3D Deep Learning

MADRID 2018
iROS

Creating and Understanding 3D Annotated Scene Meshes

Robot Vision with Point Cloud
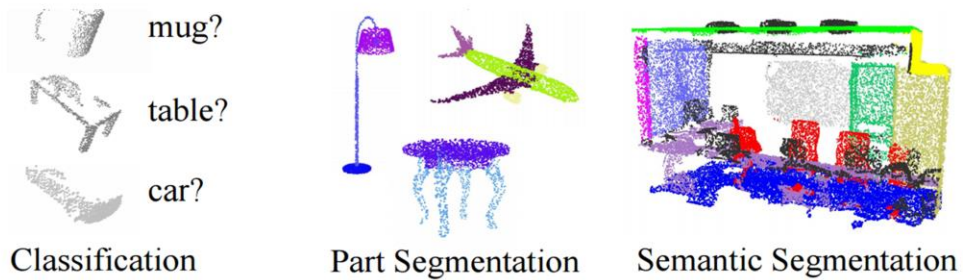
mug?
table?
car?

Classification

Part Segmentation

Semantic Segmentation

Qi et al, PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, CVPR 2017

Let us imagine that massive data is now prepared from the acquisition and annotation pipeline. Our goal is to utilize such data for training.

Common tasks in computer vision for robotics include object classification, object and scene segmentation, and so on.

As mentioned in the reconstruction, there could have multiple approaches for representing 3D data. Here we choose to represent the data with 3D point cloud, as this is a true 3D representation and it is view independent. Deep learning with point cloud is also an active research topic. Therefore, we would like to provide a more detailed explanation in this part to motivate more future works in this topic.

Challenges in Deep Learning with Point Cloud

- Points are unordered
  - Sorting
  - Mapping to order invariance
  - Recurrent neural network

- Convolution with a point cloud?

- Down-sampling and up-sampling

3

Deep learning with 3D point cloud is a challenging problem. Particularly, if we map each component in existing 2D convolutional neural network to 3D point cloud, there will have the following issues.

First, point cloud is mathematically a set. {x1, x2, x3} and {x1, x3, x2} are different vectors but basically represent the same point cloud. Somehow the network has to take this into consideration. Possible solutions are:
- Sort the point cloud
- Map the point cloud to something that is order invariance
- Use recurrent neural network which can deal with a data sequence. If the network is trained enough with different sequence, it will become robust to different ordering.

In this talk, we will highlight methods that use each of these possible solutions.

Second, unlike an image that can be represented as a 2D grid where convolution is straightforward, points can scatter arbitrarily in the 3D space, and defining a convolution (a weighted sum in principle) requires nearest neighbour queries.

Third, max pooling is not straightforward to define with point clouds. Designing

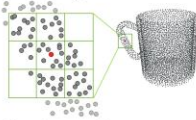networks for convolution and deconvolution is therefore challenging.

There has been on-going research works that tackle such challenges.

Courtesy of Hua et al., Pointwise Convolutional Neural Networks, CVPR 2018

Let us start with a simple approach by Hua et al. in the coming CVPR 2018.

The idea is very simple. Let's directly define how to perform convolution with 3D point cloud. They named it pointwise convolution.

The tricky part in this work is how to define the convolution, its derivative, and integrate nearest neighbours in each convolution so that it can be robust for training.

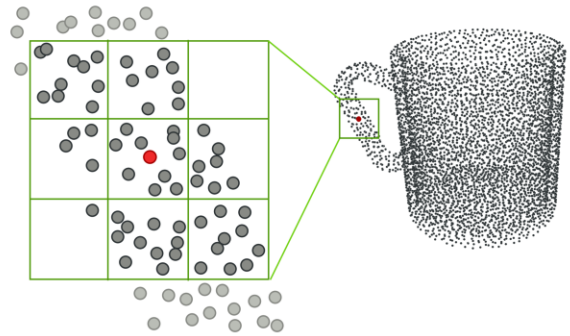## Pointwise Convolution

- At each point, center a grid
- Take points in the grid for convolution
- Points each cell have the same weight
- Nearest neighbor query on the fly

Courtesy of Hua et al., Pointwise Convolutional Neural Networks, CVPR 2018

5

The key component is a special convolution which is called pointwise convolution. It is named to mean that we apply the convolution at every point in the point cloud, and there is no pooling, up/down-sampling done in the entire pipeline.

At each point, e.g., the red one, we center a grid, in this case, 3x3, and determine which points fall into the grid and into which cells.
All points in the grid takes the same weight in the convolution.

To handle different size of the receptive field, we can use different size for this 3x3 grid in different layers.

To handle point orders, we sort all points in XYZ coordinates.

A tricky issue is how to implement nearest neighbour search efficiently. Here, we make use of a uniform grid as an acceleration data structure, instead of using hierarchical data structure such as kd-tree or octree as we found that, usually with 2048 or 4096 points, the performance of the uniform grid is better on modern CPUs. Also, such implementations are also straightforward on GPUs and CUDA.

We implement this operator in Tensorflow.

Pointwise Convolutional Neural Network
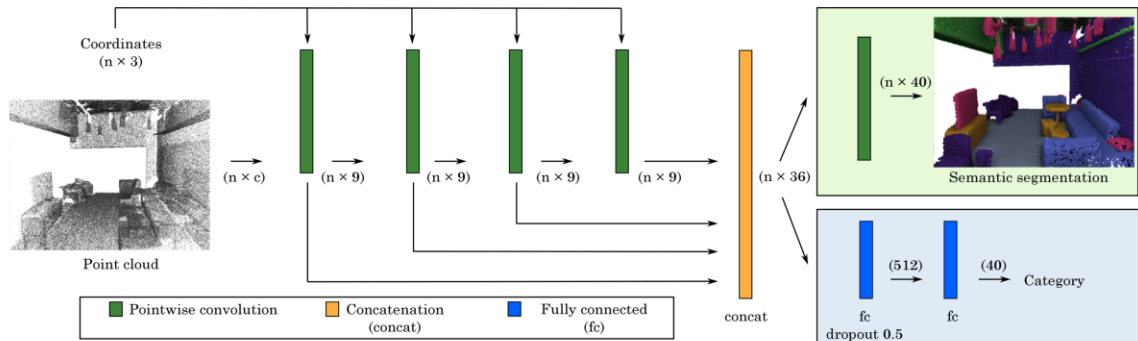
Courtesy of Hua et al., Pointwise Convolutional Neural Networks, CVPR 2018

With the availability of the new convolution, we can see that it is now possible design convolutional neural network for point cloud. And here, we use a very simple design, with just a four layers of convolution, followed by a concatenation before branching for semantic segmentation or classification, respectively.

Note that in this design, the network looks almost similar to traditional neural networks for 2D images.

Training and Testing
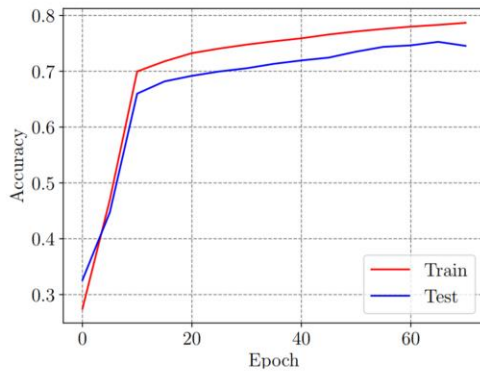
(a) Scene segmentation
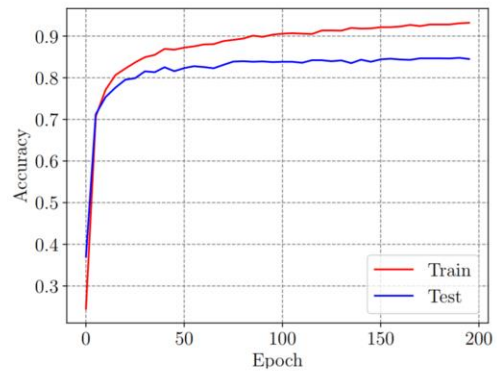
(b) Object recognition

Here is the training and testing convergence plot. The red curve denotes training, and the blue curve denotes testing.

We see that the pointwise convolution works and the learning converges.

Object Classification

| Base	Concat.	À-trous	SELU	Dropout	Accuracy
✓					78.6
✓	✓				78.0
✓		✓			75.0
✓	✓	✓			82.5
✓			✓		81.7
✓	✓		✓		81.9
✓	✓		✓	✓	85.2
✓	✓	✓	✓	✓	86.1

8

Let's look at the results of the object classification task in more details. Here we show an experiment with different network design. The base example without any fancy addons has about 78.6% of accuracy, which is a bit low compared with the state-of-the-arts.

When we add feature concatenation from all layers, a-trous (dilated) convolution to increase the receptive field, SELU to stabilize the training, and dropout to avoid overfitting, we end up getting 86.1%, which is pretty close to the state-of-the-art, given the extremely simple network design and the convolution operator.
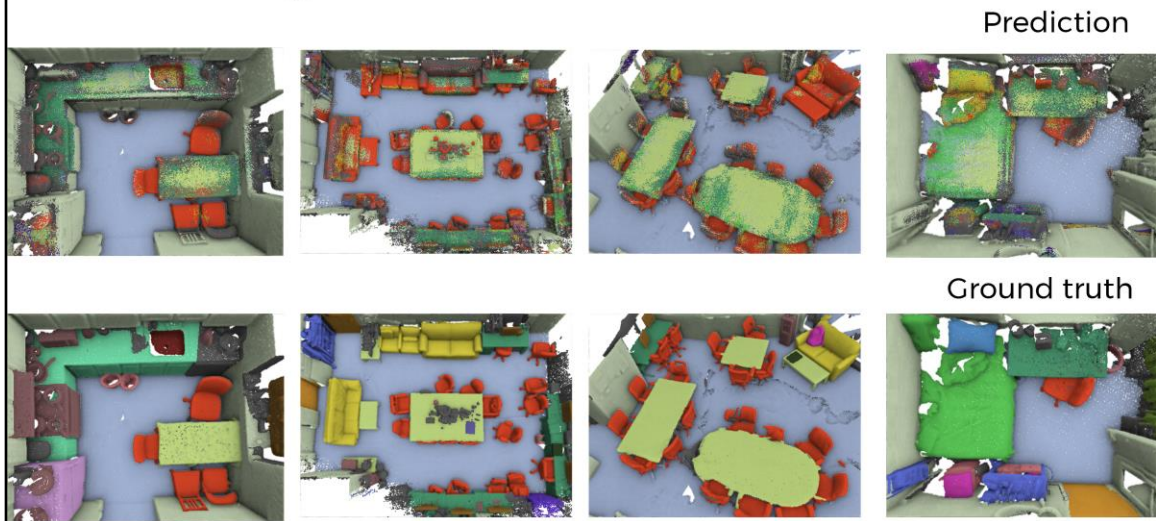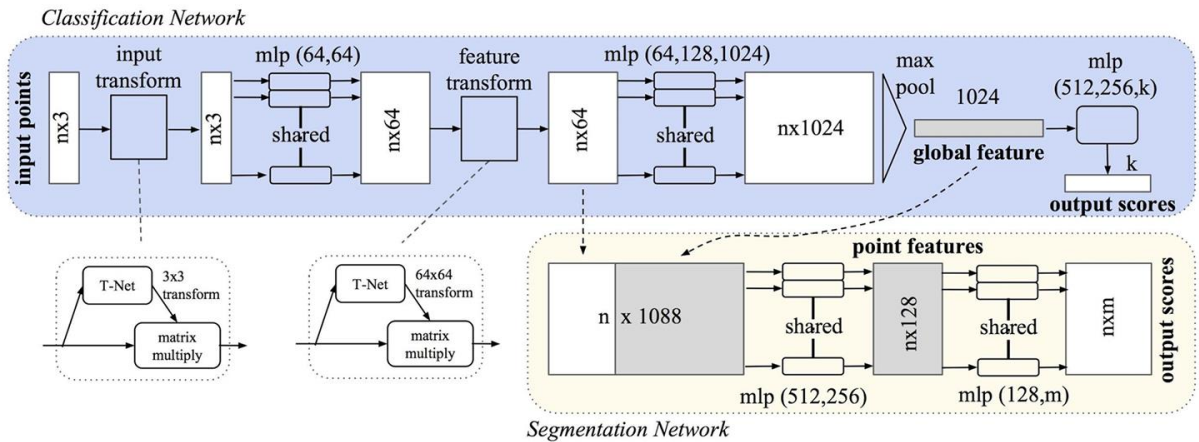
Here is the result of semantic segmentation from our neural network.
The training and testing are both done on the SceneNN dataset in this example, to demonstrate the output of our 3D acquisition and annotation earlier.

We see that the results are quite accurate for common classes such as table, chair, floor, and wall. There still has some noise and ambiguity, which would be an interesting future work. For example, we can filter out some noise if we have a constraint to enforce the coherency on labels in a local region.

(We will show how to do this with a conditional random field at the end of this talk, where we discuss an application for real-time semantic segmentation.)

PointNet

Classification Network

input points | nx3 | input transform | nx3 | mlp (64,64) shared | nx64 | feature transform | nx64 | mlp (64,128,1024) shared | nx1024 | max pool | 1024 global feature | mlp (512,256,k) | k output scores

T-Net | 3x3 transform | matrix multiply

T-Net | 64x64 transform | matrix multiply

point features

n x 1088 | shared mlp (512,256) | nx128 | shared mlp (128,m) | nxm | output scores

Segmentation Network

Qi et al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, CVPR 2017 10

Now, after discussing a very simple approach in 3D deep learning with point cloud, let us look at a more complex approach that is one of the state-of-the-arts. This is one of the pioneering algorithm for deep learning on point cloud.

Let us look at the blue part for object classification.

The idea here is to employ a symmetric function that can turn the input point set into an intermediate representation with order invariance. A symmetric function will always produce the same result despite that the input parameters can be in any order. In PointNet, they implement this using a maxpool operation, resulting in a global feature vector as shown in the diagram.

There are two small blocks that applies a T-Net. Basically, the first one is for aligning the input point cloud as it can have arbitrary orientations. This is mainly for semantic segmentation use.

The second T-Net block is with the same idea, but this time for aligning in the feature space.

Finally, the yellow block denotes how to perform semantic segmentation with PointNet. Each point has its feature vector concatenated with the global feature

vector, before being passed to some multi-layer perceptrons to produce final per-point label prediction.

With this design, PointNet achieves state-of-the-art results in object classification and semantic segmentation.

PointCNN
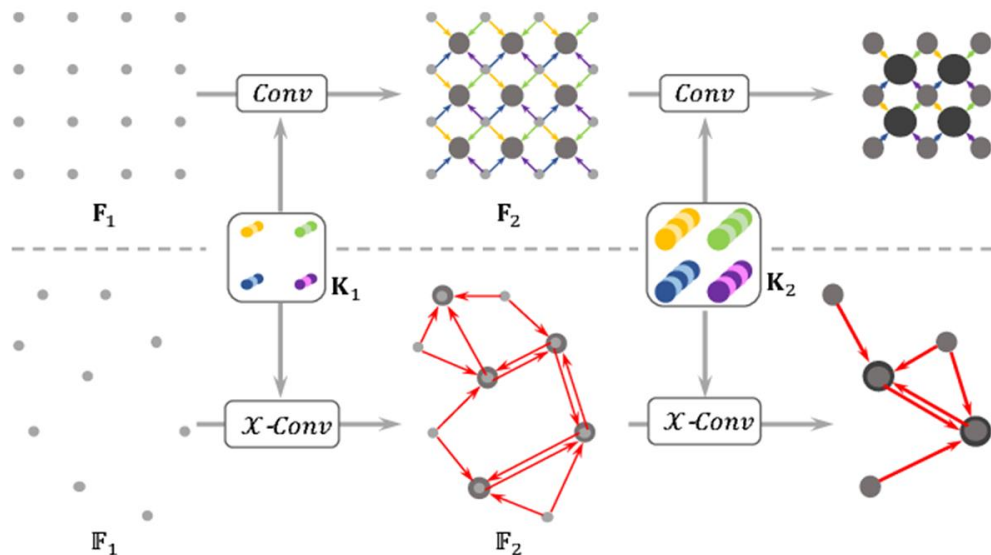
Li et al., PointCNN, arXiv 2018

PointCNN is a recent work (on arXiv) that has similar ideas to PointNet in handling point order. Instead of using a symmetric function to learn order invariance, they propose to use an MLP to estimate the transformation to canonicalize the order, which they term X-transformation. After the transformation, the point cloud is downsampled (randomly or using farthest point sampling), and the convolution continues.

This process is illustrated here.
The top row represents an ordinary convolution in 2D, applied on a regular grid.
Gradually, a 4x4 grid turns into a 3x3 grid, and then a 2x2 grid.
At the same time, the number of channels at each point is increased to store richer information contributed from the neighbors during convolution. This is represented in bigger points.

The same idea applies to the convolution designed by PointCNN. With a point set, after each convolution with the operator X-conv, a subset of points are retained by downsampling, and contained richer information (emphasized by bigger points).

So here, what we need to optimize during training is this X-transformation layer.

Recurrent Slice Networks (RSNet)
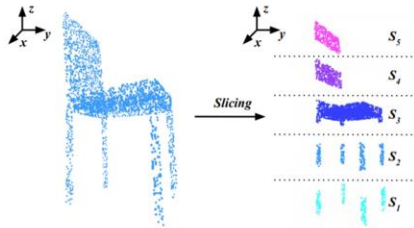
Huang et al., Recurrent Slice Networks for 3D Segmentation of Point Clouds, CVPR 2018
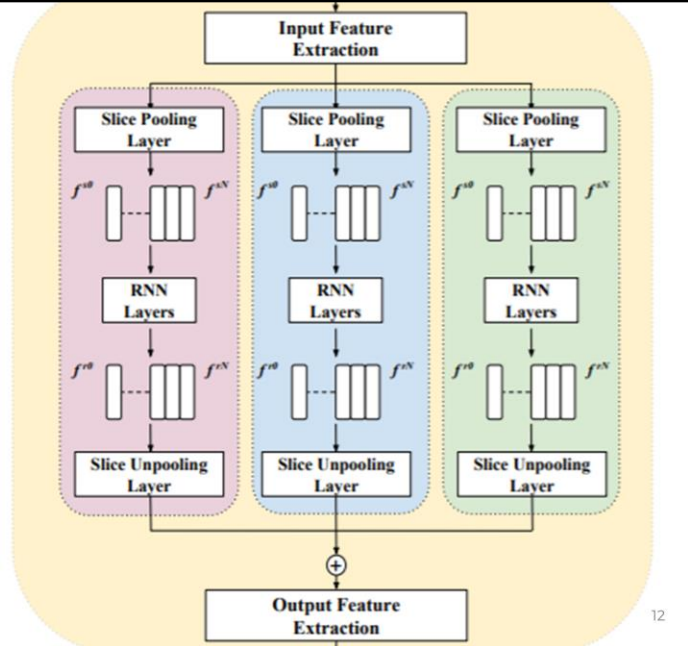
Since PointNet only makes use of global features in the learning, there has been on-going efforts to exploit local feature for point cloud deep learning. There have been several possible approaches by May 2018. Here we discuss the main idea of a few recent state-of-the-arts in this direction. Most of them are papers that will appear in CVPR 2018.
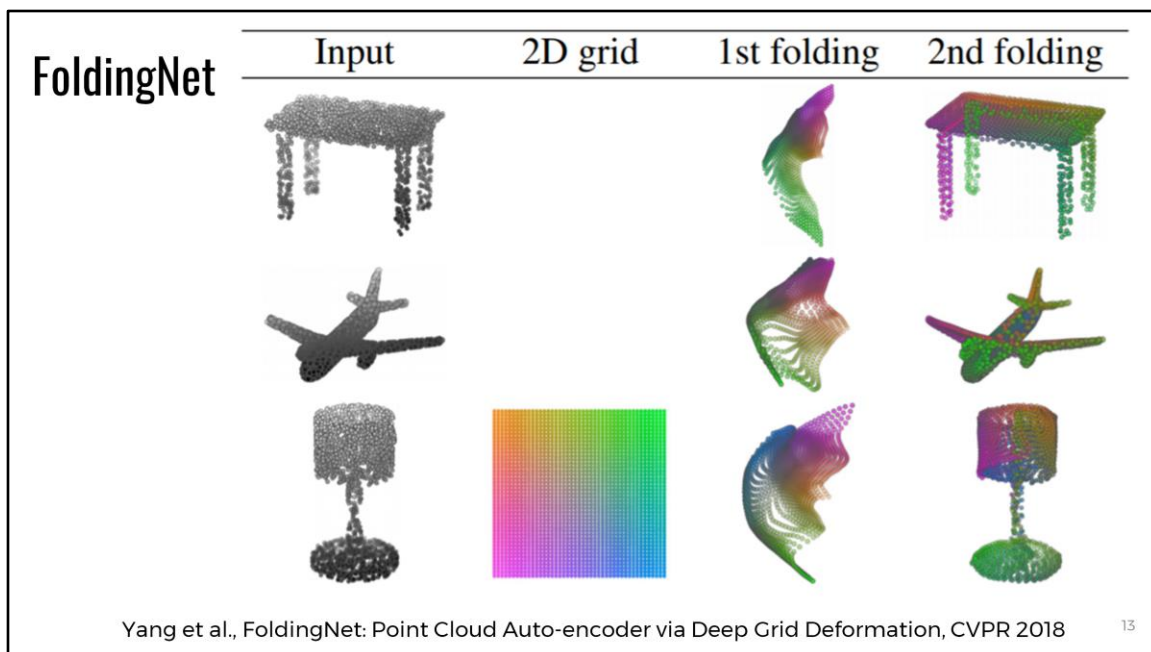
Here is recurrent slice networks. Recall that at the beginning of this talk, we mentioned about different strategies to handle point orders. Pointwise convolution relies on sorting; PointNet learns a function for order invariance. This work takes the recurrent neural network approach to tackle the point order problem.

In RSNet, the slice pooling layer is the key to turn unordered points into ordered sequence. Take the chair on the left as an example. Basically, we group input points based on their Z coordinates. In each group, a global feature of the group is extracted. After that, all group features are concatenated. Since the Z axis defines the orders of the group, the concatenated features is ordered. The same slicing is done for X and Y axis, respectively.

The features after slicing are fed to RNN layers which then learn the inter-

dependencies between the slices.

The unpooling layer is basically the reverse the slicing pooling layer. The feature of each slice after RNN is copied back to each point.

FoldingNet — Input | 2D grid | 1st folding | 2nd folding

Yang et al., FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation, CVPR 2018

An interesting recent work is FoldingNet which learns a 3D-2D mapping for point clouds.

The key observation here is that 3D point cloud are often sampled from surfaces, which is in fact a 2D manifold. Therefore, a mapping from 3D to 2D could be exploited. In this work, the authors deform a canonical 2D grid into 3D point cloud by using a neural network.

Point Cloud Local Structures by Kernel Correlation
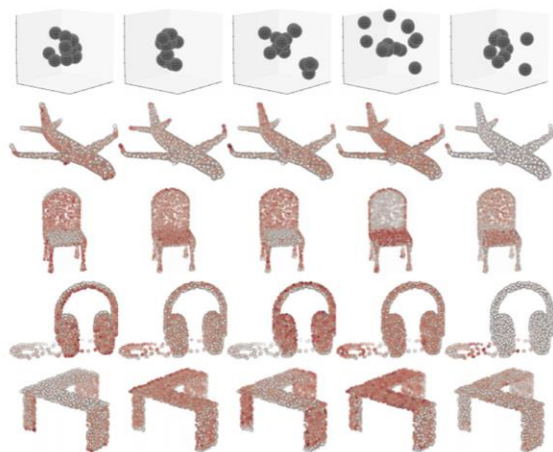
Shen et al., Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling, CVPR 2018 [14]

The idea of this work is to learn kernels for point cloud convolution that characterizes local geometric features such as corner, planarity, and convexity. As shown in this figure, the top row illustrates the kernels, and the bottom rows demonstrates the convolution results of the point cloud with the corresponding kernels.

We can see that here it is similar to the 2D case where the filters in 2D convolution is optimized to learn image features such as edges and corners.
For each row in the bottom, we can see the effect of each kernel. In general, each kernel highlights a specific geometry feature (parallel to ground or gravity axis, inner/outer edges, etc.).

EdgeConv

e_{ij}

$h_\Theta()$

$\mathbf{x}_i$ $\mathbf{x}_j$

$\mathbf{x}_{j_{i3}}$ $\mathbf{x}_{j_{i2}}$ x_i $\mathbf{x}_{j_{i4}}$ $\mathbf{x}_{j_{i1}}$ $\mathbf{x}_{j_{i5}}$

$EdgeConv$

$\mathbf{x}_{j_{i3}}$ $e_{ij_{i3}}$ $e_{ij_{i2}}$ $\mathbf{x}_{j_{i2}}$ $e_{ij_{i1}}$ x'_i $e_{ij_{i4}}$ $\mathbf{x}_{j_{i4}}$ $e_{ij_{i5}}$ $\mathbf{x}_{j_{i1}}$ $\mathbf{x}_{j_{i5}}$
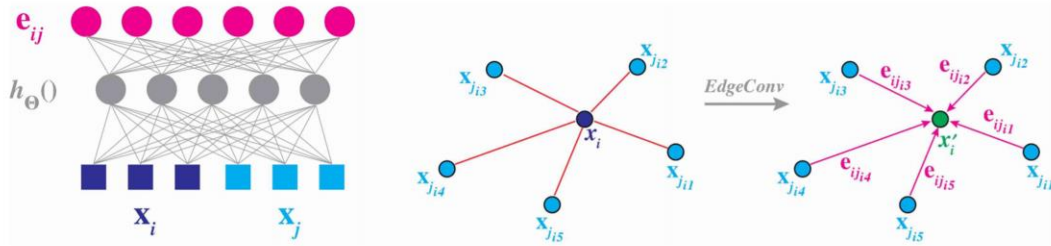
Wang et al., Dynamic Graph CNN for Learning on Point Clouds, arXiv 2018

EdgeConv is an generalization of convolution where the relations between point pairs in a local neighbourhood can be optimized during learning.

The idea is as follows. For each point, constructing a local neighborhood graph. An edge value is a function of two points h_theta(xi, xj), where parameter theta can be optimized during learning. If we choose h_theta(xi, xj) = theta_j * xj, we get back the original convolution.

SUMMARY	Order	Convolution	Applications		MN40	S3DIS
Pointwise	Sort	3D, with nearest neighbor search	O S		86.1	-
PointNet	Symmetric function	Per-point using multi-layer perceptron	O S P		89.2	47.6
PointNet++	Symmetric function	Split into groups, each group has a PointNet	O S P		90.7	-
PointCNN	X-transform	Transformation and point downsampling	O S		91.7	62.7
RSNet	Recurrent network	Recurrent network	S P		-	51.9
FoldingNet	Symmetric function	Mapping 3D points onto 2D grid	O (Unsupervised)		88.4	-
Shen et al.	Symmetric function	Kernel correlation to learn corners, planarity	O P		91.0	-
Wang et al.	Symmetric function	EdgeConv: weight between a point and its neighbors	O S P		92.2	56.1

16

Now let's summarize the methods we discussed into a table. We see that there is an on-going research effort for point cloud deep learning, and this could be an important direction for applications in robotics and robot vision. This table is by no meant complete, and will probably be outdated soon. ☺

The order column is the key technique to canonicalize the order of the points.

The convolution column gives a brief summary about the method, mainly about convolution.

The application column shows the experiments done in the papers using the corresponding technique.
Table of notation: O for object classification, S for scene segmentation, and P for object part segmentation.

The last two columns reported the accuracy of the object classification and scene segmentation task.
Scene segmentation metric is mean IoU (%).

In future work, beySo, I hope you have a big picture about deep learning with 3D point cloud, and the state-of-the-arts.
ond inventing new techniques to achieve higher accuracy, an important task is more training and testing datasets to benchmark such techniques.

Real-time Semantic Segmentation

- On-the-fly 3D segmentation and reconstruction
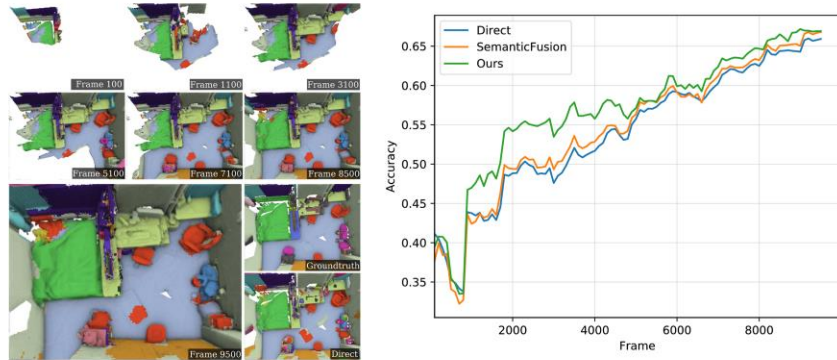- Using higher-order constraints from structures and objects
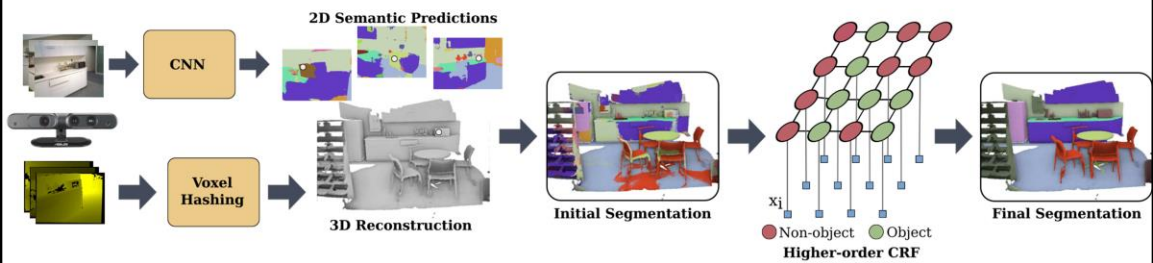
Pham et al., Real-time Progressive 3D Semantic Segmentation for Indoor Scenes, arXiv 2018

Real-time Semantic Segmentation

- Resolving semantic segmentation error while scanning
- Extensive evaluation on large-scale indoor scenes

Comparison to SemanticFusion (SF)

ID	Direct Class	SF Class	Ours Class	Ours Instance	ID	Direct Class	SF Class	Ours Class	Ours Instance
011	0.770	0.776	**0.800**	0.521	205	0.635	**0.642**	0.629	0.508
016	0.607	0.625	**0.680**	0.342	206	0.766	**0.778**	0.775	0.417
030	0.584	0.597	**0.658**	0.568	207	0.559	0.568	**0.596**	0.172
061	0.751	0.777	**0.809**	0.591	213	0.539	0.551	**0.561**	0.478
078	0.497	0.515	**0.535**	0.349	223	0.669	0.689	**0.729**	0.409
086	0.622	0.646	**0.668**	0.350	231	0.655	0.656	**0.660**	0.692
089	0.558	0.573	**0.618**	0.236	243	0.538	0.553	**0.595**	0.144
096	0.659	**0.668**	0.666	0.265	255	0.423	0.439	**0.558**	0.486
098	0.601	0.600	**0.623**	0.321	272	0.602	**0.614**	0.612	0.203

Using Other 2D Segmentation Networks

| Acc.	SegNet		FCN-8s		SSCNet		Acc.	SegNet		FCN-8s		SSCNet	
ID	Base	Ours	Base	Ours	Base	Ours	ID	Base	Ours	Base	Ours	Base	Ours
011	0.747	**0.837**	0.667	0.743	0.475	0.497	205	0.610	0.678	0.562	0.661	0.332	**0.680**
016	0.556	0.714	0.580	0.623	0.648	**0.798**	206	0.659	0.812	0.626	0.828	**0.861**	0.834
030	0.554	0.668	0.584	**0.704**	0.505	0.510	207	0.673	0.734	0.556	0.589	0.596	**0.634**
061	0.549	**0.841**	0.324	0.457	0.700	0.693	213	0.597	**0.683**	0.579	0.671	0.543	0.552
078	0.542	**0.666**	0.551	0.663	0.515	0.588	223	0.648	0.758	0.693	**0.760**	0.644	0.639
086	0.587	**0.686**	0.491	0.631	0.543	0.517	231	0.624	0.697	0.621	**0.741**	0.531	0.621
089	0.581	0.651	0.605	**0.667**	0.491	0.631	243	0.655	0.667	0.663	0.663	0.591	**0.734**
096	0.615	**0.683**	0.577	0.619	0.631	0.658	255	0.521	0.654	0.577	**0.718**	0.547	0.661
098	0.653	0.656	0.709	**0.740**	0.314	0.706	272	0.607	0.703	0.586	0.706	0.698	**0.777**

20

Instance Segmentation
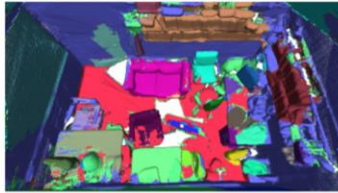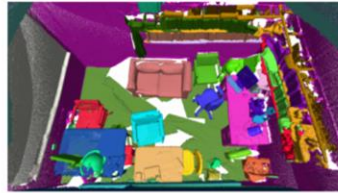
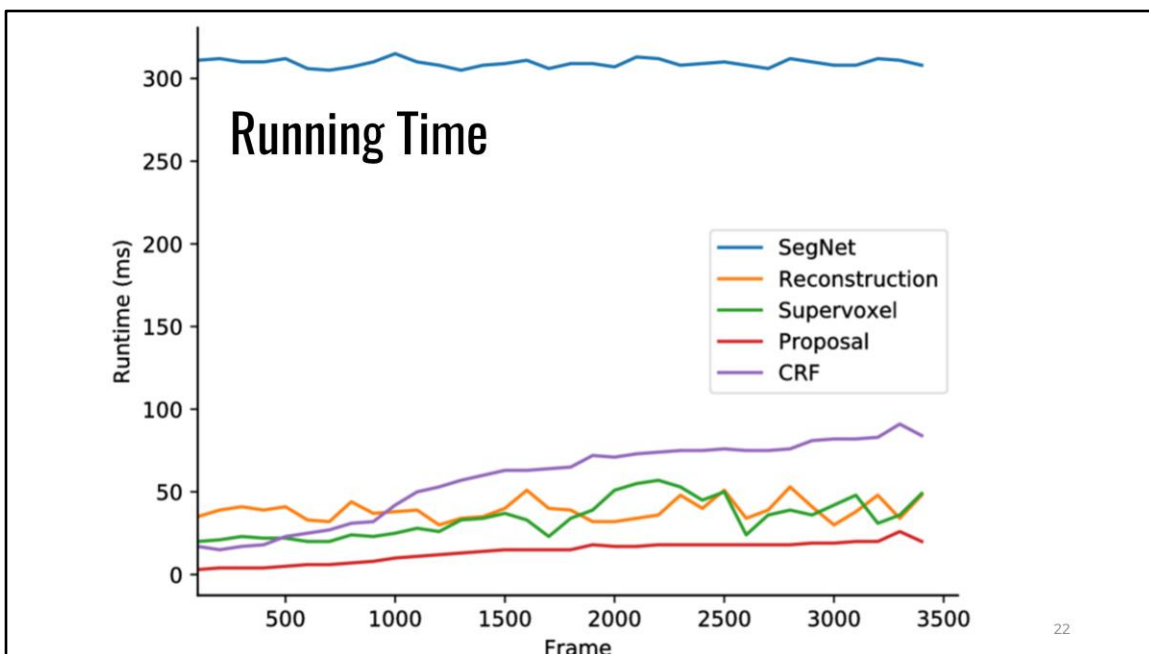Prediction · Ground truth
Prediction · Ground truth

Running Time
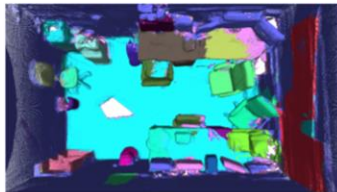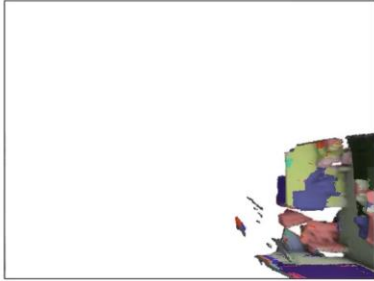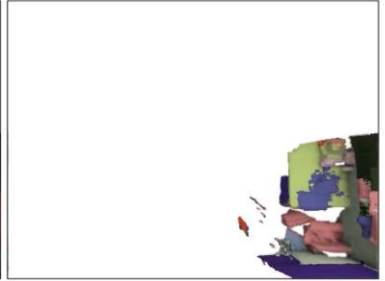
SceneNN/030: Progressive Segmentation

(a) Direct   (b) SemanticFusion   (c) Ours

Future Works

- Additional cues for point cloud deep learning: edge, triangle

- 3D point cloud networks for real-time semantic predictions

- Apply point cloud learning to object pose estimation, instance segmentation

24

In summary, point cloud deep learning is just a start. There will have several interesting future works that we can investigate. Here we list a few potential ideas.

Panel Discussion

Let's now wrap up today's talk with a panel discussion and Q&A.

Acknowledgement

- Tan-Sang Ha, Quang-Trung Truong, Fangyu Lin, Guoxuan Zhang for helping with data capture and tool development.
- Minh-Khoi Tran, Tian Feng, Zhipeng Mo, Benjamin Kang, Daniel Teo, Xuequan Lu, William Lai for helping with user study.