Local Spectral Graph Convolution
for Point Set Feature Learning

Chu Wang, Babak Samari, and Kaleem Siddiqi(✉)

School of Computer Science and Center for Intelligent Machines, McGill University,
Montreal, Canada
{chuwang,babak,siddiqi}@cim.mcgill.ca

Abstract. Feature learning on point clouds has shown great promise, with the introduction of effective and generalizable deep learning frameworks such as pointnet++. Thus far, however, point features have been abstracted in an independent and isolated manner, ignoring the relative layout of neighboring points as well as their features. In the present article, we propose to overcome this limitation by using spectral graph convolution on a local graph, combined with a novel graph pooling strategy. In our approach, graph convolution is carried out on a nearest neighbor graph constructed from a point's neighborhood, such that features are jointly learned. We replace the standard max pooling step with a recursive clustering and pooling strategy, devised to aggregate information from within clusters of nodes that are close to one another in their spectral coordinates, leading to richer overall feature descriptors. Through extensive experiments on diverse datasets, we show a consistent demonstrable advantage for the tasks of both point set classification and segmentation. Our implementations are available at https://github.com/fate3439/LocalSpecGCN.

Keywords: Point set features · Graph convolution
Spectral filtering · Spectral coordinates · Clustering · Deep learning

1 Introduction

With the present availability of registered depth and appearance images of complex real-world scenes, there is tremendous interest in feature processing algorithms for classic computer vision problems including object detection, classification and segmentation. In their latest incarnation, for example, depth sensors are now found in the Apple iPhone X camera, making a whole new range of computer vision technology available to the common user. For such data it is particularly attractive to work *directly* with the unorganized 3D point clouds and to not require an intermediate representation such as a surface mesh. The processing of 3D point clouds from such sensors remains challenging, since the sensed depth points can vary in spatial density, can be incomplete due to occlusion or perspective effects and can suffer from sensor noise.

Motivated by the need to handle unstructured 3D point clouds while leveraging the power of deep neural networks, the pointnet++ framework has shown promise for 3D point cloud feature processing for the tasks of recognition and segmentation [1]. In this approach a network structure is designed to work directly with point cloud data, while aggregating information in a hierarchical fashion, in the spirit of traditional CNNs on regular grids. To do so, a centroid sampling is first applied on the input point cloud, followed by a radius search to form point neighborhoods. Then the point neighborhoods are processed by multi-layer perceptrons [2] and the resulting point features are abstracted by a pooling operation. Through hierarchical multi-layer learning on the point cloud data, the pointnet++ framework exhibits impressive performance in both segmentation and classification on challenging benchmarks, while treating the input data as an unorganized point cloud.

In a parallel development, Defferrard *et al.* have sought to extend CNNs, traditionally applied on regular domains, such as sampled image pixels in 2D or voxels in 3D, to irregular domains represented as graphs [3]. Their approach uses Chebyshev polynomials to approximate spectral graph filters. Here an initial graph is processed by convolutional operations to yield features which are then coarsened using sub-sampling and pooling methods. Kipf and Welling [4] simplify the higher order polynomial approximations in Defferrard *et al.* and propose a first order linear approximation of spectral graph filters. The aforementioned spectral approaches operate on the full graph and have the limitation that the graph Laplacian and the graph coarsening hierarchy have to be precomputed, in an offline manner, before network training or testing. This adds significant overhead when the full graph is large.

In this article we propose to leverage the power of spectral graph CNNs in the pointnet++ framework, while adopting a different pooling strategy. This allows us to address two limitations of present deep learning methods from point clouds: (1) the fact that for each point sample the learning of features is carried out in an isolated manner in a local neighborhood and (2) that the aggregation of information in later layers uses a greedy winner-take-all max pooling strategy. Instead, we adopt a different pooling module, as illustrated by the example in Fig. 1. Further, our method requires no precomputation, in contrast to existing spectral graph CNN approaches [3,4]. Our combination of local spectral feature learning with recursive clustering and pooling provides a novel architecture for point set feature abstraction from unorganized point clouds. Our main methodological contributions are the following:

– The use of local spectral graph convolution in point set feature learning to incorporate structural information in the neighborhood of each point.
– An implementation of the local spectral graph convolution layer that requires no offline computation and is trainable in an end-to-end manner. We build the graph dynamically during runtime and compute the Laplacian and pooling hierarchy on the fly.
– The use of a novel and effective graph pooling strategy, which aggregates features at graph nodes by recursively clustering the spectral coordinates.
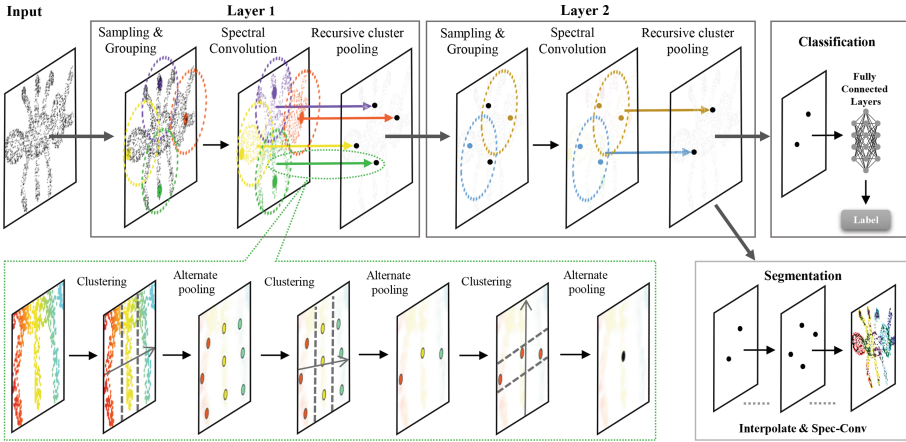
Fig. 1. Top: Starting from a point cloud, farthest point sampling leads to centroids, from which k-NN's are sampled. Then, for each neighborhood, spectral convolution is carried out followed by recursive cluster pooling. After several layers of sampling, spectral convolution and cluster pooling, we perform segmentation or classification. Bottom: The green dashed box details the process of recursive spectral cluster pooling on the Fiedler vector of a sample neighborhood. See text in Sect. 4 for a discussion.

The proposed architecture leads to new state-of-the-art object recognition and segmentation results on diverse datasets, as demonstrated by extensive experiments.

2   Challenges in Point Set Feature Learning

A limitation of feature learning in the pointnet++ framework [1], is that features from the k nearest neighbors (k-NN) of a point are learned in an isolated fashion. Let h represent the output of an arbitrary hidden layer in a deep network, typically a multilayer perceptron. In pointnet++ the individual features for each point in the k-NN are achieved with $h(x_i), i \in 1, 2, ..., k$. Unfortunately, this hidden layer function does not model the joint relationship between points in the k-NN. A convolution kernel that jointly learns features from all points in the k-NN would capture topological information related to the geometric layout of the points, as well as features related to the input point samples themselves, e.g., color, texture, or other attributes. In the following section we shall extend approaches such as the pointnet++ framework to achieve this goal by using local graph convolution, but in the spectral domain.

Another limitation in pointnet++ is that the set activation function for the k-NN is achieved by max pooling across the hidden layer's output for each point, such that

$$f(x_1, x_2, ..., x_k) = \max_{i \in 1, ..., k} h(x_i). \tag{1}$$

Max pooling does not allow for the preservation of information from disjoint sets of points within the neighborhood, as the legs of the ant in the example in Fig. 1. To address this limitation we introduce a recursive spectral clustering and pooling module that yields an improved set activation function for the k-NN, as discussed in Sect. 4. The combined point set feature abstraction operation in this paper can be summarized by

$$f(x_1, x_2, ..., x_k) = \oplus(h_1, h_2, ..., h_k), \tag{2}$$

where h_i is the convolution output $h(x_1, x_2, ..., x_k)$ evaluated at the i-th point and $\oplus$ represents our set activation function.

Figure 2 provides a comparison between the point-wise MLP in pointnet++ [1] and our spectral graph convolution, to highlight the differences. Whereas pointnet++ abstracts point features in an isolated manner, spectral graph convolution considers all points in a local neighborhood in a joint manner, incorporating both features at neighboring points as well as structural information encoded in the graph topology. This is accomplished via the graph Fourier transform and spectral modulation steps, which blend neighborhood features using the eigenspace of the graph Laplacian (see Fig. 3). In the following section we provide theoretical background and discuss implementation details of our spectral graph convolution kernel.
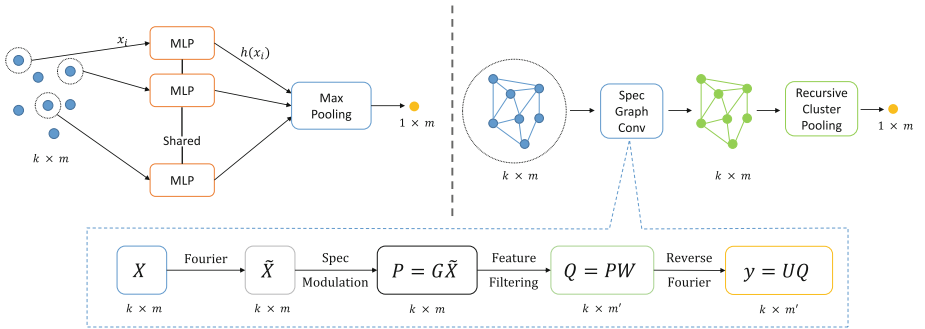

Fig. 2. TOP: A comparison between the point-wise MLP of pointnet++ (left) and our spectral graph convolution (right) for a local neighborhood of a point. The spectral graph convolution output depends on all points in the neighborhood, whereas in point-wise MLP the output depends only on the point itself. BOTTOM: The network operations in a spectral graph convolutional layer along with the input/output dimensions.

3   Graph Convolution

The convolution operation on vertices in the graph is described by

$$h = X * g, \tag{3}$$

where X stands for the input point set features and g for a graph convolution kernel. This is equivalent to an element-wise Hadamard product in the graph spectral domain, as shown in Defferrard *et al.* [3] and Shuman *et al.* [5]:

$$\tilde{h} = \tilde{X} \odot \tilde{g}. \tag{4}$$

Here $\tilde{X}$ stands for the graph Fourier transform of the point set features, $\tilde{g}$ stands for the filter in the graph Fourier domain and $\tilde{h}$ for the filtered output. In order to acquire the filtered output in the original (vertex) domain, an inverse Fourier transform is required. We elaborate on the graph Fourier transform and spectrum filtering steps below.

3.1   Graph Formulation of a Local Neighborhood

Given a set of k points $x_1, x_2, ..., x_k$ in a local neighborhood, we build a representation graph G_k whose vertices V are the points and whose edges $E \subseteq V \times V$ carry weights $a : E \rightarrow \mathbb{R}_+^*$ which represent pair-wise distance between vertices in a feature space. This provides a $k \times k$ nonnegative symmetric graph adjacency matrix A, with entries $A_{ij} = dist(x_i, x_j)$. We then compute the graph spectrum based on this adjacency matrix and perform a graph Fourier transform, spectral filtering and finally an inverse Fourier transform.

3.2   Graph Fourier Transform

To compute a graph Fourier Transform of the point features $X \in \mathbb{R}^{k \times m}$, which are graph signals on vertices of G_k, we first need to compute the normalized graph Laplacian defined as

$$\mathcal{L} = I - D^{-1/2} A D^{-1/2}, \tag{5}$$

where I is the identity matrix and $D \in \mathbb{R}^{k \times k}$ is the diagonal degree matrix with entries $D_{ii} = \sum_j A_{ij}$. It follows that $\mathcal{L}$ is a real symmetric positive semidefinite matrix, and has a complete set of orthonormal eigenvectors which comprise the graph Fourier basis

$$U = [u_0, u_1, ..., u_{k-1}] \in \mathbb{R}^{k \times k}. \tag{6}$$

The eigenvalues can be used to construct a diagonal matrix

$$\Lambda = diag([\lambda_0, \lambda_1, ..., \lambda_{k-1}]) \in \mathbb{R}^{k \times k} \tag{7}$$

which contains the frequencies of the graph. Then it follows that $\mathcal{L} = U \Lambda U^T$. The graph Fourier transform of X is then defined as $\tilde{X} = U^T X$ and its inverse as $X = U \tilde{X}$.

3.3   Spectral Filtering

The convolution operation is defined in the Fourier domain as

$$x * g = U((U^T x) \odot (U^T g)), \tag{8}$$

following Shuman *et al.* [5], where $\odot$ is the element-wise Hadamard product, x is an arbitrary graph signal and g is a spatial filter. If we define $y = x * g$ as the output of the graph convolution, it follows that a graph signal $X \in \mathbb{R}^{k \times m}$ filtered by g can be written as

$$y = \tilde{g}_\theta(\mathcal{L})X = \tilde{g}_\theta(U\Lambda U^T)X = U\tilde{g}_\theta(\Lambda)\tilde{X}, \tag{9}$$

where θ stands for an arbitrary parametrization. In the following section, we describe our implementation of spectral filtering, which is introduced as a module on top of the existing pointnet++ [1] framework, using TensorFlow.

3.4   Implementation of Spectral Filtering

We carry out spectral graph convolution using standard unparametrized Fourier kernels, where the entries of $\tilde{g}_\theta(\Lambda)$ are all learnable. With m the input feature dimension and m' the output dimension, convolution of a graph signal $X \in \mathbb{R}^{k \times m}$ with spectral filters can be achieved by the following three steps:

1. **Spectral modulation** which outputs $P = G\tilde{X}$, with the diagonal matrix G being the unparametrized kernel $\tilde{g}_\theta(\Lambda)$. The k diagonal entries of G are all free parameters in the unparametrized Fourier kernel formulation.
2. **Feature filtering** which expands the input dimension from m to m'. The output of this step is a feature matrix $Q \in \mathbb{R}^{k \times m'}$. The entry $q_{k,i}$ is the i-th output feature of the k-th point and is given by $q_{k,i} = \sum_{j=1}^{m} p_{k,j} w_{j,i}$. Here $p_{k,j}$ is the entry of P corresponding to the j-th input feature of the k-th point defined in the previous step and $w_{j,i}$ is the filter coefficient between the i-th input feature with j-th output feature. This step can be represented by $Q = PW$, where W is the matrix of learnable filter parameters. The filtering operation in steps 1 and 2 can be summarized as

$$Q = (G\tilde{X})W. \tag{10}$$

3. **Reverse Fourier transform** which provides convolution outputs in the spatial graph signal domain via $y = UQ$.

The above formulation resembles that of [3,4], with the difference that we build the k-NN graph during runtime, computing its Laplacian and pooling hierarchy on the fly, so that no offline precomputation is required. We further note that the weights of the feature filter W, as well as the spectral modulation matrix G, are shared by all the different local neighborhoods in a given graph convolution layer. Thus, unlike [3,4], the learned parameters in our work do not depend on the underlying graph structure. Figure 2 (bottom) summarizes the above spectral filtering process.

While the more sophisticated efficient kernels of [3] could be used, our goal was to demonstrate the improvement obtained by graph CNNs in general. The overhead due to eigenvalue decomposition (EVD) in our unparametrized spectral kernel does not significantly affect runtime since the EVD is computed on local k-NN graphs, with k being very small. This computation is easily handled by parallel computing on GPUs, as demonstrated by the experiments which describe training time cost in our model ablation study in Sect. 5.3. Our approach is also robust to potential ambiguity in eigenvector sign in EVD computation, for both the spectral graph convolution and cluster pooling operations, as discussed in the arXiv version of this paper at https://arxiv.org/abs/1803.05827.

4   Pooling on Local k-NN Graph

The set activation function discussed in Sect. 2, whose aim is to summarize information from a k-NN graph, is essentially a form of graph pooling, where a graph of k vertices is abstracted via feature pooling to a single vertex. We propose a novel k-NN graph pooling algorithm using hierarchical clustering and within-cluster pooling. Our method is inspired by the results in [6], where the benefit of recursive cluster pooling across multiple views in 3D object recognition is demonstrated.

The general strategy is to pool information during learning in a manner that does so only *within* a cluster of similar abstract point features, in contrast to the greedy strategy of max pooling, which is commonly applied in regular CNNs as well as in pointnet++. The intuition here is that multiple sets of distinct features may together contribute towards a salient property for the task at hand, and that the detection of these clusters combined with within cluster pooling will improve performance. For example, for the task of classifying a point set sampled from a human head one would want to simultaneously learn and capture nose-like, chin-like and ear-like features and not assume that only one of these would be discriminative for this object category. We discuss each of our steps in turn.

4.1   Spectral Clustering

We propose to group features into clusters according to a local k-NN's geometric information that is embedded using spectral coordinates. The spectral coordinates we use are based on the low frequency eigenvectors of the Graph Laplacian $\mathcal{L}$, which capture coarse shape properties since the Laplacian itself encodes pairwise distances in the k-NN graph. As exploited in [7,8] for computing point-to-point correspondences on meshes or for feature description, the low frequency spectral coordinates provide a discriminative spectral embedding of an object's local geometry (see Fig. 1 bottom and Fig. 3). The eigenvector corresponding to the second smallest eigenvalue, the Fiedler vector [9], is widely used for spectral clustering [10].
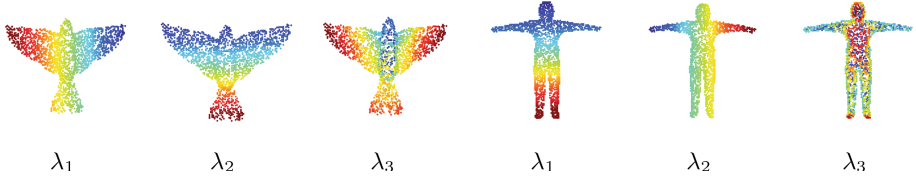
λ_1 λ_2 λ_3 λ_1 λ_2 λ_3

Fig. 3. A visualization of spectral coordinates for models of a bird (left) and a human (right), both from the McGill Shape Benchmark. $\lambda_i, i \in \{0, 1, ..., k-1\}$ is the i-th eigenvalue of the graph Laplacian.

4.2   Clustering, Pooling and Recurrence

Following the normalized cuts clustering algorithm [10], we partition the Fiedler vector to perform spectral clustering in a local k-NN. We first sort the entries of the Fiedler vector in ascending order of their numerical value, and then evenly cut it into k_1 sections in the first iteration. The points in the neighborhood whose Fiedler vector entries fall in the same section will be clustered together. This results in k_1 clusters with a cluster size of $c = \frac{k}{k_1}$. After obtaining a partition into k_1 clusters, we perform pooling operations only *within* each cluster. This allows the network to take advantage of separated features from possibly disjoint components in the k-NN graph.

Algorithm 1. Recursive Cluster Pooling

Inputs: pts ← point features ($\mathbb{R}^{k \times m}$), csize ← cluster size, and POOL ← the Pool method.

Output: Pooled point features ($\mathbb{R}^{1 \times m}$).

Methods: ARG_SORT(x) returns the sorted indices of x, REARRANGE(x,y) permutes x along its 1st dimension according to the given indices y, POOL(x) pools x along its 1st dimension.

procedure CLUSTER(pts, csize, POOL)
    $G \leftarrow$ ADJ_MATRIX(pts)
    $L \leftarrow$ LAPLACIAN(G)
    $[\Lambda, U] \leftarrow$ EVD(L)
    fiedler_vector← $U[:, 1]$
    inds← ARG_SORT(fiedler_vector)
    REARRANGE(pts, inds)
    RESHAPE(pts, [:, csize])
    return POOL(pts)

procedure MAIN(pts, csize)
    POOL← MAX_POOL
    while COUNT(pts)>csize **do**
        pts←CLUSTER(pts, csize, POOL)
        if POOL == MAX_POOL **then**
            POOL← AVG_POOL
        else
            POOL← MAX_POOL
    return POOL(pts)

The above steps result in a coarsened k_1-NN graph with aggregated point features. The same process is then repeated on the coarsened graph in a recursive manner, to obtain k_i clusters for each iteration i. We alternate between max

pooling and average pooling between different recurrences to further increase the discriminative power of the graph pooling algorithm. The proposed algorithm terminates when the number of vertices remaining is smaller or equal to a prescribed cluster size. A regular full stride pooling is then applied on the resultant graph signals. We formalize the above steps in Algorithm 1.

In practice, we found that using $k = 2c^2$ as a relationship between cluster size c and neighborhood k size gave good results, with two recurrences of cluster pooling and a final pooling of size 2. We used max pooling as the first stage in the alternating pooling scheme, and fixed these configurations for all our experiments in Sect. 5. We implemented the recursive cluster pooling module in TensorFlow, integrating it fully with the spectral graph convolution layer to make the entire network end-to-end trainable.

5   Experiments

5.1   Datasets

We evaluate our approach against present state-of-the-art methods on:

– MNIST, which contains images of handwritten digits with 60k training and 10k testing samples. It has been used to benchmark related graph CNN approaches [3,11] as well as pointnet++ [1].
– ModelNet40 [12], which contains CAD models of 40 categories, sampled as point clouds. We use the official split, with 9,843 training and 2,468 testing examples.
– McGill Shape Benchmark [13], which contains 254 articulated object models and 202 non-articulated ones, from 19 categories. We sample the meshes into point clouds and use the first two-thirds of the examples in each category for training and the remaining one-third for testing.
– ShapeNet part segmentation dataset [14], which contains 16,881 shapes from 16 classes, with the points in each model labeled into one of 50 part types. We use the official training/testing split, following [2,14,15], where the challenge is to assign a part label to each point in the test set.
– ScanNet Indoor Scene dataset [16], which contains 1513 scanned and reconstructed indoor scenes, with annotations including semantic voxel labels. We follow the experimental settings for segmentation in [1,16] and use 1201 scenes for training, and 312 scenes for testing.

5.2   Network Architecture and Training

We provide details of our network structures for the case of 1024 input points (1k) and 2048 input points (2k). The network structure for the 2k experiments is designed to be "wider" to better accommodate the increased input point density. Table 1 lists all the variations of the pointnet++ and our spectral point convolution network structures, which we will later to refer to when presenting

Table 1. Network architectures for the 1k experiments (top) and the 2k experiments (bottom). Here, for each layer, C stands for the number of centroids, k stands for the size of the k-NN, and m stands for the output feature dimension.

1k Structure		L1	L2	L3	L4	Kernel	Pooling
3l-pointnet++	C	512	128	1	-		
	k	64	64	128	-	point MLP	max
	m	128	256	1024	-		
4l-pointnet++	C	512	128	32	1		
	k	32	32	8	32	point MLP	max
	m	128	256	512	1024		
2k Structure		L1	L2	L3	L4	Kernel	Pooling
3l-pointnet++	C	1024	256	1	-		
	k	64	64	256	-	point MLP	max
	m	128	256	1024	-		
4l-pointnet++	C	1024	256	64	1		
	k	32	32	8	64	point MLP	max
	m	128	256	512	1024		

experimental results. The 3l-pointnet++ is defined in the "pointnet2_cls_ssg.py" model file on the pointnet++ GitHub page. We replace the kernels from 4l-pointnet++ with spectral graph convolution kernels to acquire the *4l-spec-max* model. Replacing max pooling with recursive cluster pooling in the 4l-spec-max model results in the *4l-spec-cp* model. Details of the network structures and the parameters used in the training process are described in the arXiv version of this paper at https://arxiv.org/abs/1803.05827.

5.3   Ablation Study for Network Models

We first evaluate the effect of the novel components in our framework: (1) local spectral filtering on a k-NN and (2) recursive cluster pooling for local outputs. We apply a 4 layer pointnet++ structure as the baseline method, then add spectral filters to each layer, and then replace max pooling with recursive cluster pooling. We also include results obtained using the 3 layer structure used in [1]. In addition, we consider the scalability of both approaches as well as their training time, under the conditions of varying the number of input points and including additional features such as surface normals. These results are presented in Table 2.

From the model ablation study in Table 2 it is evident that our proposed model, which incorporates spectral graph convolution together with recursive cluster pooling, provides a non-trivial improvement over pointnet++ on both the classification and segmentation tasks. We make the following observations: (1) *3l-pointnet++ performs better than the 4 layer version.* This is likely because

in the 4 layer version the neighborhood size is half of that in the 3 layer version. Since features are learned at each point in an isolated fashion in pointnet++, the use of larger neighborhoods gives an advantage. (2) *Spectral graph convolution on local k-NNs performs better than point-wise MLP.* The 4l-spec-max model outperforms 4l-pointnet++. This implies that the topological information encoded by spectral graph convolution improves feature learning. (3) *Recursive cluster pooling further boosts the performance of the spectral graph convolution layer.* This suggests that information aggregation following spectral coordinates increases the discriminative power of the learned point features, benefiting both classification and segmentation. (4) *The runtime of our model is comparable to those of pointnet++.* The eigenvalue decomposition used in spectral convolution and recursive cluster pooling could in theory be costly, but since we use local neighborhoods the impact is not severe. Our best model, 4l-spec-cp, has roughly the same training time as that of 3l-pointnet++, which is the best model from pointnet++. Spectral graph convolution kernels are as fast as the point-wise MLP kernels, which can be seen by comparing the runtime of the 4l-spec-max and 4l-pointnet++ models.

Table 2. Model Ablation Study on ModelNet40 (classification) and ShapeNet (segmentation). Acc stands for classification accuracy, 1k/2k refers to the number of points used and "+N" indicates the addition of surface normal features to xyz spatial coordinates. The "2k + N" column shows results averaged over 5 runs. For the segmentation experiments, mIOU stands for mean intersection over union. Here we only compare the best models from pointnet++ with ours. Training time is with respect to the number of epochs used in each experiment. Adding normals only increases the training time by a negligible amount, therefore only one runtime column is provided for the 1k experiments. The network structures for these experiments are described in Table 1.

ModelNet40	Acc 1k	Acc 1k +N	Time 250ep	Acc 2k +N	Time 250ep
3l-pointnet++	90.7	91.3	11h	91.47 ± 0.11	20h
4l-pointnet++	90.6	91.1	7.5h	91.19 ± 0.11	11h
4l-spec-max	91.2	91.6	8h	91.83 ± 0.08	12h
4l-spec-cp	91.5	91.8	12h	92.03 ± 0.09	20h
ShapeNet Seg	mIOU 1k	mIOU 1k +N	Time 100ep	mIOU 2k +N	Time 100ep
3l-pointnet++	84.2	84.7	7.5h	84.84 ± 0.10	14h
4l-spec-cp	84.6	85.0	8h	85.37 ± 0.09	14h

We now provide comparisons against the present state-of-the-art methods in classification and segmentation, using the datasets described in Sect. 5.1. When comparing against pointnet++, unless stated otherwise, we apply the 3l-pointnet++ model since it gives better results than the 4 layer version in our model ablation study in Table 2.

5.4   Classification Experiments

McGill Shape Benchmark. The classification results for the McGill Shape Benchmark are presented in Table 3, using 1024 xyz points as the inputs in all cases. Spectral graph convolution on point sets provides a consistent boost in both average instance level accuracy and category level accuracy. The use of recursive cluster pooling gives our model another 0.7% boost in overall instance level accuracy over max pooling. Since the k-NNs may contain disjoint sets of points, a recursive aggregation of k-NN features by clustering the spectral coordinates appears to increase discriminative power for articulated objects.

Table 3. McGill Shape Benchmark classification results. We report the instance and category level accuracy on both the entire database and on subsets (see Table 1 for network structures).

Model	Kernel	Pooling	Instance Level Accuracy (%)			Category Level Accuracy (%)		
			Arti	Non-Arti	Combined	Arti	Non-Arti	Combined
3l-pointnet++	point-MLP	max	91.25	95.31	93.06	91.33	95.44	93.27
4l-pointnet++	point-MLP	max	92.50	92.19	92.36	92.83	92.74	92.79
4l-spec-cp	spec-conv	max	92.50	98.44	95.14	92.75	98.41	95.43
	spec-conv	cp	93.75	98.44	95.83	93.30	98.41	95.74

Table 4. MNIST classification results. To obtain the pointnet++ results we reproduced the experiments discussed in [1].

Method	Domain	Kernel	Error Rate(%)
Multi-layer perceptron [17]	full image	spatial MLP	1.60
LeNet5 [18]	local img patch	spatial conv	0.80
Network in Network [19]	local img patch	spatial conv	0.47
ChebNet [3]	full graph	spectral graph conv	0.86
MoNet [11]	local graph	spatial graph conv	0.81
3l-pointnet++ [1]	local points	spatial point-MLP	0.55
4l-spec-cp	local k-NN graph	spectral graph conv	0.42

MNIST Dataset. 2D images can be treated as a grid graph [3,11] or a 2D point cloud [1]. We provide results on the MNIST dataset using our proposed best model, 4l-spec-cp, from the previous model ablation study. We compare our results with the state-of-the-art methods in graph CNNs [3,11], in point sets [1] (our 0.55% error rate using pointnet++ is very close to that reported by the authors) and with regular neural network/CNN approaches applied on the 2D image domain [17–19].

For both pointnet++ and our method, 784 points are provided as inputs to the network and we use the 1k experimental network, where the first layer samples 512 centroids (see Table 1). The results in Table 4 show that approaches which favor local operations on the input domain usually yield better performance, for instance, MLP vs. LeNet, and our method vs. ChebNet. Our approach gives a 20% error rate reduction over pointnet++, demonstrating the advantage of spectral convolution on a local k-NN graph over the isolated learning process in point-wise MLP. In addition, our performance surpasses that of the Network in Network model [19], which is a strong regular image CNN model.

ModelNet40 Dataset. We present ModelNet40 3D shape recognition results in Table 5, where we compare our method with representative competitive approaches. We were able to reproduce the results from pointnet++, to get very similar performance to that reported by the authors in [1]. We report two sets of accuracy results. In the first 1024 xyz point coordinates are used as inputs, with the network structure following the 1k configurations in Table 1. In the second 2048 xyz points along with their surface normals are used as inputs, with the network structure following the 2k configurations in Table 1. Our use of spectral graph convolution and recursive cluster pooling provides a consistent improvement over pointnet++, and leads to state-of-the-art level classification performance on the ModelNet40 Benchmark.

Table 5. ModelNet40 results. "Acc" stands for 1k experiments with only xyz points as input features. "Acc + N" stands for 2k experiments with xyz points along with their surface normals as input features. "graph-cp" stands for recursive cluster pooling.

Method	Domain	Kernel	Pooling	Acc (%)	Acc + N (%)
Subvolume [20]	Voxel Grid	3D conv	3D-max	89.2	-
MVCNN [21]	2D views	2D conv	view-max	90.1	-
PointNet [2]	Full Points	point-MLP	point-max	89.2	-
3l-pointnet++ [1]	Local Points	point-MLP	point-max	90.7	91.5
4l-spec-max	Local k-NN graph	graph conv	graph-max	91.2	91.9
4l-spec-cp	Local k-NN graph	graph conv	graph-cp	91.5	92.1

5.5   Segmentation Experiments

Point segmentation is a labeling task for each point in a 3D point set, and is more challenging than point set classification. We present experimental results on ShapeNet [14] and ScanNet [16] in Table 6. We then provide details on experimental settings for each case.

Table 6. Segmentation Results. We compare our method with the state-of-the-art approaches, as well as with the results from pointnet++, which we have been able to reproduce experimentally. For ShapeNet, mIOU stands for mean intersection over union on points, and for ScanNet, Acc stands for voxel label prediction accuracy.

Method	Domain	Kernel	ShapeNet-mIOU (%)	ScanNet-Acc (%)
Yi *et al.* [14]	-	-	81.4	-
Dai *et al.* [16]	Voxel Grid	3D conv	-	73.0
SynSpecCNN [22]	Full k-NN Graph	graph conv	84.7	-
PointNet [2]	Full Points	point-MLP	83.7	73.9
Pointnet++ [1]	Local Points	point-MLP	84.9	84.0
4l-spec-cp	Local k-NN Graphs	graph conv	85.4	84.8

Shapenet Part Segmentation Dataset. Following the setting in [14], we evaluate our approach assuming that a category label for each shape is already known and we use the same mIoU (mean intersection over union) measurement on points. 2048 xyz points and their surface normals are used as input features and the network structure follows that of the 2k configurations in Table 1. More specifically, the 3l-pointnet++ model is applied for pointnet++ and the 4l-spec-cp is applied for our method.

ScanNet Dataset. ScanNet is a large-scale semantic segmentation dataset constructed from real-world 3D scans of indoor scenes, and as such is more challenging than the synthesized 3D models in ShapeNet. Following [1,16], we remove RGB information in our experiments in Table 6 and we use the semantic voxel label prediction accuracy for evaluation. The training and testing procedures follow those in pointnet++ [1]. 8192 xyz points are used as input features and the network structure is that of the 2k configurations in Table 1. More specifically, the 4l-pointnet++ model is applied for pointnet++ and the 4l-spec-cp is applied for our method.[1]

The use of spectral graph convolution combined with cluster pooling in our approach once again provides a non-trivial improvement over pointnet++, achieving state-of-the-art level performance on part segmentation on the ShapeNet dataset and indoor scene semantic segmentation on the ScanNet dataset. We provide an illustrative visualization of the part segmentation results on selected models in Fig. 4. In these examples, when compared to pointnet++, our approach gives results that are closer to the ground truth overall and it better captures fine local structures, such as the axles of the skateboard, and the feet of the table. In addition, spectral graph convolution with cluster pooling provides a more faithful representation of changes in local geometry. This allows us to successfully segment connected parts of a 3D object, such as the strap from

---

[1] The 3l-pointnet++ model leads to inferior performance on this large-scale indoor dataset. For both networks, for all experiments reported in this paper, single scale grouping (SSG in [1]) is applied for a fair comparison.
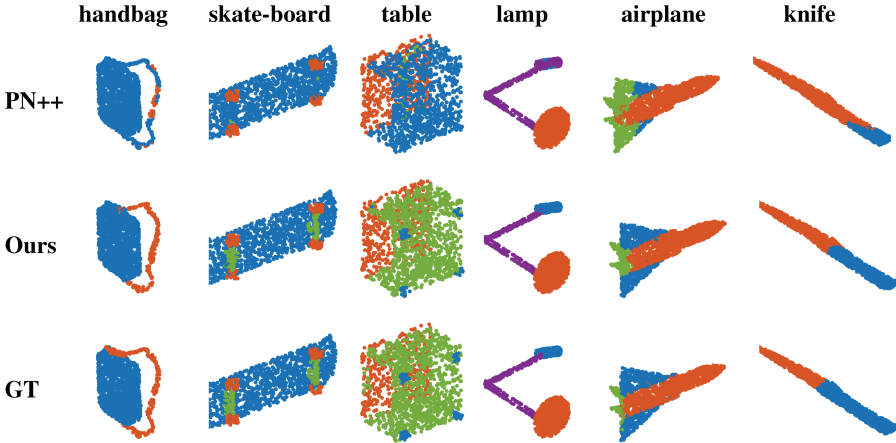
handbag	skate-board	table	lamp	airplane	knife

Fig. 4. A visualization of segmentation results on ShapeNet using pointnet++ (top row, 84.9% in Table 6), our 41-spec-cp model (middle row, 85.4% in Table 6), and the ground truth labels (bottom row). Our method appears to better capture fine local structures (see text for a discussion).

the body of the handbag, the wings from the tail fins of the airplane and the handle from the blade of the knife.

6   Conclusion

The use of spectral graph convolution on local point neighborhoods, followed by recursive cluster pooling on the derived representations, holds great promise for feature learning from unorganized point sets. Our method's ability to capture local structural information and geometric cues from such data presents an advance in deep learning approaches to feature abstraction for applications in computer vision. With its strong experimental performance, acceptable runtime, and versatility in handling a variety of datasets and tasks, our approach could have considerable practical value as 3D depth sensors begin to become more and more commonplace. The approach is not limited in application to point sets derived from cameras. It can also be applied in settings where the vertices carry a more abstract interpretation, such as nodes in a graph representing a social network, where local feature attributes could play an important role.

References

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

2. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: CVPR (2016)
3. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS (2016)
4. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
5. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Sig. Process. Mag. **30**(3), 83–98 (2013)
6. Wang, C., Pelillo, M., Siddiqi, K.: Dominant set clustering and pooling for multi-view 3D object recognition. In: BMVC (2017)
7. Lombaert, H., Grady, L., Cheriet, F.: FOCUSR: feature oriented correspondence using spectral regularization-a method for precise surface matching. IEEE Trans. Pattern Anal. Mach. Intell. (2013)
8. Bronstein, A.M., Bronstein, M.M., Kimmel, R.: Numerical Geometry of Non-rigid Shapes. Monographs in Computer Science. Springer, New York (2008). https://doi.org/10.1007/978-0-387-73301-2
9. Chung, F.R.: Spectral Graph Theory. Number 92 in Regional Conference Series in Mathematics. Am. Mathe. Soc. (1997)
10. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
11. Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model CNNs. In: CVPR (2017)
12. Wu, Z., et al.: 3D shapenets: a deep representation for volumetric shapes. In: CVPR (2015)
13. Siddiqi, K., Zhang, J., Macrini, D., Shokoufandeh, A., Bouix, S., Dickinson, S.: Retrieving articulated 3D models using medial surfaces. Mach. Vis. Appl. **19**(4), 261–274 (2008)
14. Yi, L., et al.: A scalable active framework for region annotation in 3D shape collections. ACM Trans. Graph. (TOG) (2016)
15. Yi, L., Su, H., Guo, X., Guibas, L.: SyncSpecCNN: synchronized spectral CNN for 3D shape segmentation. In: CVPR (2017)
16. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: richly-annotated 3D reconstructions of indoor scenes. In: CVPR (2017)
17. Simard, P.Y., Steinkraus, D., Platt, J.C., et al.: Best practices for convolutional neural networks applied to visual document analysis. In: ICDAR (2003)
18. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
19. Lin, M., Chen, Q., Yan, S.: Network in network. In: ICLR (2014)
20. Qi, C.R., Su, H., Niessner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view CNNs for object classification on 3D data. In: CVPR (2016)
21. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3D shape recognition. In: ICCV (2015)