that any rotation in the object shouldn't alter the class results.

PointNet [18] was the earliest model to process raw point cloud points without rendering to perform deep learning tasks. There are several works done based on this approach to achieve several different computer vision tasks. It tackles the above problems and gives a base to begin further explorations with point cloud raw data. The architecture of the model is shown in Fig. 1. For semantic segmentation, the input is the n points on which input and feature transform are applied using a spatial transformation network, which is a T-net that is also trained end-to-end with the rest of the network. To take care of the invariance to the geometric transformation, there is a matrix multiplication applied which transforms the alignment points in the embedding space. The networks approximate some continuous symmetric function to handle permutation invariance. Max pooling is applied to these features to obtain a global feature. Max pooling was reportedly better than average pooling or weighted average pooling for this task. The global features then can perform object classification. For performing segmentation, the global features are concatenated with the local features of the points and processed through another network to get a classification for each pixel, which would then eventually represent the m semantic subcategories. A drawback of this model is that it cannot learn the local structure of the objects, hence fails to recognize fine-grained patterns and its generalizability to complex scenes.

A follow up model called PointNet++ [19] enhances the learning representation of Point Cloud by capturing local structure through adaptively combining features from multiple scales.

This network is the base and inspiration for many more follow-up works that were done for processing point cloud for semantic segmentation tasks.

B. Semantic Segmentation

The task of semantic segmentation can be further classified into three categories [8]:

1) Region based semantic segmentation: In this, free-form regions are extracted, described, and classified, and then the pixel is labeled according to the highest-scoring region it belongs in.
2) FCN-based semantic segmentation: They do not extract the regions and learn the mapping from pixels to pixels.
3) Weakly supervised semantic segmentation: To overcome the overhead of annotating segmentation masks, different approaches are used to annotate and use the images for semantic segmentation task , such as by using bounding boxes.

For this paper, we will focus on the second approach. As mentioned in the previous section, many networks were processing point clouds by converting them into some other form. This is a type of indirect point cloud segmentation technique [30], which involves using multi-view methods that use different poses of point cloud to get multi-view images and get the features, such as in MVCNN [23] or volumetric methods which deal with regular volumetric occupancy grid such as in VoxNet. When it comes to direct point cloud segmentation techniques, they can be done based on point ordering, like in PointCNN [10] where X transform is used to re-weight and arrange the associated features of each point. Another way is by using multi-scale, as used in PointNet++ [19] that recursively applies PointNet on a nested partitioning of the input point set, and features are adaptively combined from multiple scales to recognize fine-grained patterns. The third way is through feature fusion where PointNet lies, and the fourth one is by fusing Graph Convolutional Neural Network(GCNN). We will be exploring the fourth technique in this paper.

C. Graph Neural Networks

A graph G is a data structure that is represented by a set of edges E and nodes or vertices V. Graphs can keep expanding in their dimension, are unordered and the nodes might depend upon each other. It can be either a directed graph, where the edges that connect the nodes have a direction, or an undirected graph. Graphs can be shown in an adjacency matrix, where each row is a node vector and the rows depict the connections of the nodes with each other. The nodes can depict a variety of object types, it could be information about a person on a social network, a vehicle on the road, a part of a chemical, etc. Graphs can be constructed by different approaches. It can be either a complete graph where every node is connected to every other node by an edge, or by using k nearest neighbor nodes, or by logically connecting the nodes with some predefined function or dependency structure. GNNs are used to infer relevant knowledge from the nodes, edges or the complete graph as a whole. These networks are supposed to be invariant to ordering and should be able to process data of arbitrary size. The first step to begin using GNNs is to encode the information into the graph. Then GNN can be used on this encoded information. There are different ways to structure the graph neural network. On a very high level, the embeddings are learned step by step. First each node can be updated with the information from the neighboring nodes by what is called message passing, which happens for every node in a single iteration, followed by aggregation of the obtained local features, and this process repeats. Similarly, edges can also store feature information. A general way to represent message passing in GNN can be done as follows:

$$f(x_i) = \phi(x_i, g, \psi(x_i, x_j)) \tag{1}$$

Here, f is the message passing function, x denotes the node values, i denotes the current node value, j denotes all the other neighboring node values, g denotes an aggregation function that should be permutation invariant, ψ denotes the learnable weights of the network and ϕ denotes the general learnable function.

Different architectures are used to work along with graph data, such as Recurrent Graph Neural Networks, Convolutional Graph Neural Networks, Graph Autoencoders and Spatial-Temporal Graph Neural Networks [29]. There are several models developed to process both 2D and 3D image data. For example, to process 2D image data, in Graph-FCN model [13], image grid data is converted into graph structure by CNN on which graph convolutional network is used to perform the semantic segmentation task via graph node classification problem. In this paper, we will focus our attention on 3D data, specifically Point Cloud data to perform semantic segmentation using some form of GNN.

III. GNNs for Semantic Segmentation of Point Clouds

This section will describe three architectures, Dynamic Graph CNN (DGCNN) [26], Graph Attention-based Point Neural Network (GAPNet) [4] and Regularized Graph CNN (RGCNN) [24], that perform semantic segmentation on Point Cloud using GNNs.

A. DGCNN

DGCNN is inspired by both PointNet architecture and convolution operation. PointNet assumed the independence of the points to maintain permutation invariance, therefore not considering the geometric relationships among points. In DGCNN, they address this issue using EdgeConv. As the name suggests, EdgeConv comprehends the relationship of any point with the other neighborhood points by generating edge features by applying convolution-like operations on edges and thus helps in understanding the semantics of the

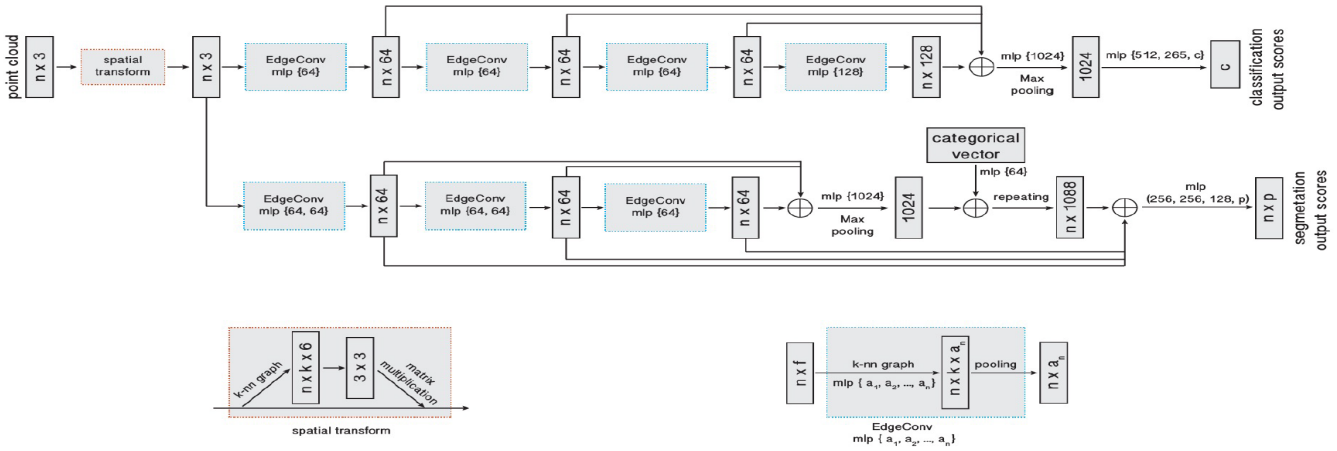

Figure 2. DGCNN Architecture [26]

graph. It has properties lying between translation-invariance and non-locality [26]. The other aspect is the creation of a graph for processing. The spatial transform part of PointNet is also modified to get the canonical pose. As seen in the Fig. 2 bottom left, the spatial transform, or the input transform, takes the point cloud as input and converts it into a canonical pose using the coordinate difference of points with k nearest neighbors for standardizing/normalizing point cloud. Then a series of EdgeConv operations are applied to this generated matrix. In the EdgeConv, as seen in Fig. 2 bottom right, a k-nearest graph for the points is created on which convolution operation is applied and the results are applied. Through the series of EdgeConv operations as seen in Fig. 2 top, information is extracted from local points and as layers progress, the semantically close points get closer. The graphs are dynamic because the structure of the graph changes after each EdgeConv operation. When the first EdgeConv is applied, the graph has points that are spatially near in the input space, and in every deeper EdgeConv layer, the points get semantically closer in the feature space. The neighborhoods are searched in Euclidean space and the clusters of analogous features are searched in the feature space, the effect on the segmentation task is positive [30]. The EdgeConv operation can be described as follows:

$$e'_{ijm} = ReLU(\theta_m.(x_j - x_i) + \phi_m.x_i) \tag{2}$$

Here i is the point to be calculated on, j are the set of neighborhood points, θ represents the learnable parameters, $(x_j - x_i)$ are the edges that give the local features, x_i gives the global features and these can be learned by MLP. The output of this convolution can be computed by an aggregation function that can be a sum or max operation. In the current approach, max has been used. Max is a symmetric function hence the output will be permutation invariant. There are two branches in the architecture where the top one is for classification and the bottom one is for segmentation where per class classification takes place by merging the global and local features generated after multiple passes from the EdgeConv layer.

B. GAPNet

The architecture of GAPNet is also inspired by PointNet. Having an attention layer in any architecture lets the model focus on very specific sections as well as help in making less use of resources, and GAPNet leverages it by having a novel addition in the model called GAPLayer. The GAPLayer is used for local structure representation using a directed graph constructed through KNN. It is multi-headed, which means M GAP layers process the input together to get sufficient structure information and stabilize the network. A single GAPLayer outputs attention features and graph features by learning self-attention and neighboring attention in parallel, as seen in the right of Fig. 4, to process different attention weights of the neighborhood. The attention feature $\hat{x_i}$ generated can be represented as:

$$\hat{x_i} = f(\Sigma_{j \in N_i}(\alpha_{ij} y'_{ij})) \tag{3}$$

where α is the normalized attention coefficient, y_{ij} denotes the edges and f is the ReLU activation function. M such layers are concatenated to generate multi-attention features and multi-graph features as seen in Fig. 4 left. Simply put, it calculates the contextual attention feature for every point. Fig. 3 shows the complete architecture of GAPNet where the structure resembles that of PointNet, where the spatial transform is now attention aware and the rest of the network has multi-layer perceptions of the point using multi-headed GAPLayer outputs, which are then processed by stacked MLP layers that eventually lead up to learning efficient local geometric information. The first GAPLayer does the classification task and the second one, which gives the global features, is used to perform the segmentation task. Two red arrowed arcs represent attention pooling operation from corresponding GAPLayer to generate a local signature

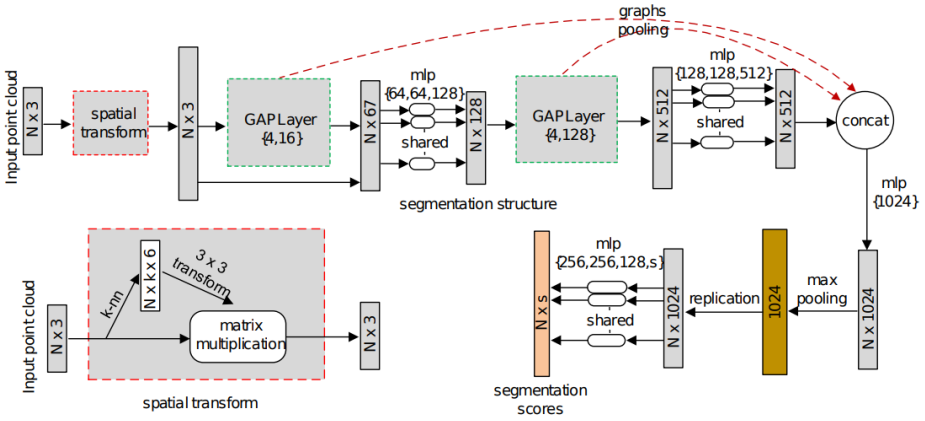

Figure 3. GAPNet Architecture [4]

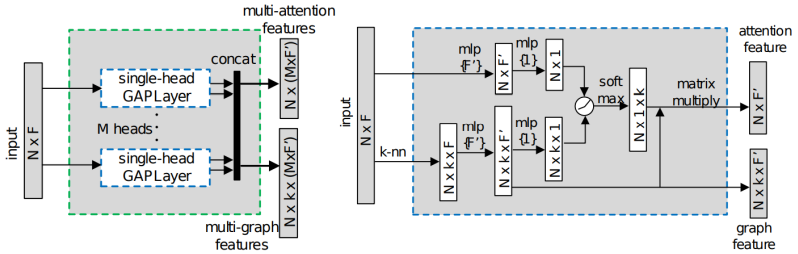

Figure 4. GAPLayer Architecture [4]

that is concatenated to intermediate layer for global feature generation [4] and it helps to make the network robust and improve its performance. The classification for each point is generated, and therefore the segmentation results.

C. RGCNN

This approach takes a different direction to process Point Clouds, following the spectral graph theory. They use three graph convolutional layers and each uses a three-step process, as can be seen in Fig. 5. The first one is graph construction. A fully connected graph is created using the edge weights, which are calculated using the distance between features of two points, and features are nothing but the data residing on the vertex, which is referred to as the graph signal. Combinatorial graph Laplacian is created using the weights which is defined from the adjacency matrix of the graph. The second step is graph convolution. The graph signal is filtered in the Graph Fourier Transform (GFT) domain using the graph laplacian and GFT coefficients are generated for each signal. Chebyshev approximation, to get truncated Chebyshev polynomials, is used to approximate this spectral filtering in order to reduce the computational complexity. The convolution of two signals is eventually calculated by the multiplication of their GFT coefficients, followed by inverse GFT. After performing the convolution, the third step is feature learning which is performed using MLP for the segmentation task. In each layer, the graph Laplacian is updated and encapsulates the structure of dynamic graphs adaptively. A loss function is proposed to add smoothness to the graph signal using a Graph-signal Smoothness Prior that helps in the regularization of learning. It can be defined as follows:

$$E(y^o, y^{'}) = -\Sigma_{i=1}^{n} y_i^o log(y_i^{'}) + \gamma \Sigma_{l=0}^{2} y_l^T \mathcal{L} y_l \qquad (4)$$

where y^o is the output score, $y^{'}$ is the ground truth label, y_l is the feature map of the l-th layer and γ is the penalty parameter for the smoothness term, set to 10^9 in the experiments [24]. The matrix learns relevant features, updates continuously and the model as a whole is permutation invariant.

IV. Results

In this section, we will discuss the results obtained from each model and how they compare to the state-of-the-art methods.

The experiment done by the three models on the ShapeNet dataset [3] is used for part semantic segmentation. It contains various representations of 3D objects. In the experiments done, 16 categories of objects are chosen to do part segmentation task.

The metric used to check for the accuracy of segmentation is mIoU. The IoU calculates the overlap ratio between the real area and the predicted area and mIoU is the average IoU

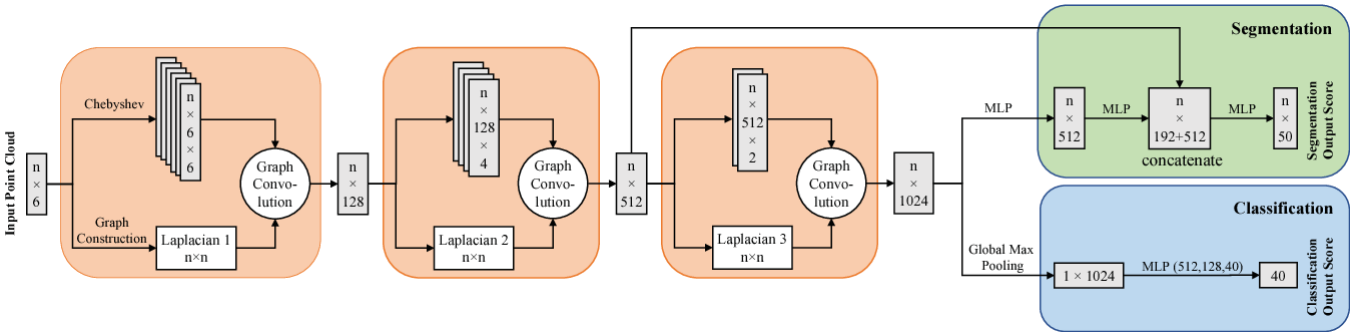

Figure 5. RGCNN Architecture [24]

Models	avg.	air.	bag	cap	car	cha.	ear.	gui.	kni.	lam.	lap.	mot.	mug	pis.	roc.	ska.	tab.
PointNet	83.7	83.4	78.7	82.5	74.9	89.6	73.0	**91.5**	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++	**85.1**	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	**83.7**	95.3	**71.6**	94.1	81.3	58.7	**76.4**	**82.6**
DGCNN	**85.1**	**84.2**	83.7	84.4	77.1	**90.9**	**78.5**	**91.5**	87.3	82.9	96.0	67.8	93.3	**82.6**	59.7	75.5	82.0
GAPNet	84.7	**84.2**	**84.1**	88.8	**78.1**	90.7	70.1	91.0	87.3	83.1	**96.2**	65.9	95.0	81.7	**60.7**	74.9	80.8
RGCNN	84.3	80.2	82.8	**92.6**	75.3	89.2	73.7	91.3	**88.4**	83.3	96.0	63.9	**95.7**	60.9	44.6	72.9	80.4

Table I
SEMANTIC PART SEGMENTATION RESULTS ON SHAPENET PART DATASET (MIOU% ON POINTS)

Models	**Model Size(MB)**	**Forward Time(ms)**
PointNet	40	25.3
PointNet++	**12**	163.2
DGCNN	21	94.6
RGCNN	22.4	**7.5**

Table II
FORWARD TIMES OF THE MODELS

based on each category.

The comparison results for the task are shown in Table I. When it comes to the average of all the objects, DGCNN performs as good as the state-of-the-art model PointNet++. Looking at the individual classification scores of each object, DGCNN outperformed other models in 5 objects, marginally higher than PointNet++. The overall performance of GAPNet is almost equal to the other best performing models and outperforms in 5 different objects as well. RGCNN however is lower than the best model by 0.8% and has the best scores for 3 objects. It is also to notice that for object pis. and roc., the model performs much worse than for any other object however maintains a comparable score in the other categories with the other models.

When it comes to forward timing and model sizes, four models report these parameters for semantic segmentation task, namely PointNet, PointNet++, DGCNN and RGCNN as shown in Table II. Both DGCNN and RGCNN have similar model size improvement from PointNet++, however, it is worth a notice that RGCNN has a forward time of 7.5 ms, a massive improvement as compared to the other models, therefore having much lower computational complexity. RGCNN is also much more robust against noise and low density in point clouds as compared to PointNet when tested with Gaussian noise and perturbed input.

V. DISCUSSION AND CONCLUSION

Looking at the strengths of each model, DGCNN performs close to the state-of-the-art results and the property of being able to generate and update graphs dynamically gives it an edge for a better understanding of the neighborhood vertices. GAPNet incorporates the attention feature mechanism, as done in many other computer vision tasks, hence generating more relevant features using the relevant vertices information. The performance is comparable to the state-of-the-art models as well. RGCNN uses a different approach to spectral graph theory and utilizes efficient functions for the model to learn efficiently and robustly. Except for a few hiccups in the results, this model performed close to the best results and the property of it being robust against noise makes it much more applicable to real-world applications because the data in the real-world will always be noisy. The forward time of the model is also much better, and therefore it is more practical to work with.

Looking at the weaknesses, the computational complexity for DGCNN is still high as a pairwise calculation for KNN is an expensive task. When looking at GAPNet, it is difficult to suggest the scalability of this approach when the graph size increases. For RGCNN, although it comes with strong positive points, it is still unknown why the performance was much worse for the two objects and how these results would imply when extended to even more object segmentation

tasks.
Based on the requirements of the applications and computational power available, a relevant model can be chosen from these models.

Currently, GNNs have been applied to many fields, such as network analysis, recommendation systems, traffic prediction, biochemistry, natural language processing and computer vision [2]. We saw through this paper what GNNs are and how they can be used on point clouds for semantic segmentation task. point clouds can be used by graph neural networks, discussed the basics of the concepts and delved into the three different architectures of DGCNN, GAPNet and RGCNN. We also saw the results and finally discussed their implications along with some pros and cons of each model. Furthermore, more research should be done on the scalability of the models to real-life scenarios and even bigger graph data. There is a possibility of nodes having long dependencies among the vertices and the approaches to do the same can be explored in further research.

REFERENCES

[1] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3547–3555, 2015.

[2] Z. Z. W. Z. Cao, P. Applications of graph convolutional networks in computer vision. *Neural Computing and Applications*, 2022.

[3] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

[4] C. Chen, L. Z. Fragonara, and A. Tsourdos. Gapnet: Graph attention based point neural network for exploiting local feature of point cloud. *CoRR*, abs/1905.08705, 2019.

[5] L. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. *CoRR*, abs/1712.04837, 2017.

[6] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. *CoRR*, abs/1511.03339, 2015.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[8] Y. Guo, Y. Liu, T. Georgiou, and M. Lew. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, 7, 06 2018.

[9] H. Li, P. Xiong, J. An, and L. Wang. Pyramid attention network for semantic segmentation. *CoRR*, abs/1805.10180, 2018.

[10] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[11] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan. Semantic object parsing with graph LSTM. *CoRR*, abs/1603.07063, 2016.

[12] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[13] Y. Lu, Y. Chen, D. Zhao, and J. Chen. Graph-fcn for image semantic segmentation. In H. Lu, H. Tang, and Z. Wang, editors, *Advances in Neural Networks – ISNN 2019*, pages 97–105, Cham, 2019. Springer International Publishing.

[14] P. Luc, C. Couprie, S. Chintala, and J. Verbeek. Semantic segmentation using adversarial networks. *CoRR*, abs/1611.08408, 2016.

[15] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[16] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.

[17] P. Pradhyumna, G. P. Shreya, and Mohana. Graph neural network (gnn) in image and video understanding using deep learning for computer vision applications. In *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 1183–1189, 2021.

[18] 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.

[19] 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.

[20] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[21] W. Shi and R. Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[22] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. *CoRR*, abs/1505.00880, 2015.

[23] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. *CoRR*, abs/1505.00880, 2015.

[24] G. Te, W. Hu, Z. Guo, and A. Zheng. RGCNN: regularized graph CNN for point cloud segmentation. *CoRR*, abs/1806.02952, 2018.

[25] F. Visin, K. Kastner, A. C. Courville, Y. Bengio, M. Matteucci, and K. Cho. Reseg: A recurrent neural network for object segmentation. *CoRR*, abs/1511.07053, 2015.

[26] 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.

[27] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. *CoRR*, abs/1502.05908, 2015.

[28] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019.

[29] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[30] J. Zhang, X. Zhao, Z. Chen, and Z. Lu. A review of deep learning-based semantic segmentation for point cloud. *IEEE Access*, 7:179118–179133, 2019.