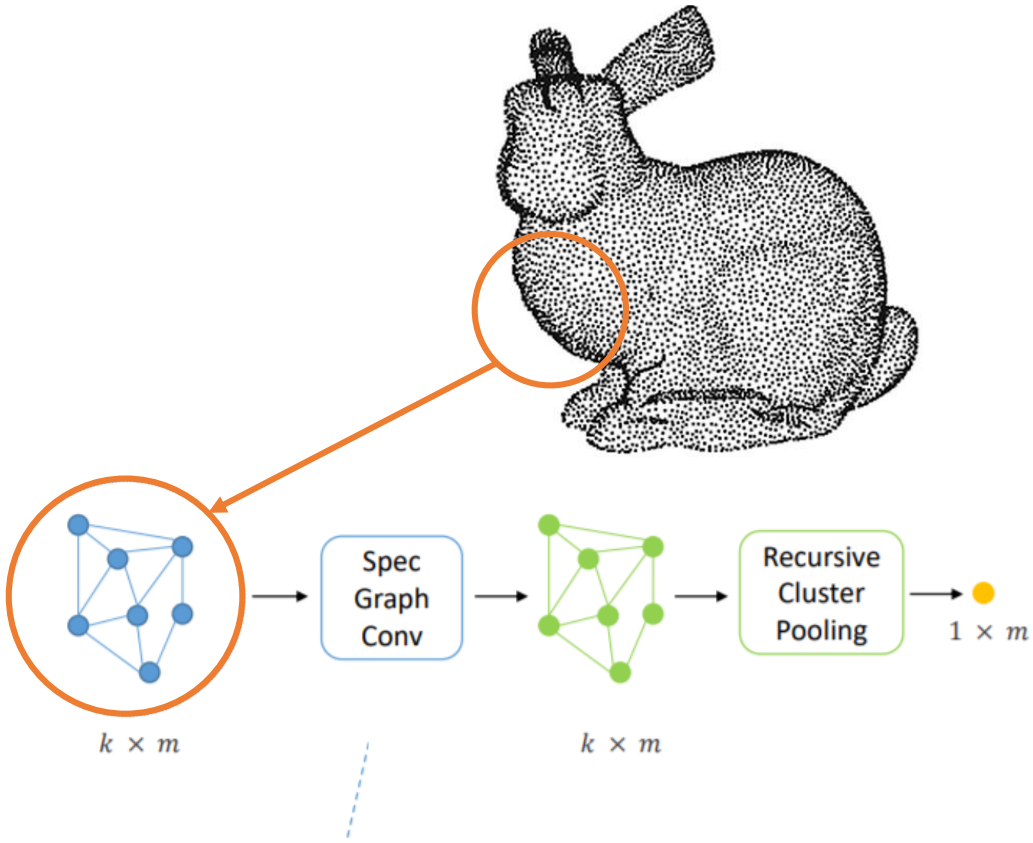
Local Spectral Graph Convolution for Point Set Feature Learning (ECCV 2018)

Chu Wang, Babak Samari, Kealeem Siddiqi



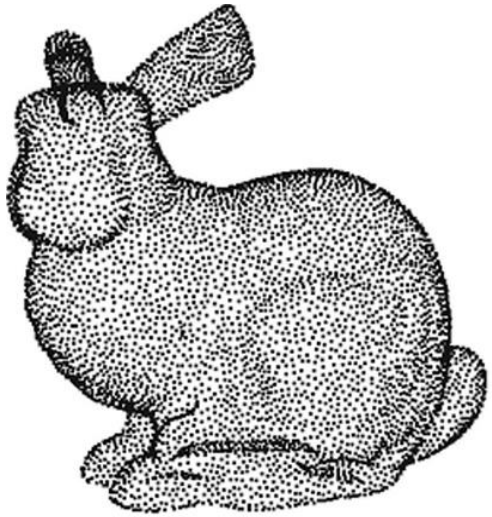
Mijeong Kim

Seoul National University

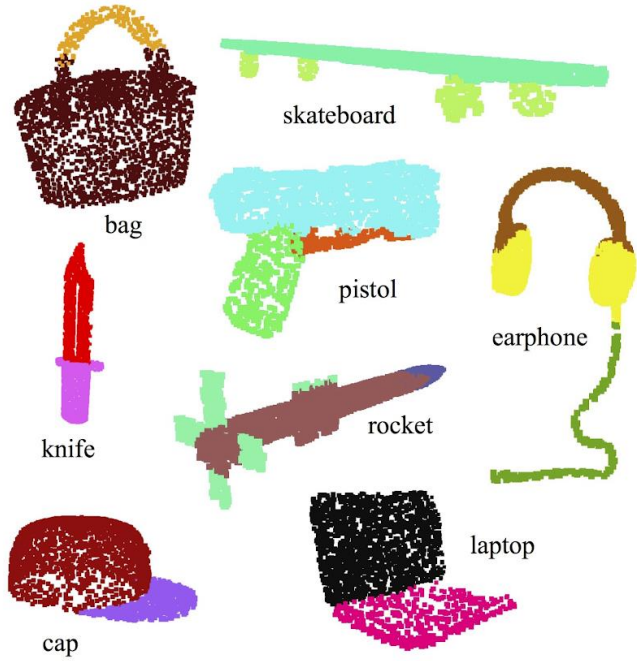
Topic

- Data : Point Cloud
  - Representing 3D model via N points(x, y, z coordinates).
- Task : Feature learning of point set for classification & part segmentation

3D bunny model



Point Cloud representation



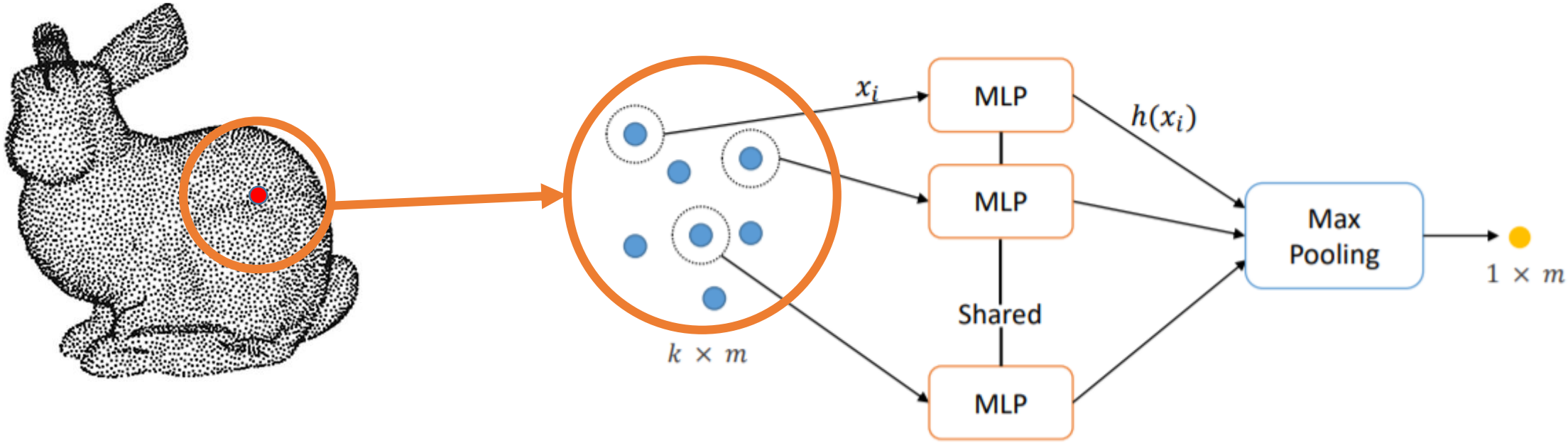
Classification & Part segmentation

Previous Work

- PointNet++
  - Using k nearest neighbor points of each point to learn feature of that point.
  - Limitation

    → MLP: each point features are obtained by an independent and isolated manner.

    → Max Pooling: can loose significant information



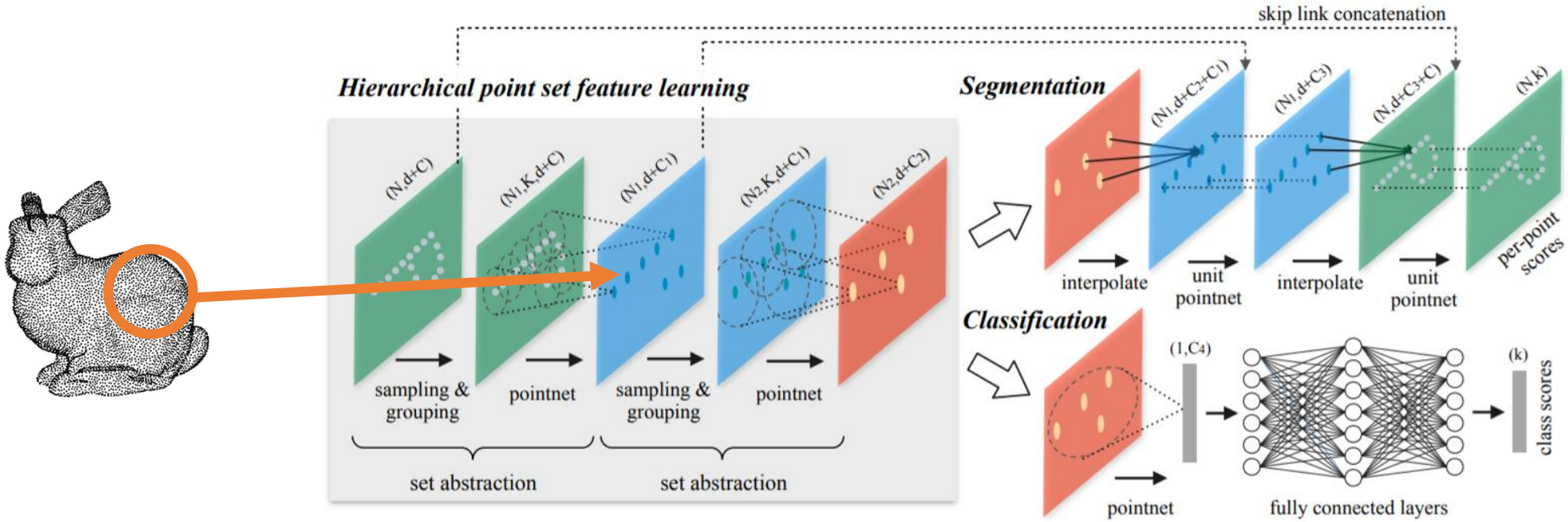
Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. NIPS (2017)

Previous Work

- PointNet++
  - Using k nearest neighbor points of each point to learn feature of that point.
  - Limitation

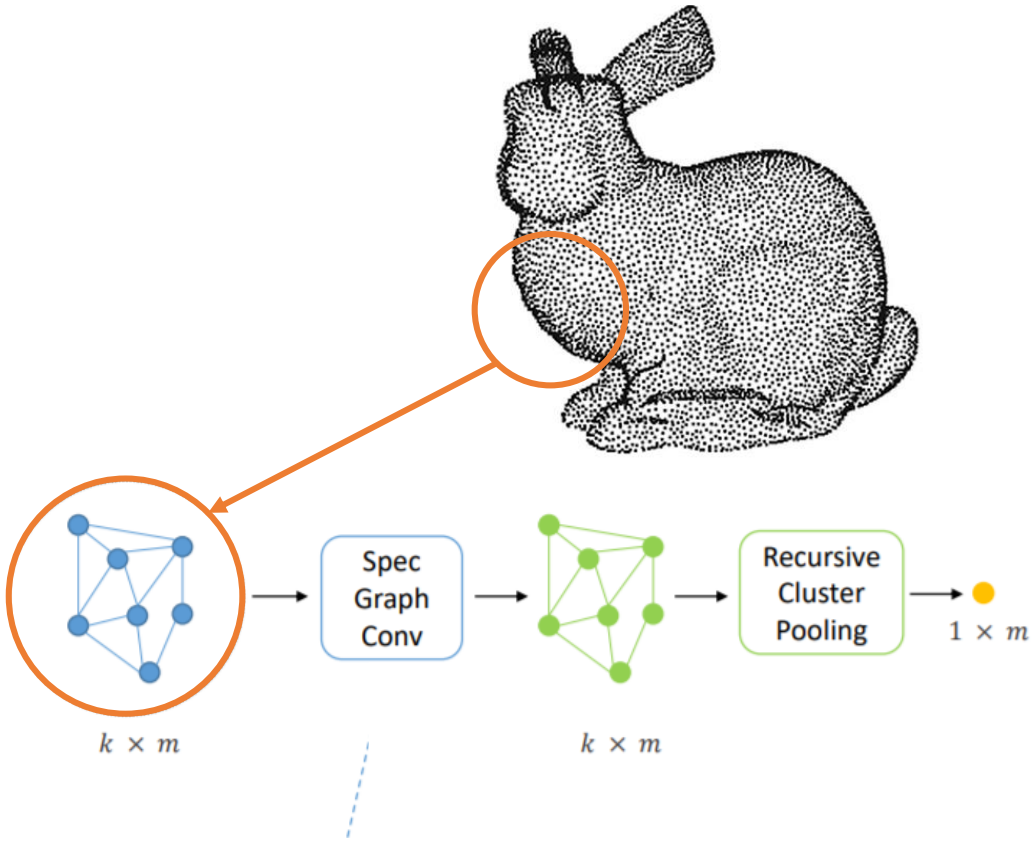
    → MLP: each point features are obtained by an independent and isolated manner.

    → Max Pooling: can loose significant information



Local Spectral Graph Convolution for Point Set Feature Learning (ECCV 2018)

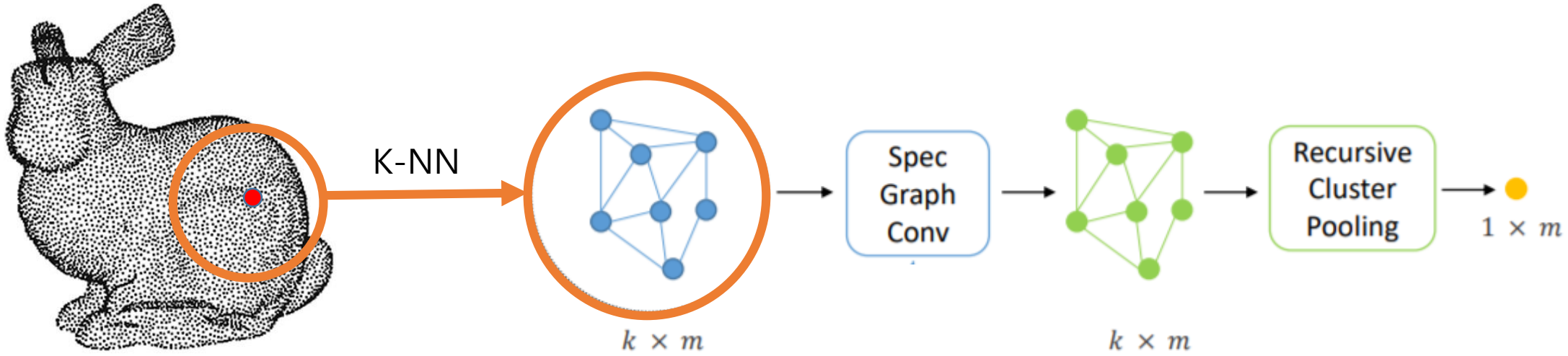
Chu Wang, Babak Samari, Kealeem Siddiqi



Local Spectral GCN

- Local k-NN graph
  - Each local graph is formulated with pair-wise distance
  - Jointly learn features using local structural information via k-NN local graph
  - Laplacian of each local graph can be computed in learning (end-to-end)

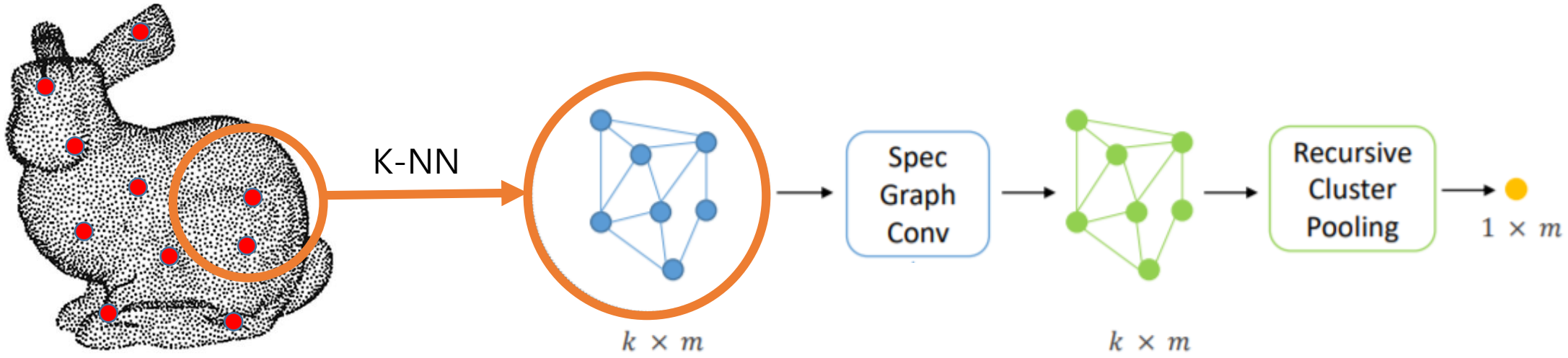
    → In previous work(ChebNet, Simplified ChebNet), graph laplacian should be precomputed before training or testing because of large graph size



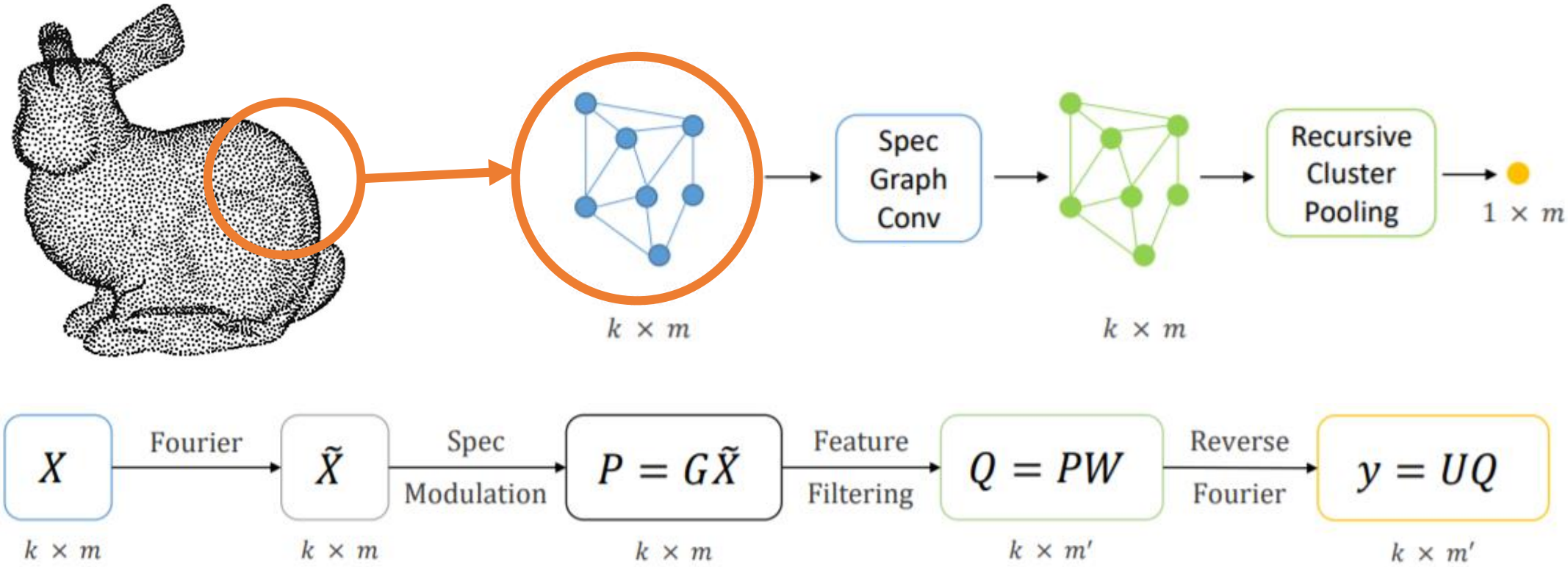
Local Spectral GCN

- Local k-NN graph
  - Each local graph is formulated with pair-wise distance
  - Jointly learn features using local structural information via k-NN local graph
  - Laplacian of each local graph can be computed in learning (end-to-end)

    → In previous work(ChebNet, Simplified ChebNet), graph laplacian should be precomputed before training or testing because of large graph size

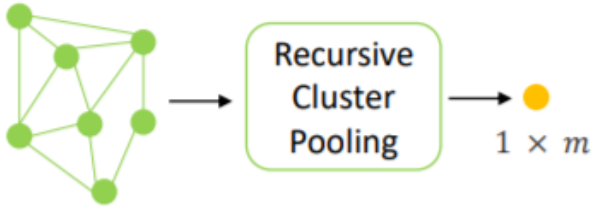


Local Spectral GCN

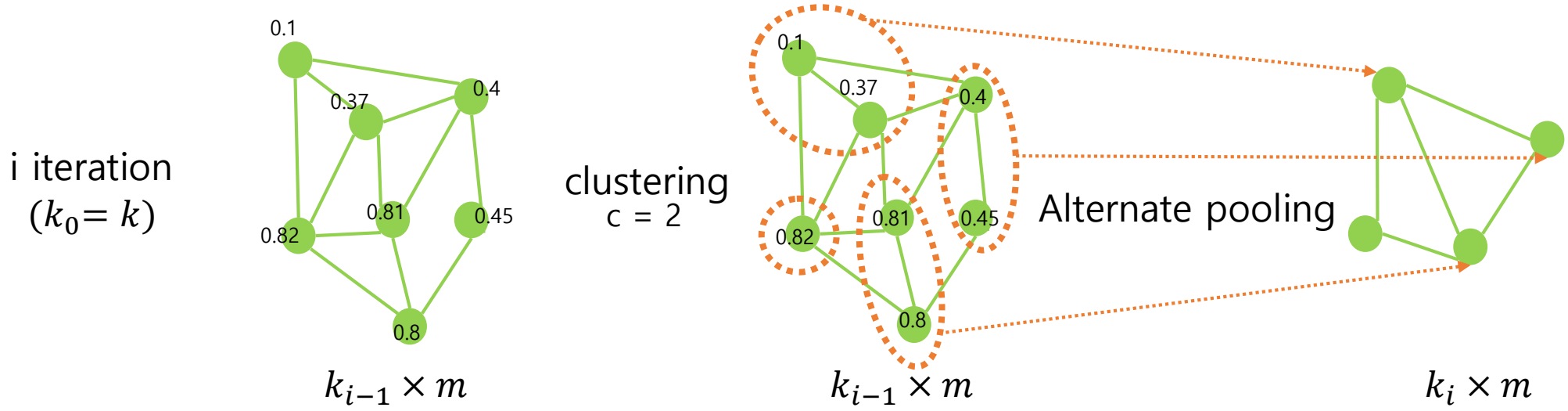


- Spectral modulation matrix, G: modulate the frequency components
- The feature filter W: change the dimension of each node
- These parameters are shared by all the k-NN local graph in same layer.

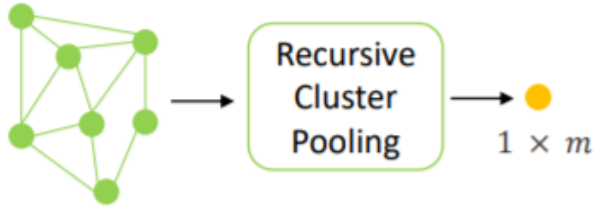
Recursive spectral clustering pooling



- Recursive spectral clustering pooling
  - Using the Fiedler vector similar to normalized cut,
    cut the local graph into k_i clusters whose size should be pre-defined c
  - Pooling each clusters to one node with max pooling or average pooling
  - Then the number of nodes became k_i. And iterating clustering, until the number of nodes smaller than pre-defined c. (i = 1, 2, …..)

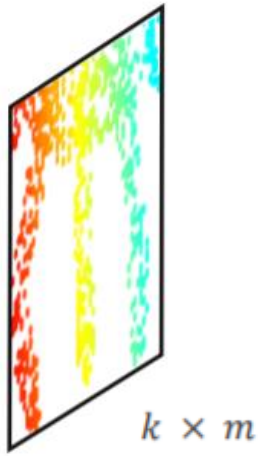


Recursive spectral clustering pooling

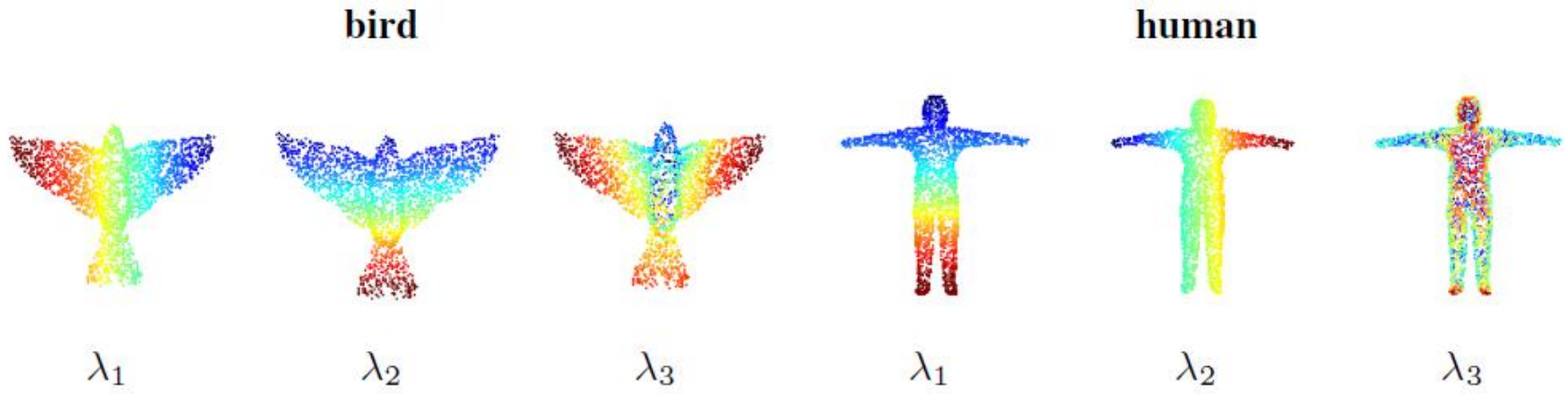


$k \times m$

- Limitation of Max pooling
  - Can not preserve information from disjoint sets of points within in k-NN
- Recursive spectral clustering pooling
  - Capture fine local structures in local neighborhood
  - Aggregating features in local k-NN graph by recursively clustering
  - Pooling only within a cluster of similar abstract point features

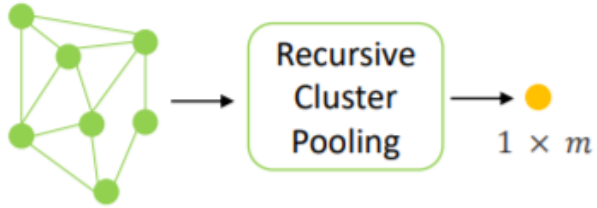


All points in a k-nn graph

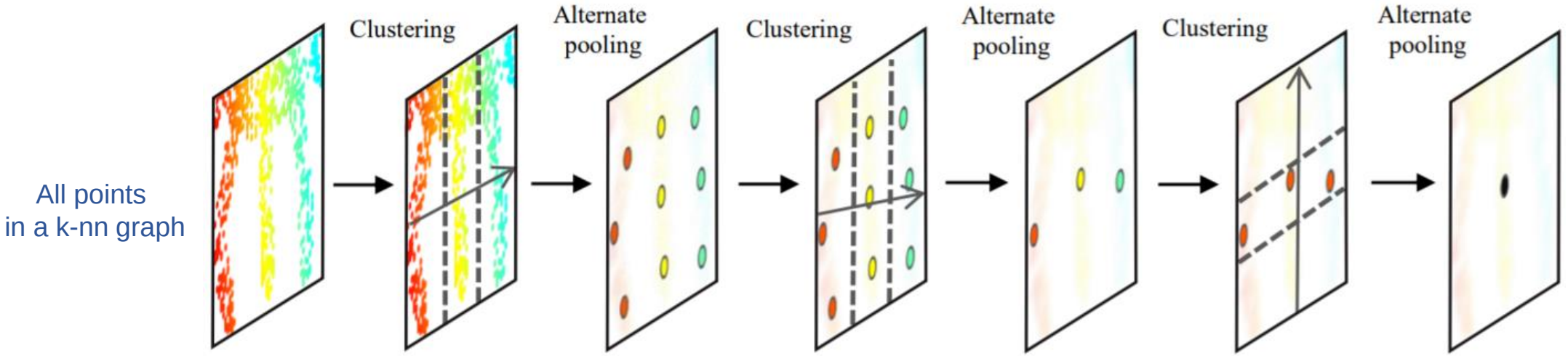


Visualization of spectral coordinates for a bird and a human

Recursive spectral clustering pooling

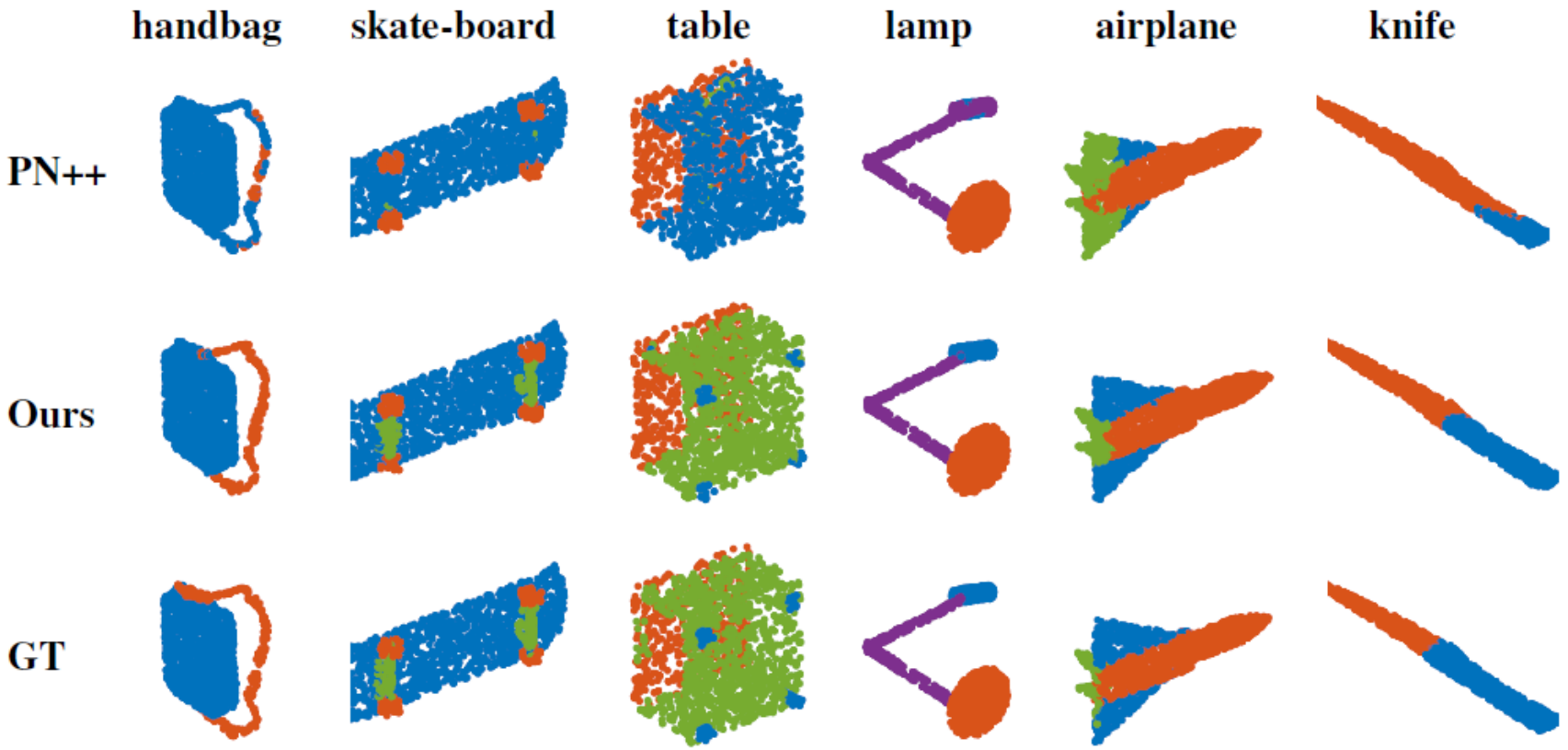


- Limitation of Max pooling
  - Can not preserve information from disjoint sets of points within in k-NN
- Recursive spectral clustering pooling
  - Capture fine local structures in local neighborhood
  - Aggregating features in local k-NN graph by recursively clustering
  - Pooling only within a cluster of similar abstract point features



Results

- Part segmentation on ShapeNet



Improvement

- Captures fine local structures
- successfully segment connected parts of a 3D object

Results

- Classification

	McGill Shape Benchmark (instance acc, %)	McGill Shape Benchmark (category cc, %)	MNIST dataset -2d pointcloud (error rate, %)	ModleNet40 (acc, %)
Others	-	-	0.81[1]	-
PointNet++	93.06	93.27	0.55	91.9
Spectral GCN	95.83	95.74	0.42	92.1

- Part Segmentation

	ShapeNet (mIOU, %)	ScanNet (Acc, %)
Others	84.70[2]	73.90[3]
PointNet++	84.90	84.00
Spectral GCN	85.40	84.80

[1] Monti, F., Boscaini, D., Masci, J., Rodol`a, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. CVPR 2017 (2016)
[2] Yi, L., Su, H., Guo, X., Guibas, L.: Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In: Computer Vision and Pattern Recognition (CVPR) (2017)
[3] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593 (2016)

Ablation study

- Classification

	McGill Shape Benchmark (instance acc, %)	McGill Shape Benchmark (category cc, %)	MNIST dataset -2d pointcloud (error rate, %)	ModleNet40 (acc, %)
Others	-	-	0.81[1]	-
PointNet++ (MLP + max pooling)	93.06	93.27	0.55	91.5
Spectral GCN + max pooling	95.14	95.43	-	91.9
Spectral GCN + recursive pooling	95.83	95.74	0.42	92.1

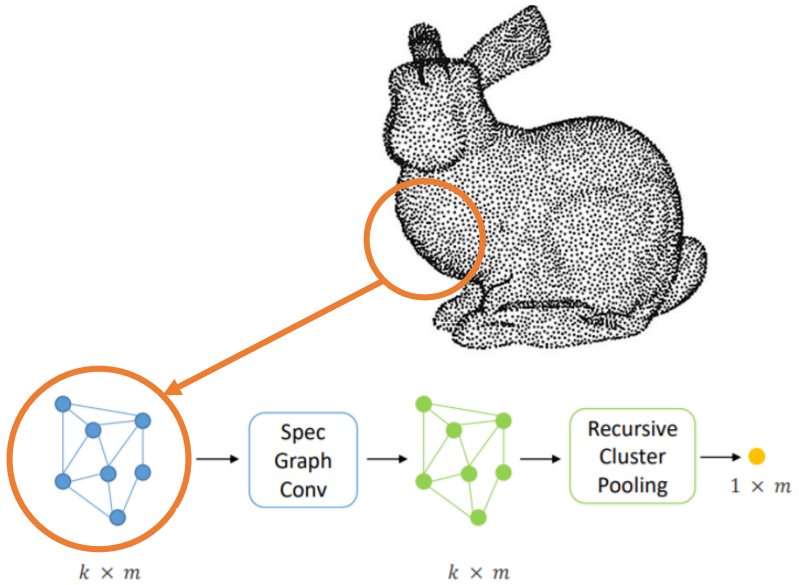
- Part Segmentation
  - There is no ablation study on part segmentation

Summary

- Local spectral GCN
  - Jointly learn features with graph convolution in spectral domain
  - Local graph: End to end learning
- Recursive spectral clustering pooling
  - Aggregate information form within clusters of nodes that are closed to one another in their spectral coordinates
  - Captures fine local structures within a cluster

- Limitation
  - There is a permutation problem on fully connected layer after hierarchical feature learning.
  - The size of a cluster is fixed in recursive pooling and this can leads wrong clustering.
  - Local spectral GCN > Recursive spectral clustering pooling

Code Reproduce and Experiments



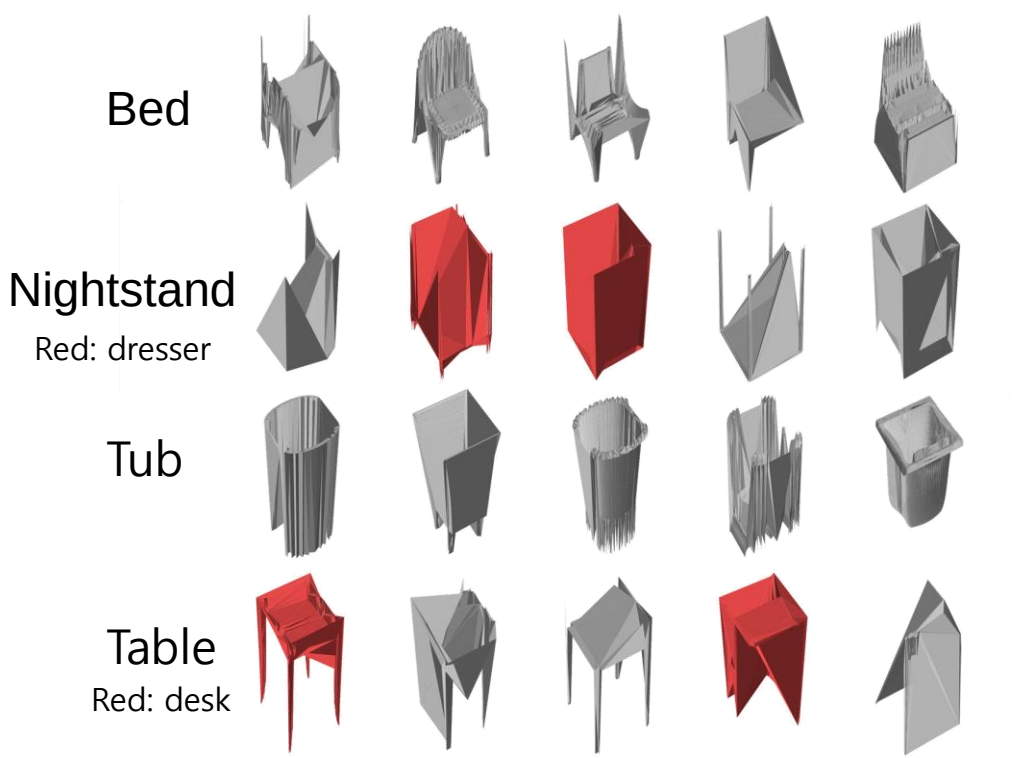
Code Reproduce – Experiment Analysis

- Classification on ModelNet 40
  - Baseline
    - Experiment3 is same setting with paper's best model except training epoch.
  - Randomness(new)
    - Replace alternative pooling to random pooling (between max pooling and average pooling) in spectral clustering layers.
    - I expect the random pooling with smaller cluster size will performs better. It is because the local features can be vary to its topology, so random pooling can be helpful. (exp 2) However, random pooling can be bad for locality, so I reduce cluster size.(exp 1). The randomness shows better performance. However, I should train more epoch to compare with paper result.

Model	Experiment1		Experiment2		Experiment3		best model (paper)
Epoch	40	83	40	83	40	83	251
Pooling	Random		Random		Alternative		Alternative
Cluster size	2		4 -> 2		4 -> 2		4 -> 2
Acc	89.991	90.559	89.830	90.640	89.508	-	92.1

Code Reproduce – Dataset Analysis

- Analysis on ModelNet:
  - I visualize training results on the modelnet via with mpl_toolkits.mplot3d, Axes3D. (not model 40 - time limit)
  - The original objects themselves have very noisy face index data. Consider the fact that most of the error came from confusion between [‘desk’, ‘table’] and [‘dresser’,‘nightstand’].



48	0	0	0	0	0	0	0	0	1
2	98	0	0	0	0	0	0	0	0
0	0	100	0	0	0	0	0	0	0
0	0	0	71	1	0	0	1	10	0
0	0	0	4	77	2	10	0	0	0
0	0	0	0	0	97	0	0	0	0
0	0	0	1	8	0	73	2	1	0
0	0	0	2	0	0	0	97	0	0
0	2	0	7	0	0	3	0	89	0
0	0	0	1	0	1	0	0	0	99

There are some correlation on between certain categories (nightstand/dresser) (table/desk)

Code Reproduce – Setup

- My GCN github link: https://github.com/mjmjeong/GCN-project
  - I change some code from open code: https://github.com/fate3439/LocalSpecGCN
  - The below descriptions are also included in README.md
- Environments
  - CUDA: 10.0
  - Python: 3.6.9
  - Tensorflow: 1.13.0
  - Requirements: cv2, hd5y
- Implements procedure1
  - download dataset from https://shapenet.cs.stanford.edu/media/modelnet40_normal_resampled.zip
  - put whole directory(modelnet40_normal_resampled) in data directory

Please refer to the Dataset Download.

```
cd data
unzip modelent40_normal_resampled.zip
```

Code Reproduce – Setup

- Implements procedure2 – tf_opt
  - You should change CUDA path in .sh file.

```
cd tf_ops/3d_interpolation
sh tf_interpolate_complie.sh
cd ../grouping
sh tf_grouping_complie.sh
cd ../sampling_nd
sh tf_sampling_compile.sh
```

If framework error,
Ln –rs libtensorflow-framework.so.1 libtensorflow-framework.so 로 soft link 생성

- Implements procedure3 – train

For configuration can be modified in training_cmd.sh or train.py Then, training outputs are saved in `classification/log`.

```
cd classification
sh training_cmd.sh
```

- Implements procedure4 – evaluate

You can get trained model from classification/log/

Experiments1: spec_cp_random_csize_2
Experiments2: spec_cp_random
Experiments3: spec_cp_default

Code Analysis

- Code structure (Only the important things)

- **Classification**
  - models/pointnet2_cls_ssg_spec_cp.py
    - Get model function (22p에 설명)
  - train.py
  - Log
    - Training log, model 등 저장

- **util**
  - Spec_Graph_util.py
    - Get adj matrix function (cos or Euclidean)
    - Get Laplacian function
    - Get knn graph function
    - Spec_hier_clustering pooling function
    - Spectral convolution function

- **tf_opt**
  - This is based on 3d point cpp code.
  - Sampling_nd
    - It sampling n dimension points Model
  - Grouping
  - 3d Interpolation

Code Analysis

- classification/models/pointnet2_c ls_ssg_spec_cp.py get model

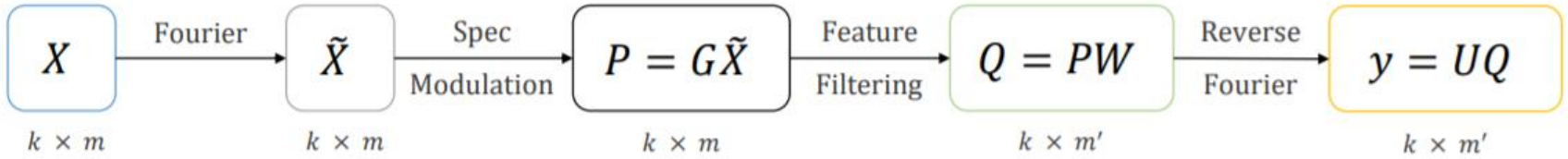
```
# Set abstraction layers
l1_xyz, l1_points, l1_indices = pointnet_sa_module(l0_xyz, l0_points, npoint=512, radius=0.2,
            nsample=32, mlp=[64,64,128], mlp2=None, group_all=False, is_training=is_training,
            bn_decay=bn_decay, scope='layer1')
l2_xyz, l2_points, l2_indices = pointnet_sa_module_spec(l1_xyz, l1_points, npoint=128,
            radius=0.4, nsample=32, mlp=[128,256], mlp2=[256], group_all=False,
            is_training=is_training, bn_decay=bn_decay, scope='layer2' , knn=True ,
            spec_conv_type = 'mlp', structure='spec' , useloc_covmat = True, pooling='max')
l3_xyz, l3_points, l3_indices = pointnet_sa_module_spec(l2_xyz, l2_points, npoint=32,
        radius=0.4, nsample=8, mlp=[256,512], mlp2=[512], group_all=False,
        is_training=is_training, bn_decay=bn_decay, scope='layer3' , knn=True,
        spec_conv_type = 'mlp', structure='spec' , useloc_covmat = True,
        pooling='hier_cluster_pool', csize = 2 )
l4_xyz, l4_points, l4_indices = pointnet_sa_module_spec(l3_xyz, l3_points, npoint=None,
        radius=None, nsample=None, mlp=[512,1024], mlp2=None, group_all=True,
        is_training=is_training, bn_decay=bn_decay, scope='layer4', knn=True ,
        spec_conv_type = 'mlp', structure='spec' , useloc_covmat = True, pooling='max')

# Fully connected layers
net = tf.reshape(l4_points, [batch_size, -1])

net = tf_util.fully_connected(net, 512, bn=True, is_training=is_training, scope='fc1', bn_deca
net = tf_util.dropout(net, keep_prob=0.5, is_training=is_training, scope='dp1')
net = tf_util.fully_connected(net, 256, bn=True, is_training=is_training, scope='fc2', bn_deca
net = tf_util.dropout(net, keep_prob=0.5, is_training=is_training, scope='dp2')
net = tf_util.fully_connected(net, 40, activation_fn=None, scope='fc3')
```

- In “get model” function
  : The property of each layers can be modified. In here, we can also change the number of nodes.

1) layer1: gcn
2) layer2: spectral gcn
3) layer3: spectral gcn
4) layer4: spectral gcn
5) Fully connected

csize: cluster size
mlp: feature dimension
nsample: number of knn sample

Code Analysis



- utlis/spec_graph_utils.py: spec_conv2d

```
# spec conv core layer function
def spec_conv2d(inputs,
                num_output_channels,
                scope,
                nn_k = None,
                local_cord = None,
                use_xavier=True,
                stddev=1e-3,
                weight_decay=0.0,
                activation_fn=None,
                bn=False,
                bn_decay=None,
                is_training=None):


    in_shape = inputs.get_shape().as_list()

    # get graph adj matrix
    W = get_adj_mat_dist_euclidean(local_cord[:,:,:,0:3] , flag_normalized = True)
    W = tf.identity(W, name='adjmat')

    # construct k nearest neighbor graph if desired
    if nn_k is not None:
        num_neigh = nn_k
        W_knn = cov_mat_k_nn_graph(W, k = num_neigh )
    else:
        W_knn = W

    # set diag to 0
    W_knn = corv_mat_setdiag_zero(W_knn)
    W_knn = tf.identity(W_knn, name='adjmat_knn')

    L = corv_mat_laplacian0(W_knn , flag_normalized = True)
    L = tf.identity(L, name='laplacian')
```

- Spec_conv2d

  : This is for spectral convolution shown in above figure

  : It get adj matrix and Laplacian during training (end to end)

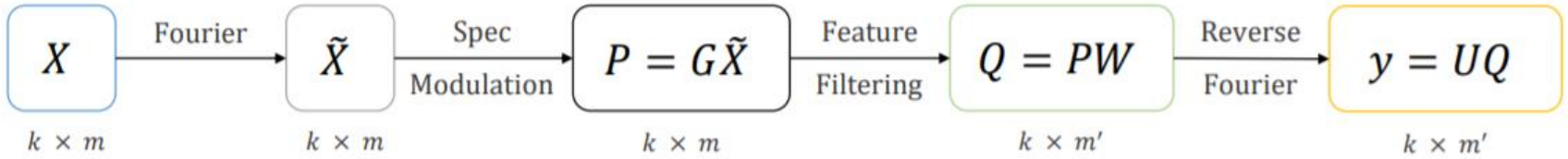
1) Get adj matrix

   - Whole adj matrix and local knn adj matrix

2) Get Laplacian matrix

   - It can be chosen between normalized L or not.

Code Analysis



- utlis/spec_graph_utils.py: spec_conv2d

```
### eigen decomp
# egvect: | | | | each vertical line is one eigen vect
# for PSD mat, SVD <-> eigen

egval , egvect = tf.self_adjoint_eig(L)
U = egvect
UT = tf.transpose(U , perm = [0,1,3,2])


# transform input to fourier domain
inputs_fourier = tf.matmul( UT , inputs)

filtered = inputs_fourier

# feat expansion
for i, num_out_channel in enumerate(num_output_channels):
    filtered = conv2d(filtered, num_out_channel, [1,1],
                      padding='VALID', stride=[1,1],
                      bn=False, is_training=is_training,
                      scope= scope + 'conv2d_%d'%(i),
                      bn_decay=bn_decay,
                      activation_fn = None)


outputs = tf.matmul( U , filtered )

#BN and relu
outputs = batch_norm_for_conv2d(outputs, is_training, bn_decay=bn_decay, scope='bn_post_spec')
outputs = tf.nn.relu(outputs)

return outputs, UT
```

- Spec_conv2d

  : This is for spectral convolution shown in above figure

  : It get adj matrix and Laplacian during training (end to end)

3) Eigen value decomposition

  - To get eigen value and eigen vector to fourier transformation

4) Convolution in spectral domain

  - feat expansion

Code Analysis

- utlis/spec_graph_utils.py: spec_hier_cluster_pool (add randomness)

```
def spec_hier_cluster_pool(inputs , pool_method = 'max' , csize = 4, use_dist_adj = False,
        fast_approx = False, include_eig = False ):
    in_shape = inputs.get_shape().as_list()
    inputs_ = inputs

    K = in_shape[2]
    eig_2nd_saved = list()

    while(K > csize and K % csize == 0):
        # in fast version, no eigen reordering is applied.
        # directly pool over near by
        # do eigen only when 1st roud or fast=false
        if (not fast_approx) or (K == in_shape[2]):
            # compute laplacian
            #adj_mat = get_adj_mat(inputs, type = '3_exp' , flag_use_laplacian = False)
            adj_mat = get_adj_mat_cos(inputs)

            L = corv_mat_laplacian(adj_mat , flag_normalized = True)
            egval , egvect = tf.self_adjoint_eig(L)
            # second smallest eigen value's egvect
            # ind = 1
            ind = tf.constant( np.array([1]) ,dtype=tf.int32)
            partition_vect = tf.squeeze( tf.gather(egvect, ind , axis=-1) ) # B N K
            eig_2nd_saved.append(partition_vect)

            # this part of sorting could be also applied to bipartite spectrual clustering
            # i.e. using median separate the 2nd eig vect, result in precise half half clustering.
            eig_2nd_sorted , sort_ind = tf.nn.top_k(input = partition_vect , k=K, sorted=True)
            # B*N , K
            sort_ind = tf.reshape(sort_ind , [in_shape[0] * in_shape[1] , K ])
            # inputs: B N K m -> BN , K,m
            inputs = tf.reshape(inputs , [in_shape[0] * in_shape[1] , K , in_shape[3]])
            # sorted K points according to 2nd eig vect; 1st half , 2nd hal forms 2 clusters
            inputs = gather_point(inputs, sort_ind) # BN , K,m
        else:
            inputs = tf.reshape(inputs , [in_shape[0] * in_shape[1] , K , in_shape[3]])
```

- Spec_hier_cluster_pool

  : node 별 feature를 node에 대해 pooling하는 layer. (node 수를 줄임)

  : spectral clustering pooling method

1) Make a graph with cos distance between points
2) Then compute Laplacian and EVD.
3) Until total number of node is smaller than csize, recursively pooling.
4) Fisrt find 2nd eigen vector, and split the vector via csize (partition vector has the indices of each cluster members.)

Code Analysis

- utlis/spec_graph_utils.py: spec_hier_cluster_pool (add randomness)

```
    else:
        inputs = tf.reshape(inputs , [in_shape[0] * in_shape[1] , K , in_shape[3]])
    # -> BN, m, k/c , c; last dimension to reduce
    inputs = tf.transpose(inputs , perm = [0,2,1])
    inputs = tf.reshape(inputs , [in_shape[0] * in_shape[1] , in_shape[3] , int(K/csize) , csiz
    if randomness:
        x = random()
        if x>0.5:
            inputs = tf.reduce_max(inputs, axis = -1)
        else:
            inputs = tf.reduce_mean(inputs, axis = -1)
    else:
    # pooling in-cluster; alternate pool method
        if pool_method == 'max':
            inputs = tf.reduce_max(inputs , axis = -1)
            pool_method = 'avg'
        elif pool_method == 'avg':
            inputs = tf.reduce_mean(inputs , axis = -1)
            pool_method = 'max'
    # BN , m, K/c
    K = int(K/csize)
    inputs = tf.reshape(tf.transpose(inputs, perm = [0,2,1]) ,
                        [in_shape[0] , in_shape[1] , K , in_shape[3]] )

# inputs now have B N K m where K <= csize, reduce on -2 dim
if pool_method == 'max':
    inputs = tf.reduce_max(inputs , axis = -2, keep_dims = True)
    pool_method = 'avg'
elif pool_method == 'avg':
    inputs = tf.reduce_mean(inputs , axis = -2, keep_dims = True)
    pool_method = 'max'

outputs = inputs
return outputs
```

- Spec_hier_cluster_pool

  : node 별 feature를 node에 대해 pooling하는 layer. (node 수를 줄임)

  : spectral clustering pooling method

5) Combine each cluster to one node(pooling)

6-1) Previous method, just alternative select between max pooling and average pooling.

6-2) Changed method, random select between max pooling and average pooling

7) If total number of nodes are smaller than csize, pooling those nodes via average or max pooling.

-> then get final one node. (finish)

Thank you