Dynamic Graph CNN for Learning on Point Clouds

Jae Ha Kim (2019-22370)

Seoul National University

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. ACM Trans. Graph. 38, 5, Article 146 (November 2019), 12 pages.

ComputerVisionLab
Seoul National University

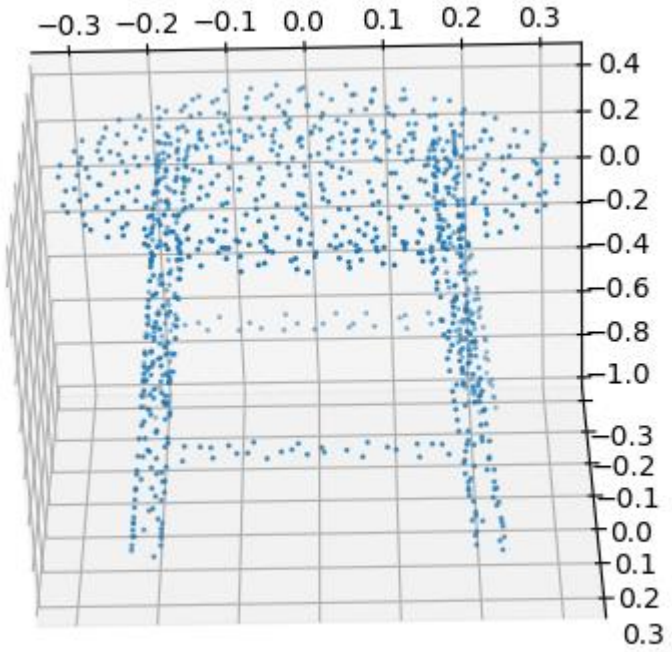
Index

1. Property of Point Cloud
2. Details of EdgeConv
3. Experimental Results on EdgeConv
4. Code implementation and actual experimental results

What is Point Cloud?

Chair?

What is Point Cloud?



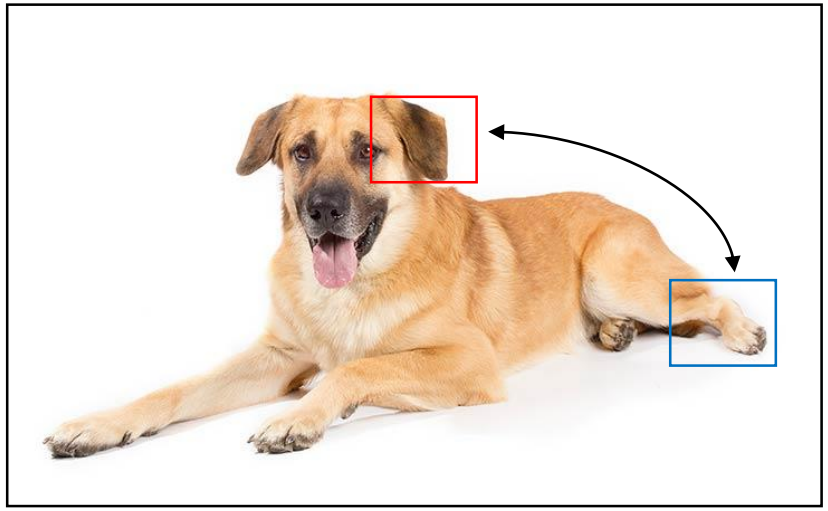
ModelNet40 (1024 points)

-0.1	-0.5	0.2
-0.2	0.7	0.5
0.3	0.1	0.2
⋮		
1.0	-0.2	0.1
0.1	0.3	0.5
-0.4	0.0	0.3

1024 x 3

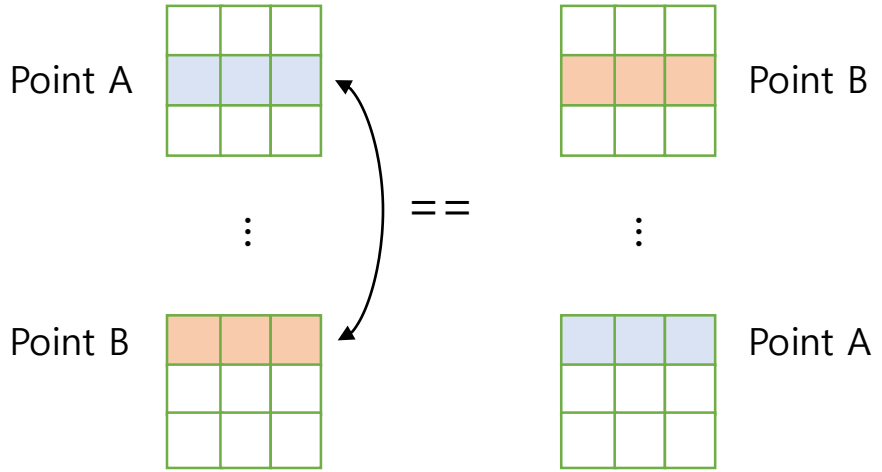
Difficulty to deal 3D point cloud

image



regular

3D point Cloud



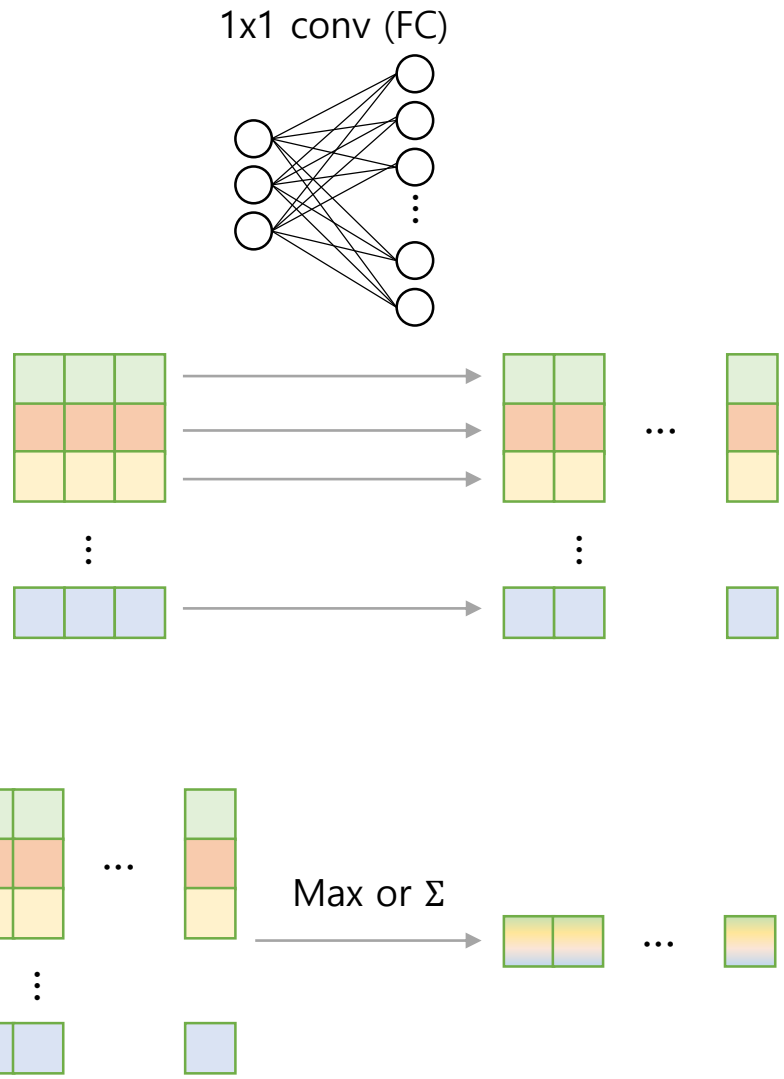
irregular

PointNet [Qi et al. 2017b]

Achieve **permutation invariance** by doing …

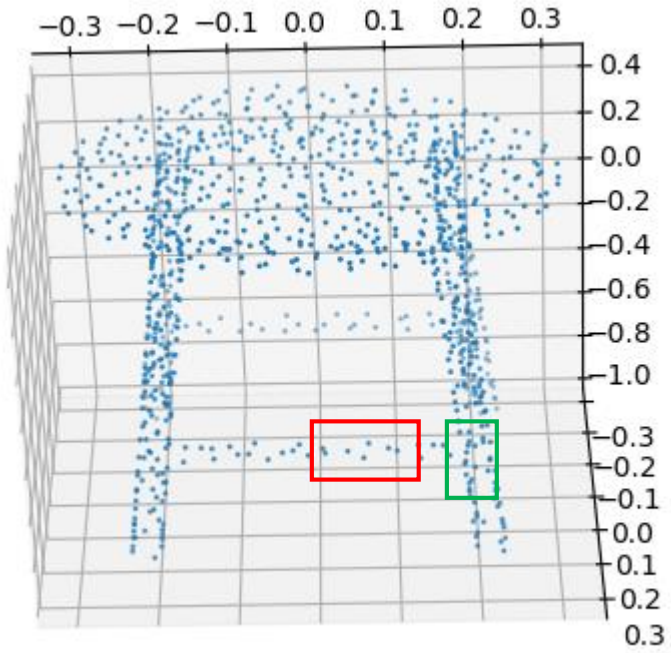
1. Operate each point independently
   - 1x1 convolution is used
   - Each point do not affect other points!

2. Use Symmetric aggregation function
   - Max or Σ function is used



Limitation of Independent operations

Lack of using local feature !

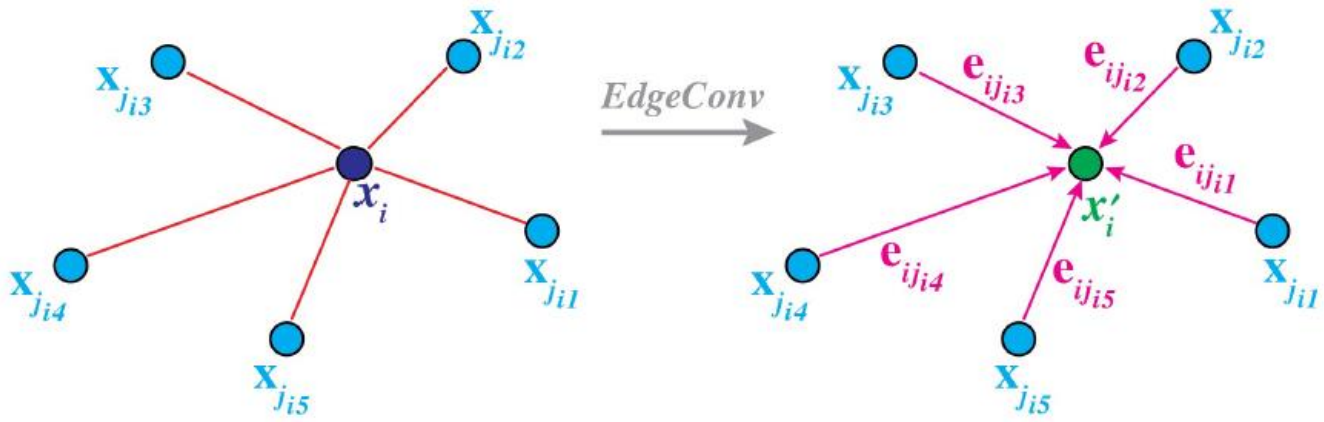


[1] PointNet++ [Qi et al. 2017c], LocalFeatureNet[Shen et al. 2017c]

Proposed method: EdgeConv

- Capture local feature while keeping Permutation Invariance

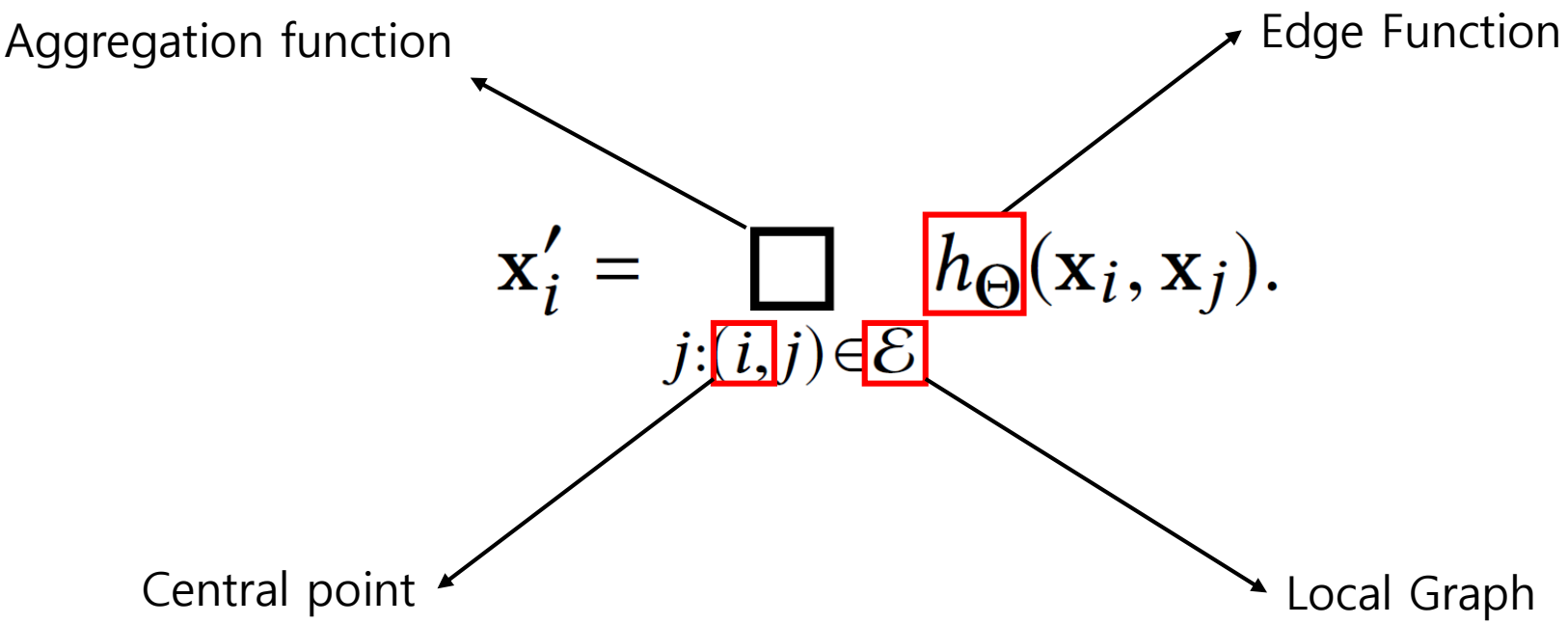
$$\mathbf{x}_i' = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j).$$



Proposed method: EdgeConv

$$\mathbf{x}_i' = \square_{j:(i,j)\in\mathcal{E}} h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j).$$

Proposed method: EdgeConv



EdgeConv : Local Graph

$$\mathbf{x}_i' = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j).$$

- *k*-nearest neighbor in feature space (k = 20)
  - Proximity in feature space differs from proximity in the input
  - This property lead non-local diffusion of information

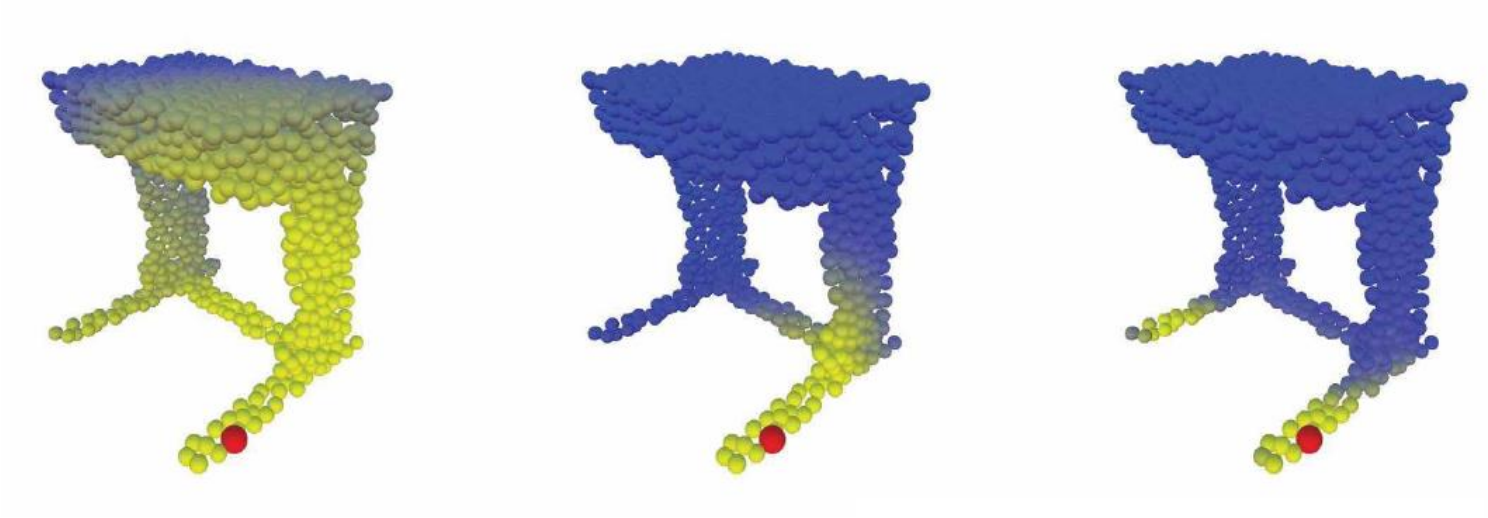


Fig. 4. **Structure of the feature spaces**

Left : Euclidian distance in the input R^3 space

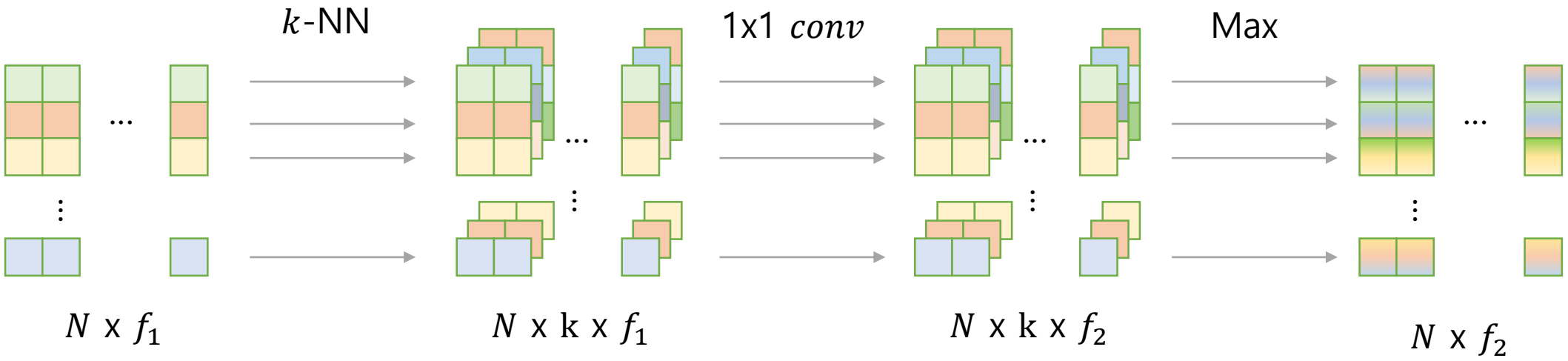
Middle : Distance after the point cloud transform stage

Right : Distance in the feature space of last layer

EdgeConv : Local Graph

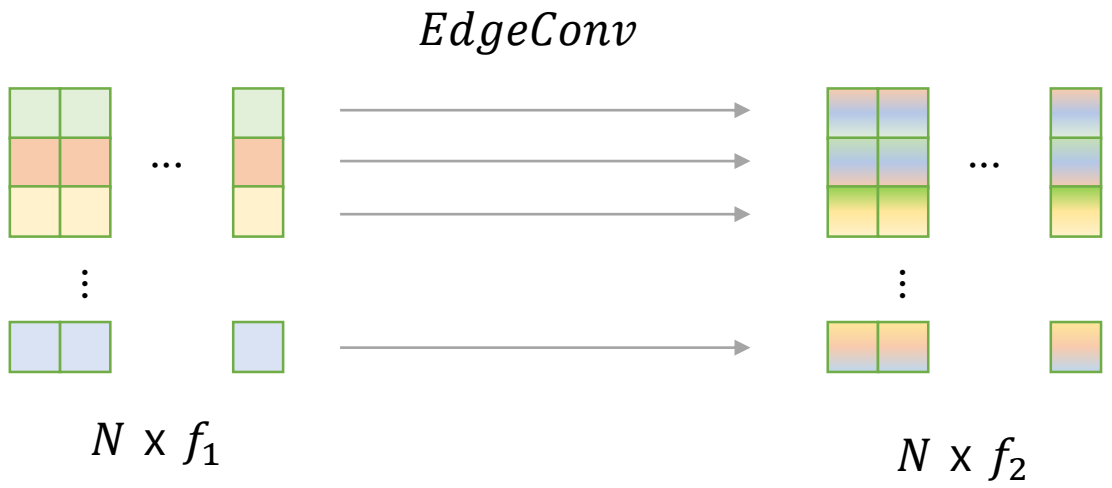
$$\mathbf{x}_i' = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j).$$

For simplicity, let's assume that $h_\Theta(x_i, x_j) = x_j$ and $\square$ as max
Note that k-NN graph in include self-loop



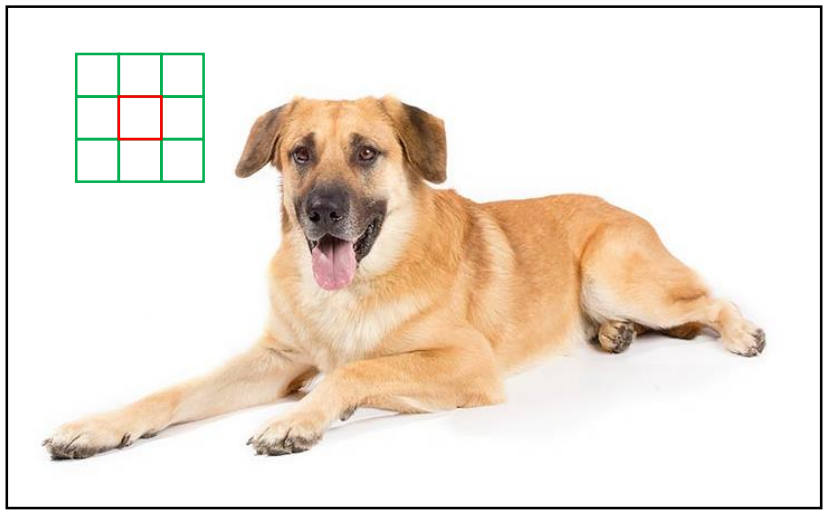
EdgeConv : Local Graph

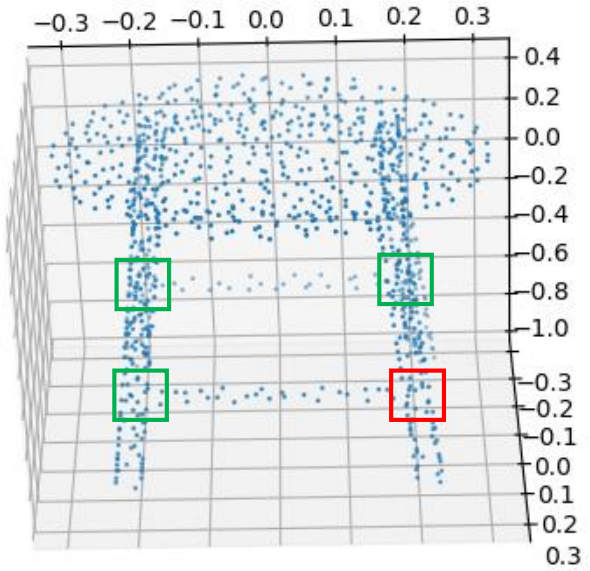
$$\mathbf{x}_i' = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j).$$



- Although input is point, output is representative of group (k-NN of input point)
- Points whose feature are similar will construct similar local graph, thus their output will also be similar. That is, EdgeConv accomplish kind of grouping.
- EdgeConv learn feature extraction to effectively group points.

EdgeConv : Local Graph





- Selected local graph varies as input, even varies along layer for same input
- EdgeConv learns how to construct the graph g used in each layer, rather than taking it as a fixed.
  -> meaning of Dynamic Graph

EdgeConv : Edge function

$$\mathbf{x}_i' = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j).$$

$h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = \bar{h}_\Theta(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i)$: combination of global and local feature

EdgeConv : Summary

- Capture local feature while keeping Permutation Invariance
  - By making local graph and aggregating them with symmetric function.
- In addition to, EdgeConv learns how to group points in a point cloud
- Proposed edge function effectively utilize local and global feature

Experimental Results

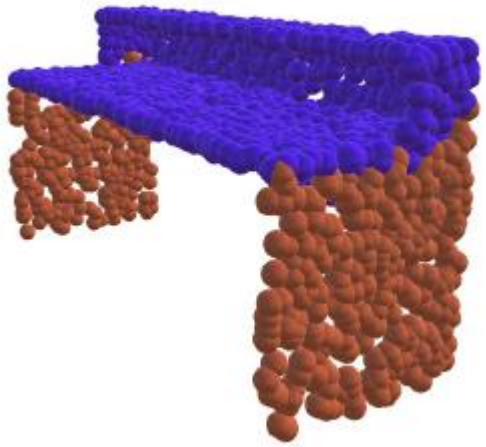
1. Classification

(dataset: *ModelNet40*)

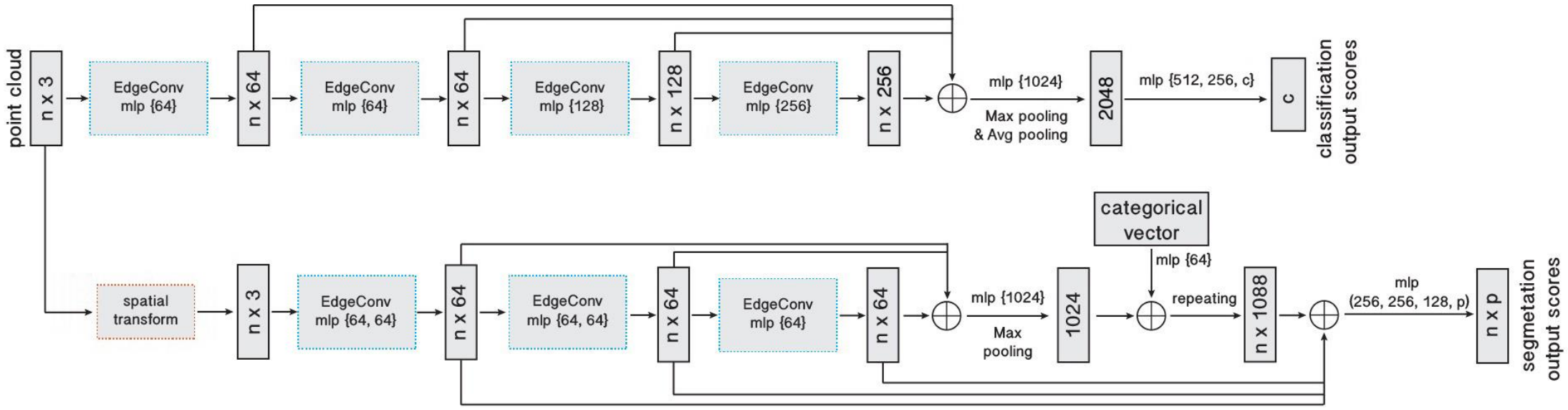
Chair?

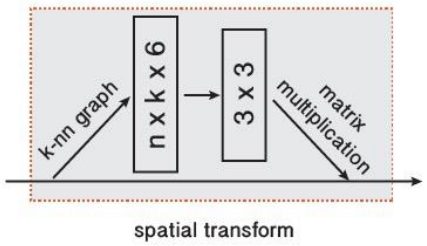
2. Part Segmentation

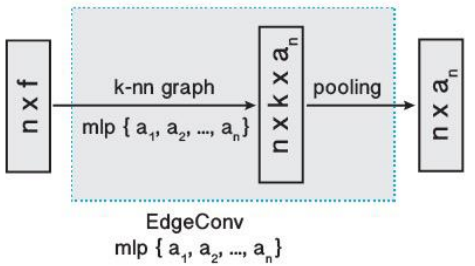
(dataset: *ShapeNet*)



Experimental Results : Used model







Experimental Results : classification

	Mean Class Accuracy	Overall Accuracy
3DShapeNets [Wu et al. 2015]	77.3	84.7
VoxNet [Maturana and Scherer 2015]	83.0	85.9
Subvolume [Qi et al. 2016]	86.0	89.2
VRN (single view) [Brock et al. 2016]	88.98	-
VRN (multiple views) [Brock et al. 2016]	91.33	-
ECC [Simonovsky and Komodakis 2017]	83.2	87.4
PointNet [Qi et al. 2017b]	86.0	89.2
PointNet++ [Qi et al. 2017c]	-	90.7
Kd-net [Klokov and Lempitsky 2017]	-	90.6
PointCNN [Li et al. 2018a]	88.1	92.2
PCNN [Atzmon et al. 2018]	-	92.3
Ours (baseline)	88.9	91.7
Ours	**90.2**	**92.9**
Ours (2048 points)	**90.7**	**93.5**

Table 2. Classification results on ModelNet40.

Experimental Results : classification

	Model size(MB)	Time(ms)	Accuracy(%)
PointNet (Baseline) [Qi et al. 2017b]	9.4	6.8	87.1
PointNet [Qi et al. 2017b]	40	16.6	89.2
PointNet++ [Qi et al. 2017c]	12	163.2	90.7
PCNN [Atzmon et al. 2018]	94	117.0	92.3
Ours (Baseline)	11	19.7	91.7
Ours	21	27.2	92.9

Table 3. Complexity, forward time, and accuracy of different models

Experimental Results : classification

CENT	DYN	MPOINTS	MEAN CLASS ACCURACY(%)	OVERALL ACCURACY(%)
			88.9	91.7
x			89.3	92.2
x	x		90.2	92.9
x	x	x	90.7	93.5

Table 4. Effectiveness of different components. CENT denotes centralization, DYN denotes dynamical graph recomputation, and MPOINTS denotes experiments with 2048 points

Baseline : $h_\Theta(x_i, x_j) = concat(x_i, x_j)$

Cent : $h_\Theta(x_i, x_j) = concat(x_i, x_i - x_j)$

Dyn : k-NN in feature domain

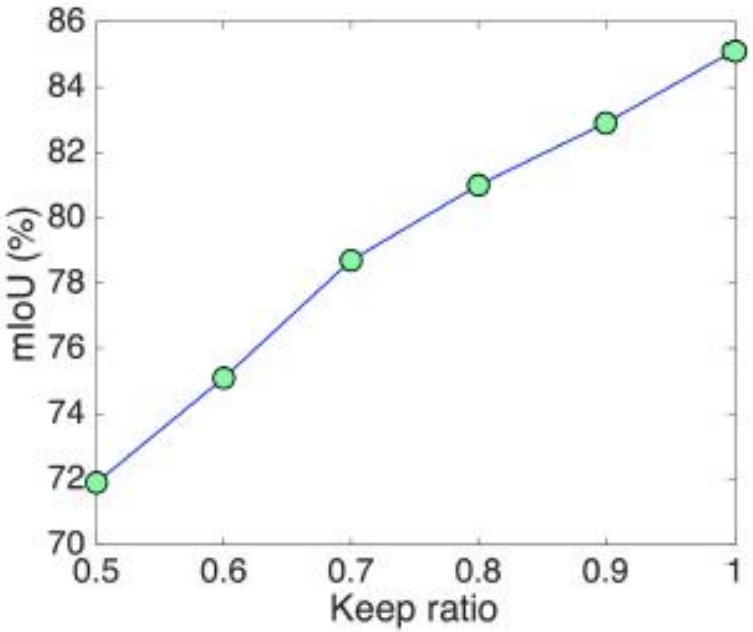
MPoints : Use 1024 -> 2048 points

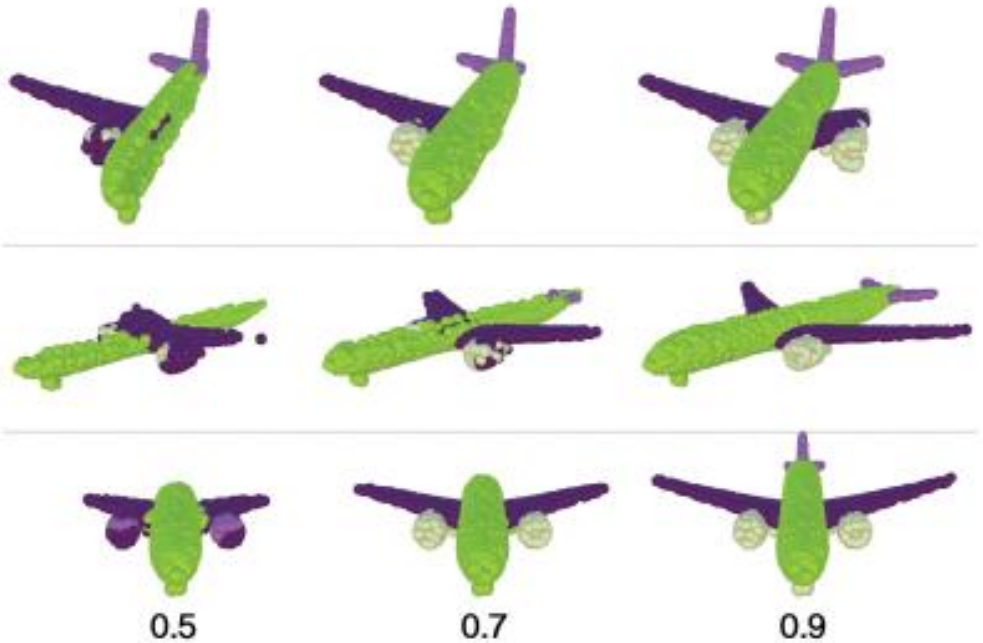
Experimental Results : Part Segmentation

	MEAN	AREO	BAG.	CAP	CAR	CHAIR	EAR PHONE	GUITAR	KNIFE	LAMP	LAPTOP	MOTOR	MUG	PISTOL	ROCKET	SKATE BOARD	TABLE
# SHAPES		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
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
KD-NET	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
LOCALFEATURENET	84.3	**86.1**	73.0	54.9	77.4	88.8	55.0	90.6	86.5	75.2	**96.1**	57.3	91.7	83.1	53.9	72.5	**83.8**
PCNN	85.1	82.4	80.1	85.5	79.5	**90.8**	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
POINTCNN	**86.1**	84.1	**86.45**	86.0	**80.8**	90.6	**79.7**	**92.3**	**88.4**	**85.3**	**96.1**	**77.2**	**95.3**	**84.2**	**64.2**	**80.0**	83.0
OURS	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6

Table 6. Part segmentation results on ShapeNet part dataset. Metric is mIoU(%) on points.

Experimental Results : Part Segmentation





Experimental Results : Indoor Scene Segmentation

	Mean IoU	overall accuracy
PointNet (baseline) [Qi et al. 2017b]	20.1	53.2
PointNet [Qi et al. 2017b]	47.6	78.5
MS + CU(2) [Engelmann et al. 2017]	47.8	79.2
G + RCU [Engelmann et al. 2017]	49.7	81.1
PointCNN [Li et al. 2018a]	**65.39**	-
Ours	56.1	**84.1**

Table 7. 3D semantic segmentation results on S3DIS. MS+CU for multi-scale block features with consolidation units; G+RCU for the grid-blocks with recurrent consolidation Units.

Conclusion

Contribution

- This paper suggest EdgeConv, which can capture local feature with permutation invariance
- In addition to, EdgeConv learns how to group similar feature by using dynamic graph
- EdgeConv shows SOTA results on classification tasks

Limitation

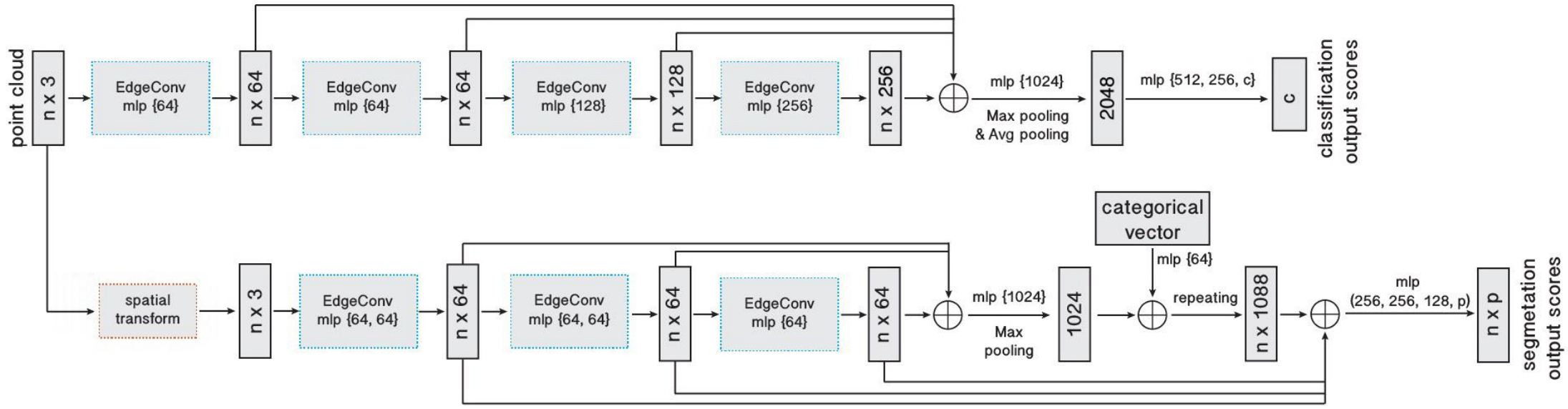
- Although they show great results on classification task, their results on Part Segmentation was not that impressive
- To claim robustness on partial data, they should show result of other method for clear comparison

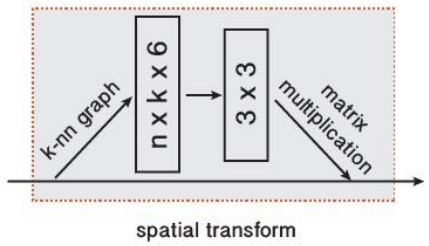
Code implementation and actual experimental results

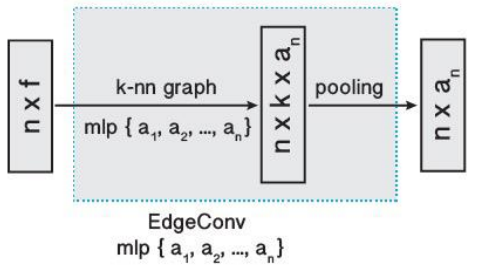
Code Links

- Official code exists!
  - https://github.com/WangYueFt/dgcnn
  - Only include classification part
- Revised version of AnTao
  - https://github.com/AnTao97/dgcnn.pytorch
  - Include classification, part segmentation.
- **My revised version**
  - https://github.com/JaehaKim97/dgcnn.pytorch
  - Add visualization function and comments, fix some errors during downloading data.

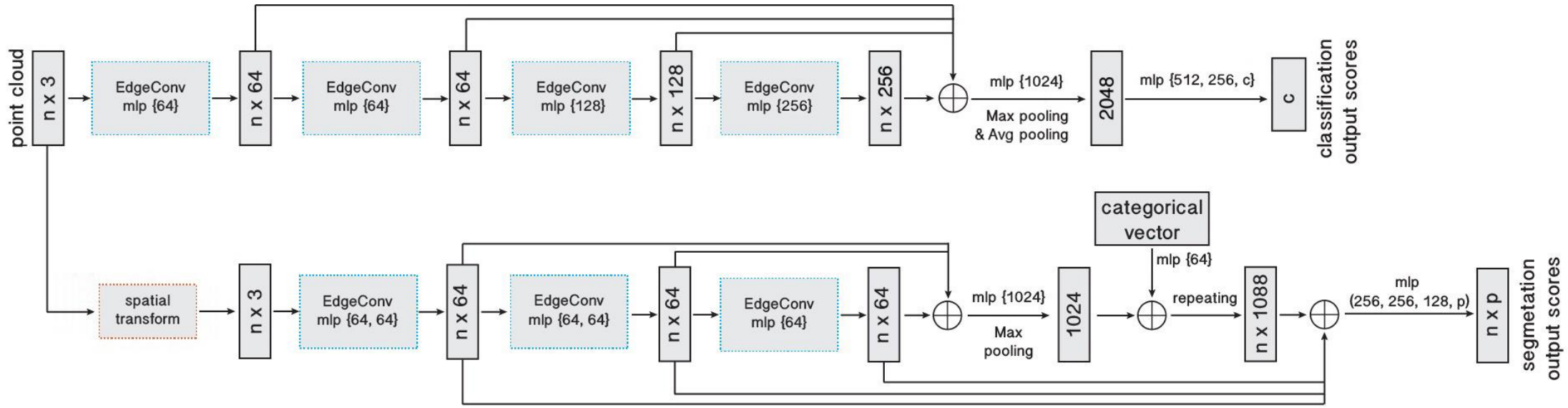
Full model with EdgeConv







Full model with EdgeConv

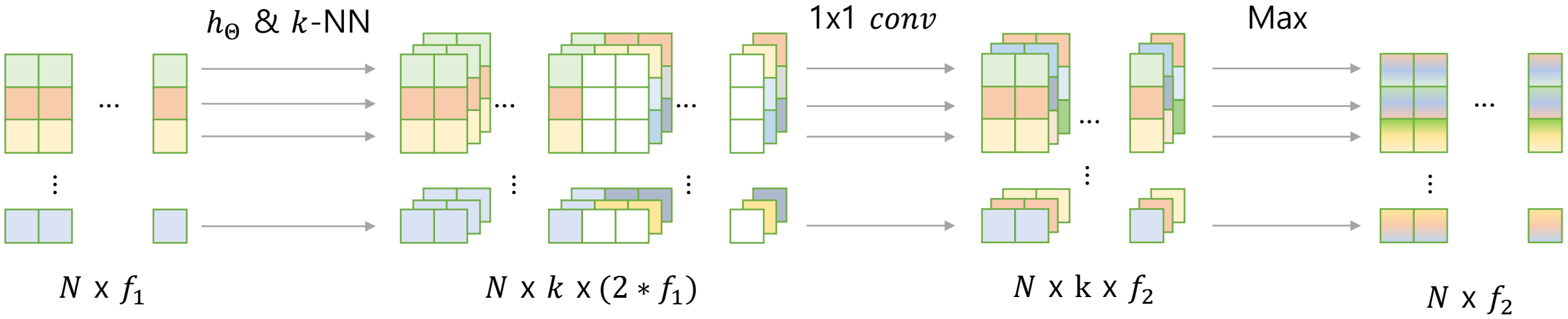


- Actually, figure is slightly different from original paper.
- However, this figure is correct according to authors' actual implementation in official code.
- Upper path for classification, lower path for part segmentation

EdgeConv : Actual Process

$$\mathbf{x}'_i = \max_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j) \quad Where \quad h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = \bar{h}_\Theta(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i)$$

concatenate



- Here is actual process of EdgeConv.

Code Review

- Data downloading process
  - Fixed by me
  - Original code produces error caused by incorrected filename.

```
def download_shapenetpart():
    BASE_DIR = os.path.dirname(os.path.abspath(__file__))
    DATA_DIR = os.path.join(BASE_DIR, 'data')
    if not os.path.exists(DATA_DIR):
        os.mkdir(DATA_DIR)
    if not os.path.exists(os.path.join(DATA_DIR, 'shapenet_part_seg_hdf5_data')):
        www = 'https://shapenet.cs.stanford.edu/media/shapenet_part_seg_hdf5_data.zip'
        zipfile = os.path.basename(www)
        os.system('wget %s; unzip %s' % (www, zipfile))
        #os.system('mv %s %s' % (zipfile[:-4], os.path.join(DATA_DIR, 'shapenet_part_seg_hdf5_data')))
        os.system('mv %s %s' % ('hdf5_data', os.path.join(DATA_DIR, 'shapenet_part_seg_hdf5_data')))
        os.system('rm %s' % (zipfile))
```

Code Review

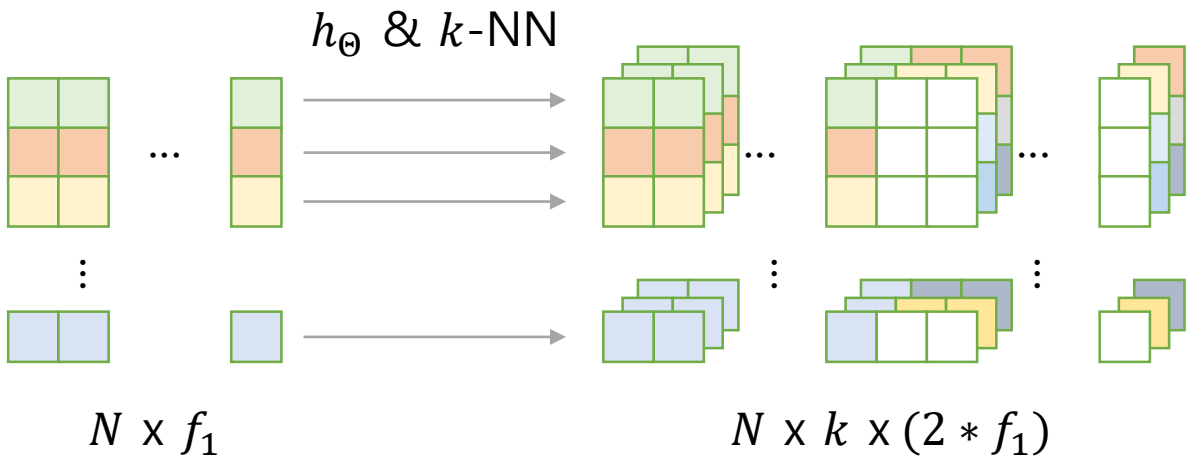
- Calculating k-NN points
  - Commented by me

```
##############################################################
# input
# x : current feature map
# k : how many points for k nearest neighborhood
# output
# idx : index of feature map x, which belongs to k-NN of x
##############################################################
def knn(x, k):
    inner = -2*torch.matmul(x.transpose(2, 1), x)
    xx = torch.sum(x**2, dim=1, keepdim=True)
    pairwise_distance = -xx - inner - xx.transpose(2, 1) # calculate euclidian distance

    idx = pairwise_distance.topk(k=k, dim=-1)[1]   # (batch_size, num_points, k)
    return idx
```

Code Review

- Grouping k-NN points
  - Concatenation with edge function
  - Commented by me




```
############################################################
# input
# x : current feature map
# output
# feature : grouping with k-NN, and concatenation with ed-
#           ge function ((xj-xi), xi). i.e. centralization
############################################################
def get_graph_feature(x, k=20, idx=None, dim9=False):
    batch_size = x.size(0)
    num_points = x.size(2)
    x = x.view(batch_size, -1, num_points)
    if idx is None:
        if dim9 == False:
            idx = knn(x, k=k)   # (batch_size, num_points, k)
        else:
            idx = knn(x[:, 6:], k=k)
    device = torch.device('cuda')

    idx_base = torch.arange(0, batch_size, device=device).view(-1, 1, 1)*num_points

    idx = idx + idx_base

    idx = idx.view(-1)

    _, num_dims, _ = x.size()

    x = x.transpose(2, 1).contiguous()   # (batch_size, num_points, num_dims)  -> (b
    feature = x.view(batch_size*num_points, -1)[idx, :]
    feature = feature.view(batch_size, num_points, k, num_dims)
    x = x.view(batch_size, num_points, 1, num_dims).repeat(1, 1, k, 1)

    feature = torch.cat((feature-x, x), dim=3).permute(0, 3, 1, 2)

    return feature      # (batch_size, 2*num_dims, num_points, k)
```

Code Review

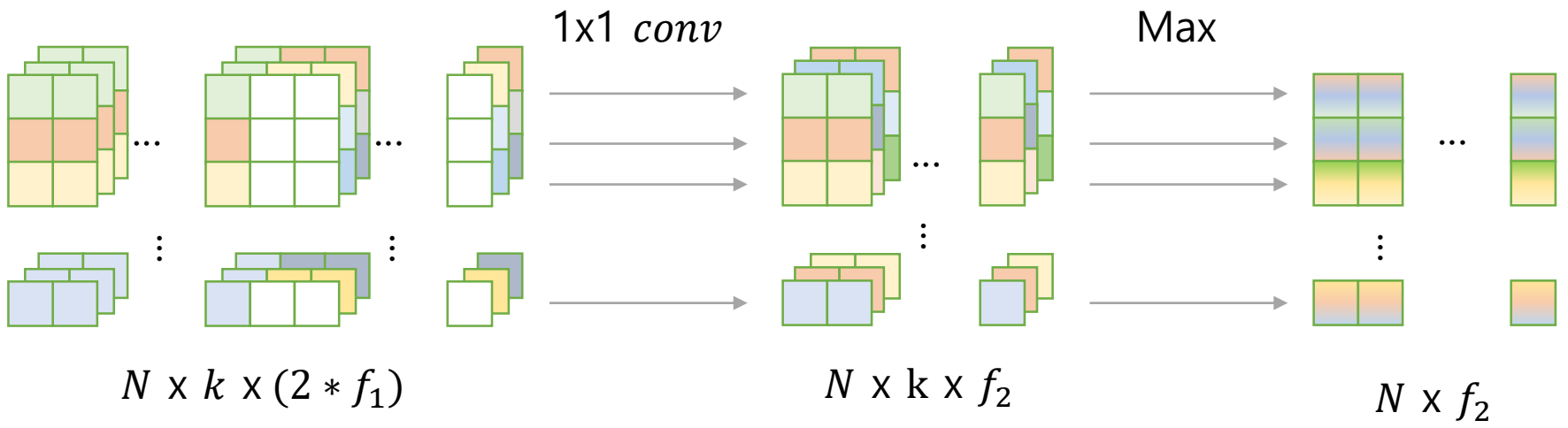
- Actual working process of EdgeConv
  - Operate 1x1 conv
  - Apply MaxPooling

conv

```
self.conv1 = nn.Sequential(nn.Conv2d(6, 64, kernel_size=1, bias=False),
                           self.bn1,
                           nn.LeakyReLU(negative_slope=0.2))
```

forwarding

```
x = self.conv1(x)
x1 = x.max(dim=-1, keepdim=False)[0]
```



Code Review

- Total process (classification)
  - Dropout rate : 0.5
  - Commented by me

```python
# Total process of classfication model
def forward(self, x):
    batch_size = x.size(0)
    x = get_graph_feature(x, k=self.k)      # (batch_size, 3, num_points) -> (batch_size, 3*2, num_points, k)
    x = self.conv1(x)                       # (batch_size, 3*2, num_points, k) -> (batch_size, 64, num_points, k)
    x1 = x.max(dim=-1, keepdim=False)[0]    # (batch_size, 64, num_points, k) -> (batch_size, 64, num_points)

    x = get_graph_feature(x1, k=self.k)     # (batch_size, 64, num_points) -> (batch_size, 64*2, num_points, k)
    x = self.conv2(x)                       # (batch_size, 64*2, num_points, k) -> (batch_size, 64, num_points, k)
    x2 = x.max(dim=-1, keepdim=False)[0]    # (batch_size, 64, num_points, k) -> (batch_size, 64, num_points)

    x = get_graph_feature(x2, k=self.k)     # (batch_size, 64, num_points) -> (batch_size, 64*2, num_points, k)
    x = self.conv3(x)                       # (batch_size, 64*2, num_points, k) -> (batch_size, 128, num_points, k)
    x3 = x.max(dim=-1, keepdim=False)[0]    # (batch_size, 128, num_points, k) -> (batch_size, 128, num_points)

    x = get_graph_feature(x3, k=self.k)     # (batch_size, 128, num_points) -> (batch_size, 128*2, num_points, k)
    x = self.conv4(x)                       # (batch_size, 128*2, num_points, k) -> (batch_size, 256, num_points, k)
    x4 = x.max(dim=-1, keepdim=False)[0]    # (batch_size, 256, num_points, k) -> (batch_size, 256, num_points)

    x = torch.cat((x1, x2, x3, x4), dim=1)  # (batch_size, 64+64+128+256, num_points)

    x = self.conv5(x)                       # (batch_size, 64+64+128+256, num_points) -> (batch_size, emb_dims, num_points)
    x1 = F.adaptive_max_pool1d(x, 1).view(batch_size, -1)           # (batch_size, emb_dims, num_points) -> (batch_size, emb_dims)
    x2 = F.adaptive_avg_pool1d(x, 1).view(batch_size, -1)           # (batch_size, emb_dims, num_points) -> (batch_size, emb_dims)
    x = torch.cat((x1, x2), 1)              # (batch_size, emb_dims*2)

    x = F.leaky_relu(self.bn6(self.linear1(x)), negative_slope=0.2) # (batch_size, emb_dims*2) -> (batch_size, 512)
    x = self.dp1(x)
    x = F.leaky_relu(self.bn7(self.linear2(x)), negative_slope=0.2) # (batch_size, 512) -> (batch_size, 256)
    x = self.dp2(x)
    x = self.linear3(x)                                             # (batch_size, 256) -> (batch_size, output_channels)

    return x
```

Code Review

- Training process (classification)
- Note that we use:
  - Loss : cross entropy loss with smoothing
  - Optimizer : SGD(lr=0.1, momentum=0.9, weight_decay=1e-4)
  - Scheduler : cosine annealing

```python
for epoch in range(args.epochs):
    ####################
    # Train
    ####################
    train_loss = 0.0
    count = 0.0
    model.train()
    train_pred = []
    train_true = []
    for data, label in train_loader:
        data, label = data.to(device), label.to(device).squeeze()
        data = data.permute(0, 2, 1)
        batch_size = data.size()[0]
        opt.zero_grad()
        logits = model(data)
        loss = criterion(logits, label)
        loss.backward()
        opt.step()
        preds = logits.max(dim=1)[1]
        count += batch_size
        train_loss += loss.item() * batch_size
        train_true.append(label.cpu().numpy())
        train_pred.append(preds.detach().cpu().numpy())
    if args.scheduler == 'cos':
        scheduler.step()
    elif args.scheduler == 'step':
        if opt.param_groups[0]['lr'] > 1e-5:
            scheduler.step()
        if opt.param_groups[0]['lr'] < 1e-5:
            for param_group in opt.param_groups:
                param_group['lr'] = 1e-5

    train_true = np.concatenate(train_true)
    train_pred = np.concatenate(train_pred)
```

Code Review

- Visualizing function
  - Written by me

```
############################################################
# input
# data : input 3d points
# label : prediced(or ground-truth) label
# class_label : classification label
# idx : index of batch to visualize
############################################################
def visualize(data, label, class_label, idx):
    ax = plt.axes(projection='3d')
    data = np.array(data.cpu())

    data = data[idx]
    label = label[idx]

    label_lists = []
    color_lists = ['blue', 'red', 'green', 'yellow', 'black', 'white', 'cyan', 'magneta']

    for i in range(len(label)):
        if not (label[i] in label_lists):
            label_lists.append(label[i])

    for j in range(len(label_lists)):
        m_data = data * ( label == label_lists[j])
        ax.scatter(m_data[0]*(m_data[0] != 0), m_data[1]*(m_data[1] != 0),
                   m_data[2]*(m_data[2] != 0), s=1, color=color_lists[j])

    plt.title(class_lists[class_label[idx]])
    plt.show()
```

Executing Code

- For details, please refer https://github.com/JaehaKim97/dgcnn.pytorch

Point Cloud Classification

Run the training script:

- 1024 points

```
python main_cls.py --exp_name=cls_1024 --num_points=1024 --k=20
```

- 2048 points

```
python main_cls.py --exp_name=cls_2048 --num_points=2048 --k=40
```

Run the evaluation script after training finished:

You can visualize results with `--visualize` option. Note that whole evaluation process will stop until you close 'Figure' window of matplot. Therfore, I recommend you to not use `--visualize` option when you want to get full evaluation results.

- 1024 points

```
python main_cls.py --exp_name=cls_1024_eval --num_points=1024 --k=20 --eval=True --model_path=checkpoints/cls_1024/model
```

- 2048 points

```
python main_cls.py --exp_name=cls_2048_eval --num_points=2048 --k=40 --eval=True --model_path=checkpoints/cls_2048/model
```

Miscellaneous

- Total training time for classification : about **1.5 hours** (with 4 GPUs, RTX 2080TI)
- Total training time for part segmentation : about **7 hours** (with 4 GPUs, RTX 2080TI)

- For classification, 1 GPU is enough for training.
- For part segmentation, at least 4 GPUs are mandatory to not make memory issue.

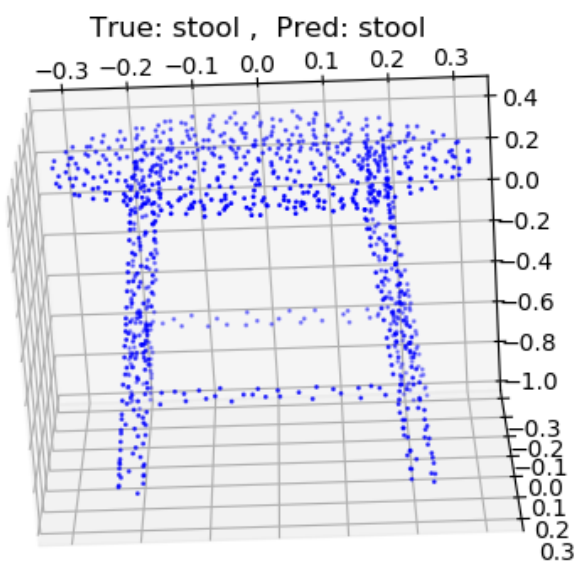
Actual Experimental Results - classification

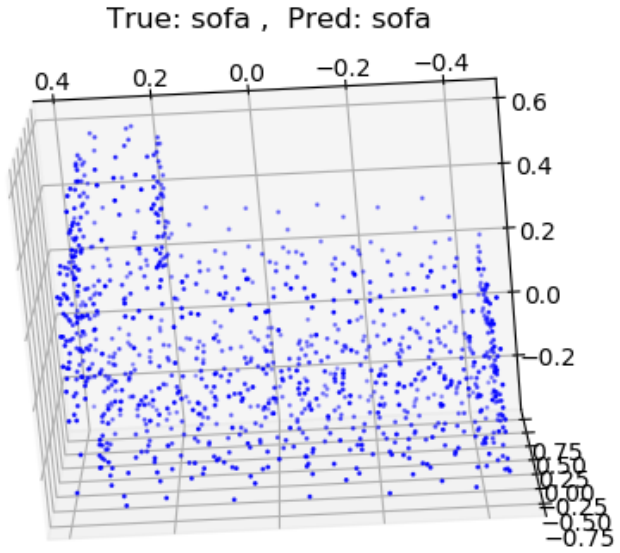
- Classification on ModelNet40

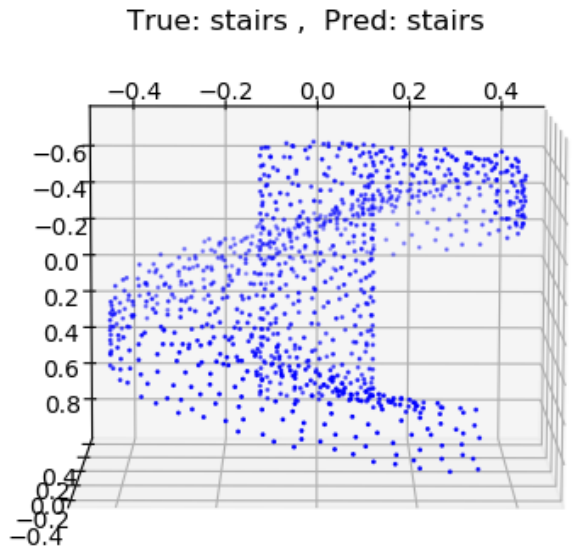
	Mean Class Acc	Overall Acc
Paper (1024 points)	90.2	92.9
My Results (1024 points)	**89.8**	**92.6**

Actual Experimental Results - classification

- Visualization results







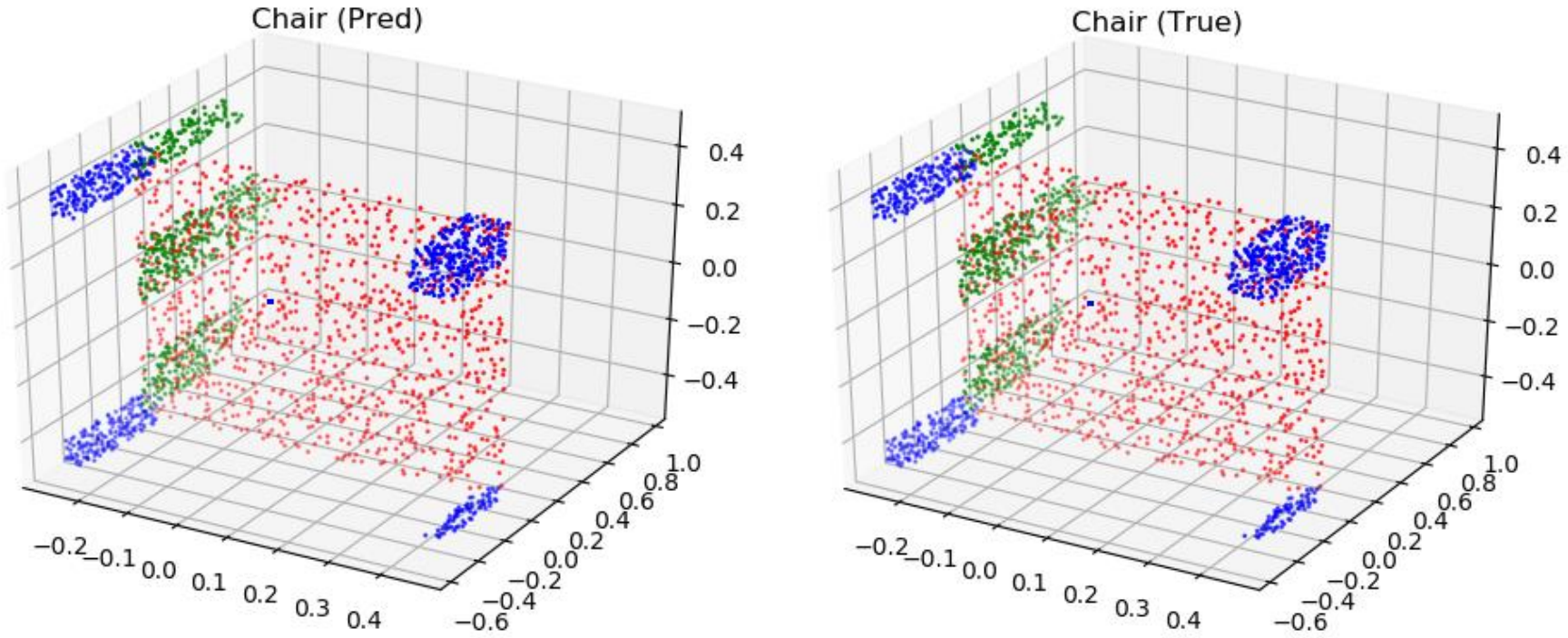
Actual Experimental Results – Part Segmentation

- Part Segmentation on ShapeNet

	Mean IoU
Paper	85.2
My Results	**85.1**

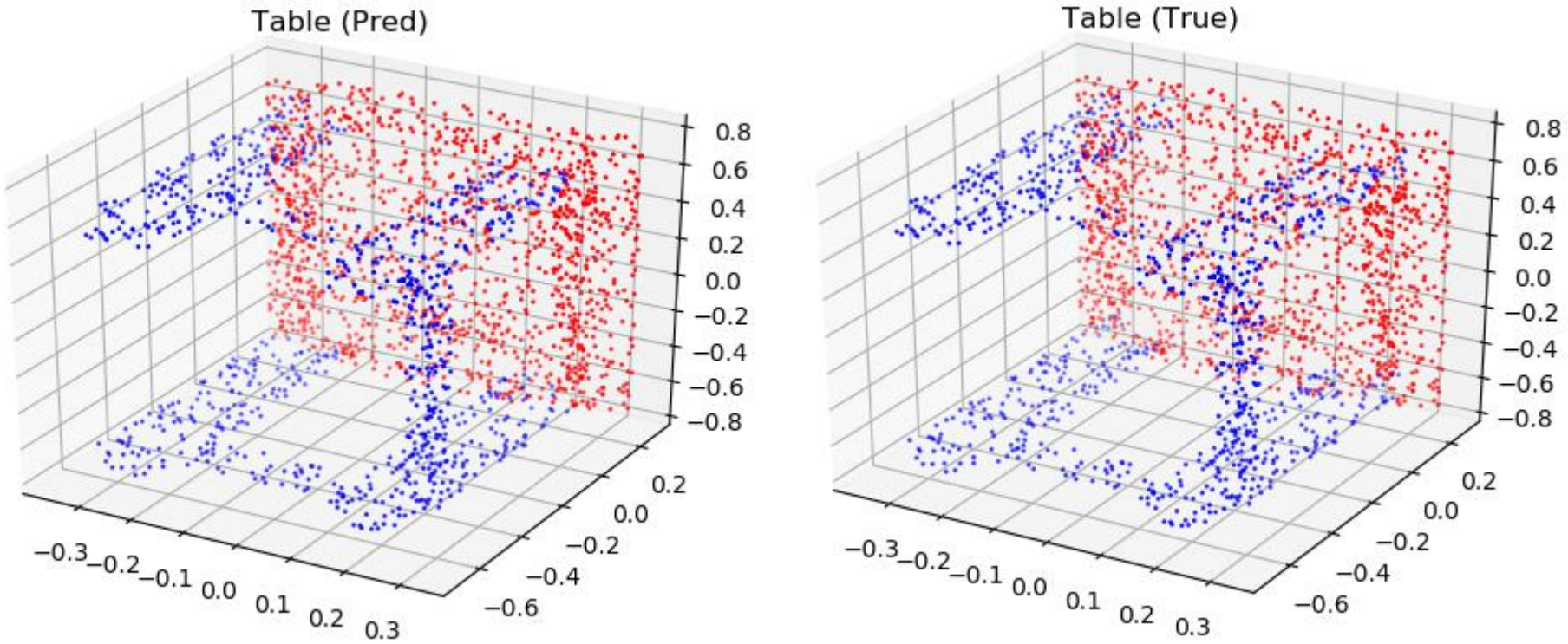
Actual Experimental Results – Part Segmentation

- Visualization results



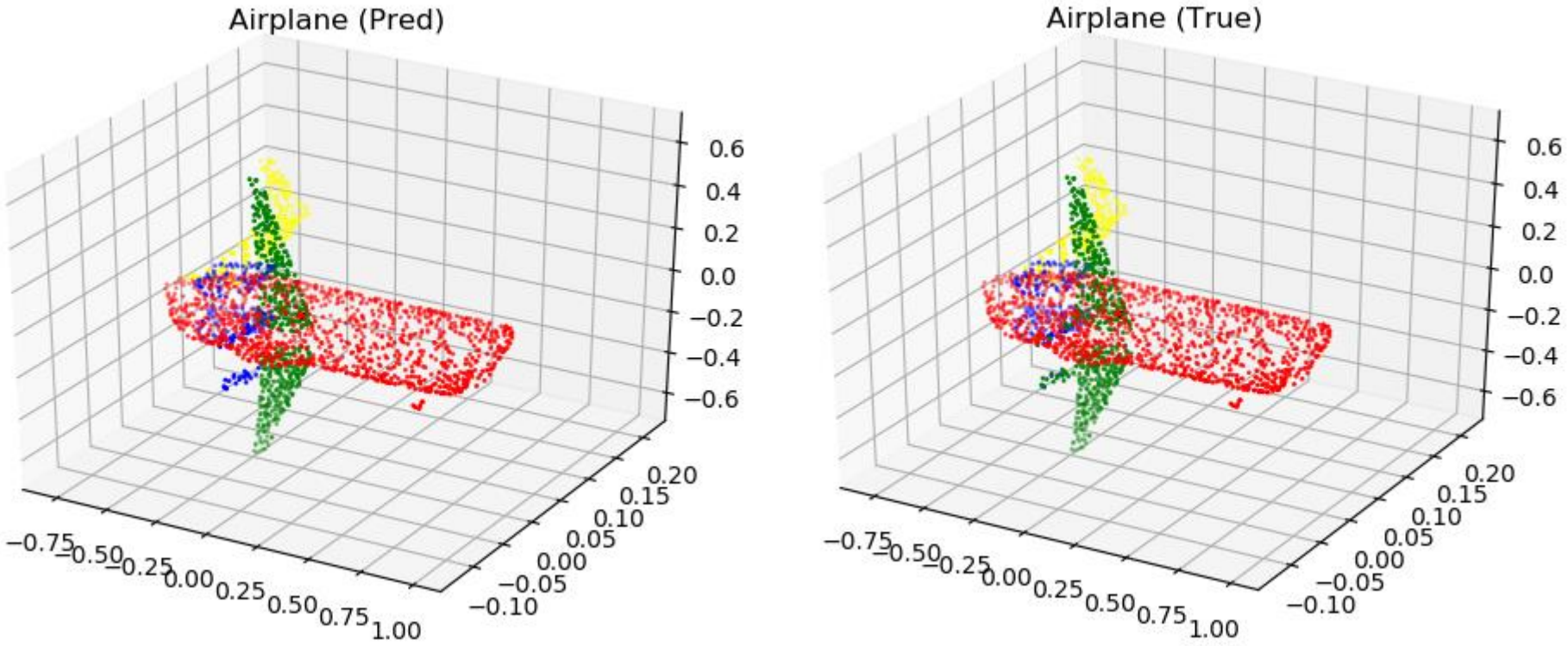
Actual Experimental Results – Part Segmentation

- Visualization results



Actual Experimental Results – Part Segmentation

- Visualization results



Thank You !

If you have any questions, please contact me

jhkim97s2@gmail.com