Reaction Report for "Dynamic Graph CNN for Learning on Point Clouds"

Nishant Raj

What I like about this paper?
One of the major downsides in Point Net is that even though they can maintain permutation invariance, they still lose out on a very important parameter i.e., geometric dependencies and relationships between different points. DGCNN overcomes this huge limitation. The idea of EdgeConv helps in constructing local graphs in the network. Since graphs are updated at each step during training, unlike fixed graph structure CNNs, it is also able to learn to group points in a point cloud. Thus, DGCNN can get the best of both i.e., permutation invariance, geometrical dependency capture and grouping of points. One good part about the paper is the extensive depth of experimentation that has been carried out for classification and segmentation tasks. For example, they show that even the with fixed k-NN graph, it can outperform the PointNet++ architecture and that using the concept of centralization (concat of xi and xi-xj instead of xi, xj), they show that semantic based features do help in improving accuracy. One another part of the paper that they have mentioned is that of being translationally invariant which can be triggered based on use-case requirements. One final aspect is that the design of architecture of dynamic graph CNN architecture makes it possible to be used for other tasks such as document retrieval and medical image processing as mentioned at the end of paper.

What I do not like about this paper? While using the edge convolution network, assumption is that edges would help to extract meaningful information always. However, consider a case where all the neighboring points are exactly similar. In this case, the edges would be same and would not provide any meaningful information. Also, if the neighbors are very far away or non-uniform, it would not be a good idea to use KNN. Secondly, since the architecture of DGCNN is going to be huge, there would be many trainable parameters due to the presence of transformation network and hence this would require an extensive degree of compute. This is something that must be worked upon. In the end, though the authors have done the depths of experimentation for classification and segmentation tasks, it would have been interesting to see the breadth of applications of DGCNN as well like application in scene understanding tasks.

Future Directions:

There are different directions that can be investigated. Since we would always want to maintain permutation invariance, we can use self-attention networks which essentially acts as a set operator. Also, points are a set of positional coordinates in 3D-space, and this would help us to capture important positional embedding around each of the points through self-attention mechanism. Apart from these points, I would like to investigate upon how the entire architecture of DGCNN works for massive tasks like 3D scene understanding. If this is very compute intensive, another aspect that can be investigated is working on reduction of the size of network graphs in DGCNN and to try and reduce the number of trainable parameters. This can be done by linking of hierarchical features across different dynamic graphs that are generated. Also, instead of using transform block, we can try and use MLP to estimate the positional invariances. There are other tasks which paper mentions in the end: investigation of higher order relationship between points instead of using pairwise distances. However, I feel that the gain would be marginal in this case.