Differentiable Convolution Search for Point Cloud Processing

Xing Nie[1,2], Yongcheng Liu[1], Shaohong Chen[4], Jianlong Chang[3],
Chunlei Huo[1*], Gaofeng Meng[1,2,5], Qi Tian[3], Weiming Hu[1], Chunhong Pan[1],

[1] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences.
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences. [3] Huawei Cloud & AI.
[4] Xidian University. [5] Centre for Artificial Intelligence and Robotics, HK Institute of Science & Innovation, CAS.
Email: niexing2019@ia.ac.cn, {yongcheng.liu, clhuo}@nlpr.ia.ac.cn

Abstract

Exploiting convolutional neural networks for point cloud processing is quite challenging, due to the inherent irregular distribution and discrete shape representation of point clouds. To address these problems, many handcrafted convolution variants have sprung up in recent years. Though with elaborate design, these variants could be far from optimal in sufficiently capturing diverse shapes formed by discrete points. In this paper, we propose PointSeaConv, i.e., a novel differential convolution search paradigm on point clouds. It can work in a purely data-driven manner and thus is capable of auto-creating a group of suitable convolutions for geometric shape modeling. We also propose a joint optimization framework for simultaneous search of internal convolution and external architecture, and introduce epsilon-greedy algorithm to alleviate the effect of discretization error. As a result, PointSeaNet, a deep network that is sufficient to capture geometric shapes at both convolution level and architecture level, can be searched out for point cloud processing. Extensive experiments strongly evidence that our proposed PointSeaNet surpasses current handcrafted deep models on challenging benchmarks across multiple tasks with remarkable margins.

1. Introduction

Recently, 3D point cloud processing has received great attention, since it plays an important role in the fields of autonomous driving, robotics, geomatics, and so on. Nevertheless, compared with 2D image processing, this task is quite challenging due to the non-grid structure and orderless permutation of point clouds. Furthermore, it is extremely difficult to perform shape analysis for point clouds, as the underlying shape formed by those discrete points is visually elusive to capture.

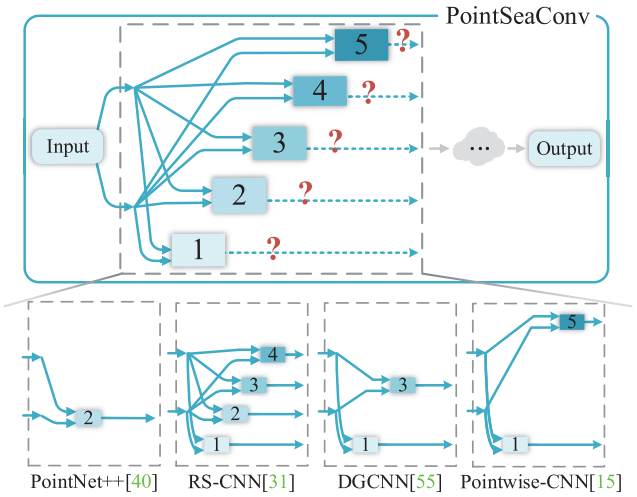


Figure 1. A sketch of the proposed convolution search paradigm, *i.e.*, PointSeaConv. PointSeaConv is achieved by constructing a dynamic and learnable directed acyclic graph (DAG). Most handcrafted convolutions can be regarded as its special cases.

In order to tackle this task, many geometric descriptors [56] have been manually designed over the past decades. Though usable in certain scenarios, these descriptors often suffer from unsatisfactory performance and poor task-adaptivity. Recently, convolutional neural networks (CNN) in deep learning technologies has made remarkable achievements in the processing of regular data, *e.g.*, image [22, 45], video [48], and speech [37]. Accordingly, there has been a growing interest in exploiting the power of CNN for irregular point cloud processing [33, 44, 58].

To facilitate the application of classic CNN, some researchers transform point clouds into regular multi-view images [13, 50] or voxel grids [42, 71]. While practicable, these transformations usually lead to the loss of shape information because of self-occlusions or quantization artifacts. As a pioneer, PointNet [43] learns directly on point

*Corresponding author.

clouds with shared multi-layer perceptrons (MLP) and max-pooling operation. Despite its encouraging performance on shape analysis, PointNet has difficulty in learning fine-grained shape representation due to the lack of local modeling. To overcome this issue, PointNet++ [44] builds an explicit local-to-global CNN-like architecture with multiple set abstraction layers. However, it performs convolution by simply applying PointNet on the local regions, which could be powerless in capturing diverse local structures.

Plenty of follow-up research, therefore, is devoted to manually design convolution variants, which is expected to grasp local structures well. The typical methods along this route are EdgeConv [58], PointCNN [29], RS-Conv [33], KPConv [54], PointConv [59], and so on. They either construct local graph connections collocated with graph convolution methods, or empirically introduce local geometric statistics (*e.g.*, density) into convolution operation. Though achieving decent performance, these convolutions greatly depend on heuristic rules and experienced engineering.

In this paper, we argue that the manually-designed convolution could be suboptimal for point cloud processing, especially in the era of data-driven deep learning. The key challenge for convolution learning on point clouds is how to make it being capable to sufficiently capture diverse local structures. This motivates us, accordingly, to construct an auto-created convolution search paradigm, which can be directly driven by irregular structures in point clouds.

To this end, we propose PointSeaConv, *i.e.*, a novel differentiable convolution search paradigm on point clouds. Concretely, we first formulate a general convolution for geometric structure modeling and transform it into a searchable process. This is achieved by constructing a dynamic and learnable directed acyclic graph (DAG). Consequently, the convolution expression can be determined by the DAG while the convolution weight can be learned on the DAG. Moreover, most handcrafted convolutions can be regarded as special cases of our searchable one (Fig. 1). We then develop a joint and differentiable optimization framework for optimizing the search of internal convolution and external architecture, simultaneously. Especially, the epsilon-greedy algorithm is introduced into the search process, which greatly alleviates the effect of discretization error. As a result, PointSeaNet, a deep network that captures geometric structures at both convolution level and architecture level, can be searched out for point cloud processing.

The key contributions can be summarized as follows:

- We propose a novel differentiable convolution search paradigm, *i.e.*, PointSeaConv. It can work in a purely data-driven manner and thus is capable of creating a group of suitable convolutions for point cloud processing. To our best knowledge, we are the first to conduct fundamental convolution search on point clouds.
- We propose a joint and differential optimization framework for simultaneous search of internal convolution and external architecture. Under the framework, PointSeaNet, a deep network that sufficiently captures geometric structures of point clouds at both convolution level and architecture level, can be searched out.
- We innovatively introduce epsilon-greedy algorithm into the search framework. Thanks to the algorithm, the adverse effect of discretization error can be greatly alleviated during the whole search process.

2. Related Work

2.1. Point Cloud Processing

In this section, we briefly review existing deep learning methods for for point cloud processing. According to the data type of input, these methods can be generally divided into projection-based networks and point-based networks.

Projection-based networks [50, 53, 65] project 3D point clouds into 2D multiple views from various angles. Despite of impressive performance, most of them suffer from information loss due to occluded surfaces and viewpoint selections. Alternatively, volumetric-based networks [12, 36, 49] convert point clouds into uniform 3D grids and then apply CNNs on the volumetric grids. The key criticisms of these methods are the heavy computational burden and loss of details. Unlike these methods, our work is able to directly process point clouds without any pre/post-processing step.

Point-based networks directly consume point cloud and become increasingly popular. Inspired by PointNet [43], much research has been devoted to elaborately designing sophisticated networks to learn pointwise local features. These methods can be generally classified as 1) pointwise MLP networks [1, 15, 44, 51, 70], 2) point convolution networks [3, 14, 33, 54, 59], 3) data indexing networks [20, 26, 46, 69]. However, these methods lack internal mechanisms to generate convolution operators according to local geometric structures. In contrast, our PointSeaNet can automatically search fundamental convolution operations driven by input point clouds.

2.2. Neural Architecture Search (NAS)

Neural architecture search (NAS) methods inherently aim to provide an automatic way of designing architectures to replace the manual ones. Early methods employ reinforcement learning [72, 73] and evolutionary algorithm [10, 61] to find the optimal architecture. Further, one shot approaches [4, 6, 7, 31] are proposed to reduce the computational costs by training the super-network only once, which is sampled and evaluated subsequently. The pioneering work DARTS [31] introduces a differentiable framework to relax the search space and hence improves the efficiency of search period. Most of them are elaborately designed

to tackle various 2D vision problems and have achieved superior performance [11, 40]. Recently, some approaches have focused on neural architecture search for irregular point cloud processing [28, 52, 68]. However, these methods heavily rely on fixed convolution operators, such as existing graph convolutions (*e.g.*, EdgeConv [58], GAT [57] and SemiGCN [19]) and pre-defined convolution kernels on 2D images, which results in incapability to sufficiently capture geometric structures for point clouds. Through our differentiable convolution search paradigm, by comparison, fundamental convolution operators collaboratively working with external architecture can be searched out.

3. Methodology

In this section, we first formulate a general convolution (Sec. 3.1), of which the image convolution can be seen as a special case. We then adapt this general convolution to learn geometric information in point clouds, by transforming it into a convolution search problem (Sec. 3.2). Finally, we show how the convolution search can be collocated with external architecture search in a joint and differentiable optimization manner (Sec. 3.3).

3.1. General Convolution Formulation

General convolution. The key properties of convolution are local connectivity and weight sharing (over different local regions) [25]. Technically, inside a local region, the convolution can be generally decomposed into two steps: (i) transforming the feature vector of each unit in this local region and (ii) aggregating all the transformed features for summarizing the local information. Formally, given a local region $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_n\}$ with n units, in which $\boldsymbol{p}_j \in \mathbb{R}^{F\times 1}$ denotes the feature vector of the j-th unit, then the general convolution with above two steps can be formulated as

$$\boldsymbol{p} = \mathcal{G}\big(\{\psi(\boldsymbol{p}_j)\}_{j=1,\dots,n}\big)^{1}, \tag{1}$$

where the output $\boldsymbol{p} \in \mathbb{R}^{F'\times 1}$ is obtained by a transformation function $\psi(\cdot)$ at step (i) and an aggregation function $\mathcal{G}(\cdot)$ at step (ii). In addition, $\psi(\cdot)$ is usually shared over different local regions to achieve weight sharing property.

Image convolution. Notably, the image convolution can be seen as a special case of this general convolution. To be specific, the local region in the image is arranged with a regular grid structure and all the units (*i.e.*, pixels) in this region are fixed and ordered. Thus the image convolution can be written as

$$\boldsymbol{p} = \sum_j \mathbf{W}_j \odot \boldsymbol{p}_j, \tag{2}$$

where $\mathbf{W}_j \in \mathbb{R}^{F'\times F}$ denotes the convolutional weight matrix for $\boldsymbol{p}_j$ and "$\odot$" indicates the matrix multiplication. That is, $\psi(\boldsymbol{p}_j)$ and $\mathcal{G}(\cdot)$ in Eq. (1) are implemented as $\mathbf{W}_j \odot \boldsymbol{p}_j$ and summation here, respectively. Moreover, note that the weight $\mathbf{W}$ is learned on pixel values, *i.e.*, $\boldsymbol{p}_j|_{j=1,\dots,n}$, hence it shows great power to capture semantic patterns reflected by color information.

[1] In this paper, we omit the bias term and activation function for clarity.

Essential association (EA)	Advantages
e_1: $\boldsymbol{n}_i$	global features [43]
e_2: $\boldsymbol{n}_j$	local features [3, 21, 27, 33, 44]
e_3: $\boldsymbol{n}_i - \boldsymbol{n}_j$	geometric relations [15, 29, 33, 47, 62]
e_4: $\|\|\boldsymbol{n}_i - \boldsymbol{n}_j\|\|_2$	Euclidean distance [15, 33, 38, 53, 54]
e_5: $\boldsymbol{n}_i - \sum_{\boldsymbol{n}_k\in\mathcal{N}(\boldsymbol{n}_i)} \frac{\boldsymbol{n}_k}{\lvert\mathcal{N}(\boldsymbol{n}_i)\rvert}$	salient information [16, 53]

Table 1. A summary of five essential association (EA) candidates. $|\mathcal{N}(\boldsymbol{n}_i)|$ indicates the number of all points in $\mathcal{N}(\boldsymbol{n}_i)$.

3.2. Convolution Search on Point Clouds

Existing challenges. Recently, the image convolution in Eq. (2) has been adapted by many researchers for transferring its great power in image processing into point cloud processing. However, this is in fact quite challenging. The reasons are twofold: (i) it is very intractable to achieve the permutation invariance to point set, whilst ensuring that the convolution is capable of sufficiently learning local structures; (ii) the weight sharing property of convolution is hard to implement due to the irregular structures (*i.e.*, variable number of points) over different local regions. Although these issues are partly alleviated by recent convolution variants [29, 33, 44, 58, 62], most of them are manually designed. Such handcrafted convolutions not only rely heavily on expert knowledge with long-term design cycle, but also show poor generalization in various scenarios [34].

Geometric modeling. In this paper, we argue that it could be suboptimal to manually design the convolution for point cloud processing. Hence we propose an entirely different route, *i.e.*, transforming the general convolution in Eq. (1) into a convolution search problem on point clouds. Formally, we model the points in a local region as the central point $\boldsymbol{p}_i$ and its surrounding neighbors $\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)$. Note that the shape information in point clouds is from relative spatial distribution among points. This is quite different from the 2D image, where meaningful information is from the value of pixels, not grid distribution. Accordingly, we propose to learn the shape information by learning the geometric associations between $\boldsymbol{p}_i$ and its neighbors $\mathcal{N}(\boldsymbol{p}_i)$. Thus, Eq. (1) becomes

$$\boldsymbol{p}'_i = \mathcal{G}\Big(\big\{\psi\big(\mathcal{D}(\boldsymbol{p}_i, \boldsymbol{p}_j)\big)\big\}_{\boldsymbol{p}_j\in\mathcal{N}(\boldsymbol{p}_i)}\Big), \tag{3}$$

where $\mathcal{D}(\boldsymbol{p}_i, \boldsymbol{p}_j)$ indicates the encoding function of the geometric association between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$. The convolutional output $\boldsymbol{p}'_i$ aggregates all the geometric associations between $\boldsymbol{p}_i$ and $\mathcal{N}(\boldsymbol{p}_i)$, thus it could show superior shape awareness.

Function: Searchable construction. The key problem for the convolution in Eq. (3) is how to design the concrete expressions of $\psi(\cdot)$ and $\mathcal{D}(\cdot,\cdot)$. Instead of the common hand-

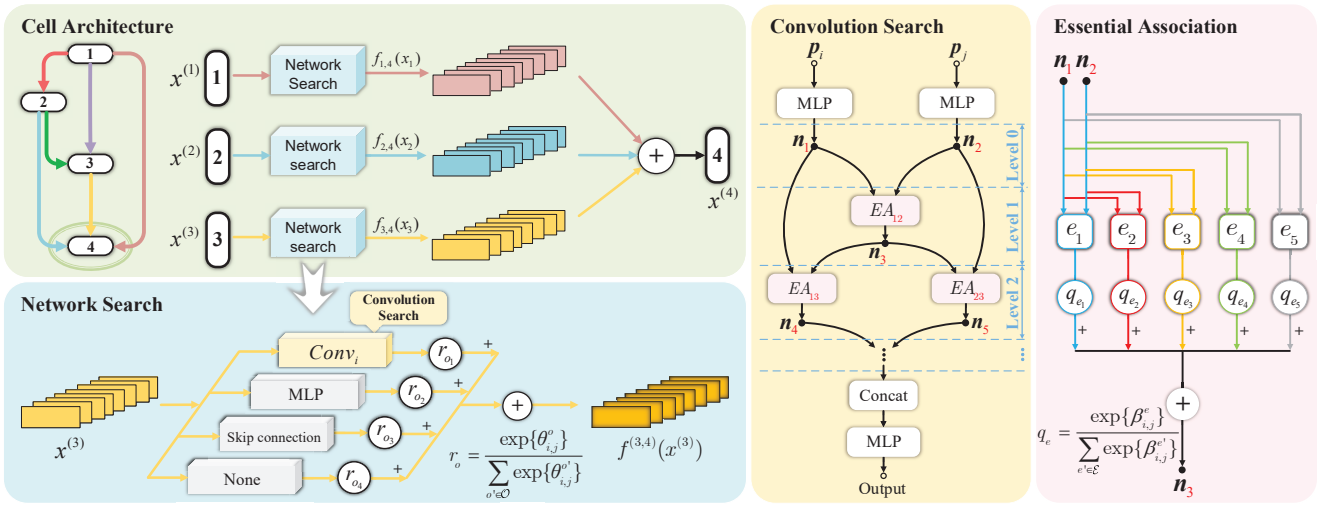


Figure 2. An overview of our PointSeaNet. For clarity, a cell architecture (the upper part of (a)) formed by four ordered nodes is illustrated. The search process of the whole network can be divided into convolution search and cell architecture search (*i.e.*, network search). The convolution search is achieved under our geometric convolution modeling in Eq. (3), which is transformed from the general convolution in Eq. (1). Technically, it is constructed with a multi-level directed acyclic graph (DAG, Sec. 3.2), in which a set of searchable essential associations (EA, Sec. 3.2) are conducted. The essential association is searched out from five fundamental candidates, *i.e.*, $e_1 \sim e_5$ in Tab. 1. Moreover. the proposed convolution search can be collocated with external network search in a joint and differentiable optimization manner (Sec. 3.3). As a result, a group of suitable convolutions can be searched out for point cloud processing in a purely data-driven manner. Note that the number of searchable convolutions, *i.e.*, $Conv_i$ in network search, is variable. Best viewed in color.

crafted manner, we transform this problem into a NAS-like search process. That is, $\mathcal{D}(\cdot,\cdot)$ is devoted to the construction of geometric association encoding while $\psi(\cdot)$ is responsible for all the learnable parameters on this construction.

Technically, we construct $\mathcal{D}(\cdot,\cdot)$ using a directed acyclic graph (DAG). As the example in Fig. 2(b) shows, our DAG can be represented as an ordered sequence of several hidden nodes. Each node is a feature vector and the i-th node is denoted as $\boldsymbol{n}_i$. To deeply encode the geometric association, a number of searchable essential associations (EA, and EA_{ij} indicates the association between $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$) are conducted on these hidden nodes. Like deep network with multiple layers, our DAG can also be constructed with multiple levels ℓ. Furthermore, it is noticeable that there must be a connection between any two nodes (except the final output nodes) in our DAG, and all the connections must go through the essential associations. Therefore, our DAG is capable of sufficiently encoding the geometric association between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$. As a result, the whole convolution with our DAG can be searched out in a purely data-driven manner. Note that this search process is different from S-GAS [28], which just manually selects seven existing graph convolutions (*e.g.*, EdgeConv [58], GAT [57] and SemiGCN [19]) as searchable convolution candidates.

Essential association (EA). The essential association is the core of our DAG. It has the function of transferring the key information from preceding nodes to the output nodes. Instead of elaborate manual design, we propose to search an optimal association for each EA from predefined association candidates. As summarized in Tab. 1, we define five fundamental association candidates after a full investigation in the field of point cloud processing. Due to the capability to learn structural relational features on multiple aspects, the five candidates provide informative enough search space for our DAG in terms of learning geometric associations between $\boldsymbol{p}_i$ and its neighbors $\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)$.

Parameterization: Searchable convolution. To learn our constructed searchable convolution, we group the learnable parameters in function $\psi(\cdot)$ (Eq. (3)) on the DAG into two parts, which are parameterized with f_β and h_γ. Concretely, f_β is responsible for the parameters in all the essential associations (EA), which actually determine the construction of DAG. h_γ is responsible for the parameters in all the multi-layer perceptrons (MLP) on the DAG. Thus the searchable convolution version of Eq. (3) can be written as

$$\begin{aligned} \boldsymbol{p}_i' &= \mathcal{G}\Big(\big\{h_\gamma\big(\mathcal{D}(\boldsymbol{p}_i,\boldsymbol{p}_j)\big)\big\}_{\boldsymbol{p}_j\in\mathcal{N}(\boldsymbol{p}_i)}\Big), \\ \mathcal{D}(\boldsymbol{p}_i,\boldsymbol{p}_j) &= f_\beta\big(\forall\,\text{EA}\in\mathcal{D}(\boldsymbol{p}_i,\boldsymbol{p}_j)\big). \end{aligned} \tag{4}$$

In this way, an optimal combination of essential associations can be searched out to create a suitable convolution, which is capable of sufficiently capturing diverse geometric structures of point clouds.

In implementation, on one hand, both f_β and h_γ are shared over each neighboring point $\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)$. Then, with a symmetric aggregation function $\mathcal{G}$, PointSeaConv

Method	$\mathcal{G}(\cdot)$	$\{h_\gamma(\mathcal{D}(\boldsymbol{p}_i, \boldsymbol{p}_j))\}_{\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)}$	Function expression in PointSeaConv
PointNet++ [44]	$\max(\cdot)$	$\mathrm{MLP}(e_2)$	$\max_{\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)}\{\mathrm{MLP}(e_2)\}$
PointWeb [70]	$\max(\sum\{\cdot\})$	$\mathrm{MLP}(e_1, e_3)$	$\max\{\sum_{\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)}\{\mathrm{MLP}(e_1, e_3)\}\}$
DGCNN [58]	$\max(\cdot)$	$\mathrm{MLP}(e_1, e_3)$	$\max_{\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)}\{\mathrm{MLP}(e_1, e_3)\}$
RS-CNN [33]	$\max(\cdot)$	$\mathrm{MLP}(e_1 \oplus e_2 \oplus e_3 \oplus e_4)$	$\max_{\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)}\{\mathrm{MLP}(e_1 \oplus e_2 \oplus e_3 \oplus e_4)\}$
Pointwise-CNN [16]	$\sum(\cdot)$	$\mathrm{MLP}(e_1 - e_5)$	$\sum_{\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)}\{\mathrm{MLP}(e_1 - e_5)\}$

Table 2. Several deep learning methods on point clouds can be derived as particular settings of PointSeaConv in Eq. (4), by appropriately selecting aggregation function and combinations of essential associations. The definition of $e_1 \sim e_5$ is shown in Tab. 1. max denotes max pooling and $\sum$ denotes summation.

can be permutation invariant to unordered points while be capable of capturing local structures sufficiently. On the other hand, we adopt k-nearest neighbor approach to acquire $\mathcal{N}(\boldsymbol{p}_i)$. Hence PointSeaConv can achieve the weight sharing property despite that different local regions are of irregular structures (*i.e.*, variable number of points).

In addition, our searchable convolution shows good generalization in point cloud processing. As summarized in Tab. 2, most recent convolutions can be seen as special cases of our searchable convolution. For example, DGCNN [58] can be implemented by configuring MLP with e_1 and e_3 in Tab. 1 to learn geometric associations.

3.3. Joint Differentiable Optimization Approach

Differentiable architecture search. Before introducing our approach, we first briefly review cell-based NAS methods [30, 41, 73]. This class of methods represent the architecture as a set of identical cells with different weights, which is represented by directed acyclic graphs (DAG) with an ordered series of nodes. Formally, $x^{(i)}$ denotes the output of the i-th node and (i, j) denotes a directed edge from the i-th node to j-th node. The candidate operations are denoted as $\mathcal{O}$, in which each element $o^{(i,j)}(\cdot)$ propagates the information from $x^{(i)}$ to $x^{(j)}$ across the edge (i, j). In differentiable architecture search methods [5, 31, 63], the continuous relaxation of candidate operations is conducted to obtain the optimal architecture. Consider continuous variables $\alpha = \left\{\alpha^{(i,j)}\right\}$ as architecture parameters for edge (i, j) and the network weights ω, the selection of candidate operations can be relaxed as a softmax mixture over all the possible operations within the operation space $\mathcal{O}$. Then, the output at j-th node is the sum of information flows from all its predecessors. Intrinsically, the goal of NAS is to derive the optimal architecture $\alpha^\star$ and network weights $\omega^\star(\alpha)$ associated with the architecture $\alpha^\star$ by solving the following bilevel optimization problem

$$\begin{aligned} \alpha^\star &= \arg\min_{\alpha} \mathcal{L}_{val}\left(\omega^\star(\alpha), \alpha\right), \\ \text{s.t.} \quad \omega^\star(\alpha) &= \arg\min_{\omega} \mathcal{L}_{train}\left(\omega, \alpha\right), \end{aligned} \tag{5}$$

where $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ indicate the training and validation loss, respectively. After the search process, the final architecture is derived by selecting the path with the highest architecture parameters.

Joint Optimization. To enable end-to-end training for convolution search, we perform architecture search for the optimal convolution and cell architecture simultaneously under the differentiable architecture search framework as in [5, 31, 63], denoted as *convolution search* and *network search*, respectively. Intuitively, the overall search framework is shown in Fig. 2, which takes a cell structure with 4 nodes and its connection from $x^{(i)}$ to $x^{(j)}$ as an example. Similar to the selection of candidate operations in cell structure search, we define five essential association candidates as search space in convolution search as shown in Tab. 1, denoted as $\mathcal{E}$. In our framework of joint optimization, in addition to the weights ω in the network, the whole architectural parameters are denoted as $\rho = \{\theta, \beta\}$, where θ and β indicate parameters of network search and convolution search, respectively. In network search, as the connection from $x^{(i)}$ to $x^{(j)}$, the output of $f^{(i,j)}\left(x^{(i)}\right)$ becomes

$$f^{(i,j)}(x^{(i)}) = \sum_{o \in \mathcal{O}} \frac{\exp\left\{\theta_o^{(i,j)}\right\}}{\sum_{o' \in \mathcal{O}} \exp\left\{\theta_{o'}^{(i,j)}\right\}} o\left(x^{(i)}\right). \tag{6}$$

In convolution search, given an input $\boldsymbol{n}_i$ and its neighbors $\boldsymbol{n}_j \in \mathcal{N}(\boldsymbol{n}_i)$ in a local neighborhood, the choice of a particular essential association can be relaxed to a softmax mixture in dimension $|\mathcal{E}|$

$$\bar{e}^{(i,j)}(\boldsymbol{n}_i, \boldsymbol{n}_j) = \sum_{e \in \mathcal{E}} \frac{\exp\left(\beta_e^{(i,j)}\right)}{\sum_{e' \in \mathcal{E}} \exp\left(\beta_{e'}^{(i,j)}\right)} e(\boldsymbol{n}_i, \boldsymbol{n}_j). \tag{7}$$

Accordingly, the tasks of convolution search and network search can be summarized to learn a set of parameters $\rho = \{\theta, \beta\}$. This bilevel optimization process can be described to updated ρ and ω alternately

$$\begin{aligned} \omega_{t+1} &\leftarrow \omega_t - \eta_\omega \cdot \nabla_\omega \mathcal{L}_{val}\left(\omega_t, \rho_t\right), \\ \rho_{t+1} &\leftarrow \rho_t - \eta_\rho \cdot \nabla_\rho \mathcal{L}_{train}\left(\omega_{t+1}, \rho_t\right), \end{aligned} \tag{8}$$

where η_ω and η_ρ denote the learning rates for ω and ρ, respectively. For simplicity, we incorporate the parameters of

MLP in convolution (denoted by γ in Eq. (4)) into network weights ω.

Notably, the discrete architecture for network search is obtained by retaining each operation with the highest weight, $f^{(i,j)}\left(x^{(i)}\right) = \text{argmax}_{o\in\mathcal{O}}\,\theta_o^{(i,j)}$. With respect to convolution search, we will introduce epsilon-greedy algorithm to reduce the discretization error in the following.

Epsilon-greedy Algorithm. As pointed in [5, 9, 55], the optimization method in DARTS [31] leads to a large discretization error after the search process due to deleting substantial candidate operations with moderate weights. Note that, since convolution search and network search are conducted simultaneously in our method, the risk of discretization error further grows.

To alleviate the discretization error and its accumulation during the search process, we introduce the epsilon-greedy algorithm for efficient optimization of convolution search. First, we make essential association candidates fixed in each step of optimizing the weights ω, where each of them is discretized by selecting the strongest one using greedy algorithm, denoted as $\hat{\beta}_e^{(i,j)} = \max_{e'\in\mathcal{E}} \beta_{e'}^{(i,j)}$. Only in the stage of optimizing $\beta_e^{(i,j)}$, all the choices of essential association candidates are relaxed. Further, in order to avoid removing all the moderate candidates and reduce the dependence on the parameter initialization, the essential association candidates with the highest weight are selected by a certain probability ε, so that the optimization process of convolution search can be described as

$$\begin{cases} P(\beta_e^{(i,j)} = \hat{\beta}_e^{(i,j)}) = 1-\varepsilon \\ P(\beta_e^{(i,j)} = \beta_{\text{random}}) = \varepsilon \end{cases}, \tag{9}$$

where $P(\cdot)$ is a probability distribution of $\beta_e^{(i,j)}$, β_{random} is a random one-hot vector, and ε is a hyper-parameter to balance greedy algorithm and random algorithm. Instead of eliminating all weak candidates, epsilon-greedy algorithm retains more candidates that can contribute more or less to training accuracy. In this way, dramatic improvements are achieved by our PointSeaNet on multiple tasks. Detailed settings and analyses are provided in the Sec. 4.

4. Experiment

In this section, we conduct comprehensive experiments to demonstrate the capability of PointSeaNet. We first briefly introduce some experimental settings (Sec. 4.1). Then, we systematically evaluate PointSeaNet on challenging benchmarks across various point cloud understanding tasks (Sec. 4.2). Finally, we provide detailed ablation studies (Sec. 4.3) to validate PointSeaNet thoroughly.

4.1. Experimental Setting

The cell architecture has 5 candidate operations: two PointSeaConv, MLP, *skip-connection* and *zero* operation.

Method	OA	#params	Search Cost
Pointwise-CNN [16]	86.1	-	manual
PointNet [43]	89.2	3.48	manual
PointNet++ [44]	90.7	1.48	manual
PointCNN [29]	92.2	0.45	manual
DGCNN [58]	92.2	1.84	manual
PCNN [3]	92.3	8.10	manual
PointASNL [64]	92.9	-	manual
InterpCNN [35]	93.0	12.8	manual
GeoCNN [23]	93.4	-	manual
RS-CNN [33]	<u>93.6</u>	-	manual
SGAS [28]	93.2	8.49	0.19
PointSeaNet†	94.0	6.70	0.23
PointSeaNet	**94.2**	6.75	0.25

Table 3. Shape classification results (OA: overall accuracy) on ModelNet40.

Dataset	Method	#points	mAP(%)
ModelNet40	PointNet [43]	1k	70.5
	PointCNN [29]	1k	83.8
	DGCNN [58]	1k	85.3
	Densepoint [32]	1k	<u>88.5</u>
	PointSeaNet†	1k	89.9
	PointSeaNet	1k	**90.3**

Table 4. Shape retrieval results (mAP, %) on ModelNet40.

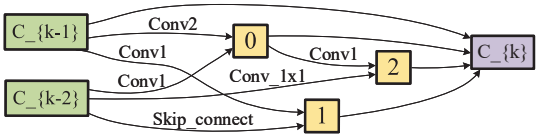
Each PointSeaConv has 3 levels with 5 nodes. Neighboring points are firstly gathered by *k nearest neighbor* in the first operation of each cell. PointSeaNet is obtained through two stages, a search phase and an evaluation phase. More details are provided in the supplementary material. Furthermore, PointSeaNet† that omits epsilon-greedy algorithm in PointSeaNet is employed as a baseline of our model.

4.2. PointSeaNet for Point Cloud Processing

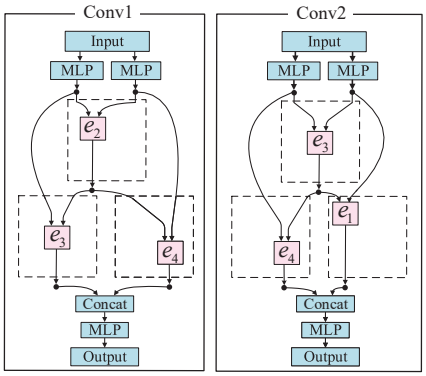
Shape classification. We conduct architecture search and evaluation on ModelNet10 and ModelNet40 classification benchmarks [60], respectively. The former contains 3991 training models and 908 test models in 10 classes, and the latter consists of 9843 training models and 2468 test models in 40 classes. 1024 points are uniformly sampled by farthest point sampling. During training, we augment the input data with random anisotropic scaling and translation as in [20]. During testing, similar to [43, 44], we conduct ten voting tests with random scaling and average the predictions. Additionally, we do not use normals as additional input.

The quantitative comparisons with the other advanced methods are shown in Tab. 3. Our PointSeaNet outperforms all the other methods, while using only a search cost of 0.23 GPU day on one NVIDIA TITAN Xp. Compared with SGAS [28], which conducts architecture search with fixed graph convolutions, PointSeaNet reduces the model params by 20.5% and improve the overall accuracy by 1.0%. We visualize the searched optimal cell architecture and convolution on ModelNet10 in Fig. 3.

Shape retrieval. To further explore the recognition a-



(a) Cell architecture



(b) Convolution

Figure 3. The best architecture and convolution on ModelNet10.

Method	Input	Class mIou	Instance mIou
Kd-Net [20]	4k	77.4	82.3
PointNet [43]	2k	80.4	83.7
PCNN [3]	2k	81.8	85.1
PointNet++ [44]	2k, nor	81.9	85.1
SyncCNN [67]	mesh	82.0	84.7
SPLATNet [49]	-	82.0	84.6
DGCNN [58]	2k	82.3	85.1
RS-CNN [33]	2k	84.0	86.2
Densepoint [32]	2k	84.2	86.4
PointSeaNet†	2k	85.2	87.3
PointSeaNet	2k	**85.7**	**87.8**

Table 5. Shape part segmentation results (%) on ShapeNetPart (nor: normal, '-': unknown).

bility of PointSeaNet, we conduct 3D shape retrieval on ModelNet40. Specifically, we employ the outputs of the penultimate fully-connected layer for shape classification as the global features. We evaluate PointSeaNet on ModelNet40 for this task and uniformly sample 1024 points as the input. The cosine distance is applied to obtain relative ranking order of each query shapes from the test set. We report *mean Average Precision* (mAP). Tab. 4 shows the results for the shape retrieval task, where PointSeaNet achieves the best performance with mAP of 90.3% on ModelNet40. Note that, our PointSeaNet surpasses its variant, *i.e.*, PointSeaNet†, with 0.4% ↑ in mAP.

Shape part segmentation. For this task, we search the optimal convolution and cell architecture using stacked identical cells on the ShapeNetPart benchmark [66], and then the searched cell is stacked to form a larger network, which is retrained on ShapeNetPart. ShapeNetPart consists of 16881 shapes with 16 categories, which is labeled in 50 parts in total. Following [43], we randomly sample 2048 points as

Method	Area-5		6-fold	
	OA	mIoU	OA	mIoU
PointNet [43]	-	41.1	78.6	47.6
PointNet++ [44]	-	-	81.0	54.5
DGCNN [58]	-	-	84.1	56.1
PointCNN [29]	85.9	57.3	88.1	65.4
LSANet [8]	-	-	86.8	62.2
SPG [24]	86.4	58.0	85.5	62.1
RandLA-Net [15]	-	-	87.2	68.5
PAG [39]	86.8	59.3	88.1	65.9
PointSIFT [18]	-	-	88.7	70.2
Pointweb [70]	87.0	60.3	87.3	66.7
HPEIN [17]	87.2	61.9	88.2	67.8
KPConv [54]	-	67.1	-	70.6
PointSeaNet†	88.1	68.2	89.6	71.2
PointSeaNet	**89.2**	**69.0**	**90.3**	**71.9**

Table 6. Scene segmentation results (%) on S3DIS ('-': unknown).

the input and concatenate the one-hot encoding of the object label into the last feature layer. During testing, we also perform ten voting tests using random scaling. Evaluation metrics contain two types of mIoU that are averaged across all classes and all instances respectively. Tab. 5 gives the results in this experiment, where PointSeaNet outperforms the best handcrafted method Densepoint [32] with 1.5 ↑ in class mIoU and 1.4 ↑ in instance mIoU respectively. The dramatic improvements validate the capability of our method to learn fine-grained features. We detail each class mIoU and result visualizations in the supplementary material.

Large-scale scene segmentation. In this experiment, we perform 3D scene segmentation to evaluate our PointSeaNet on the S3DIS [2] benchmark. As a large-scale public dataset, the S3DIS dataset contains 271 million points belonging to 6 large-scale indoor areas with 13 classes. We conduct the optimal convolution and cell architecture search on S3DIS, and provide more details in the supplementary material. To adequately measure the generalization ability of our PointSeaNet, we adopt both Area-5 and standard 6-fold cross validation as test setting. As shown in Tab. 6, our PointSeaNet outperforms other state-of-the-art methods. Notably, PointSeaNet can achieve a superior results compared with its variant PointSeaNet†, with 0.7 ↑ in overall accuracy and 0.7 ↑ in mIoU with the standard 6-fold cross validation as test setting.

Normal estimation. We evaluate PointSeaNet with the same parameters as in the shape part segmentation. The optimal architecture are searched on ModelNet40, and then a larger network composed of searched cells is evaluated with normal estimation as a supervised regression task on ModelNet40. 1024 points are uniformly sampled as the input. The cosine-loss between the normalized output and the normal ground truth is used to train PointSeaNet. The results in Tab. 7 show that PointSeaNet outperforms all the compared methods with a lower error of 0.10, which significantly reduces the error of RS-CNN (0.15) by 33.3%. Some result

Dataset	Method	#points	error
ModelNet40	PointNet [43]	1k	0.47
	PointNet++ [44]	1k	0.29
	PCNN [3]	1k	0.19
	MC-Conv [14]	1k	0.16
	RS-CNN [33]	1k	0.15
	Densepoint [32]	1k	0.15
	PointSeaNet†	1k	0.12
	PointSeaNet	1k	**0.10**

Table 7. Normal estimation error on ModelNet40.

visualizations are provided in the supplementary material.

4.3. Ablation study

Sensitivity to hyperparameters. We conduct experiments to evaluate the sensitivity of our method to hyperparameters, *i.e.*, upon different settings of the number of cells, PointSeaConv and DAG levels. As shown in Tab. 8, PointSeaNet can get a decent accuracy of 93.7% with only 3 cells, 2 PointSeaConv and 2 DAG levels. Note that, our PointSeaNet with 2 DAG levels, 2 PointseaConv and 6 cells achieves the best performance, instead of the version with the largest amount of parameters. This clearly indicates that deeper level can improve performance to some extent, yet the success of PointSeaNet does not entirely come from introducing more parameters.

Analysis of essential associations. We experiment on ModelNet40 for shape classification to evaluate the searched architecture, to analyse the five essential associations (Tab.1). The results in Tab. 9 show that the baseline (model A) gets a low accuracy of 81.3%, which is set to architecture search with only e_1. Yet with local features denoted by e_2, it is significantly improved to 87.1% (model B), which shows that local features are crucial for improve performance. Then, when using geometric relations e_3 to enhance the representation ability of PointSeaNet, the accuracy can be further improved to 90.2% (model D). Noticeably, Euclidean distance e_4 can bring a boost of 2.7% (model E). Finally, the salient information e_5 can result in an accuracy variation of 1.1% (model F).

Effectiveness of convolution search and epsilon-greedy algorithm. We provide a detailed analysis to better understand the contributions of PointSeaNet. As can be seen in Tab. 3, even without the epsilon-greedy algorithm, PointSeaNet† can also achieve a superior result (94.0%) compared with the state-of-the-art handcrafted method RS-CNN [33] (93.6%) and the best point-cloud-NAS method SGAS [28] (93.2%). Though equipped with differentiable architecture search framework, SGAS adpots the existing graph convolutions, leading to restrict its capability to sufficiently capture geometric structures on point clouds. This adequately validate the effectiveness of our searchable convolution. Furthermore, the epsilon-greedy algorithm we introduce into NAS can significantly boost performance for various point cloud analysis tasks. As shown in Tab. 4, PointSeaNet surpasses its variant that omits the epsilon-greedy algorithm, *i.e.*, PointSeaNet†, with 0.4 ↑ in mAP on ModelNet40 shape retrieval. Regarding large-scale scene segmentation on S3DIS, the results in Tab. 6 show that PointSeaNet brings 0.9% overall accuracy gains and 0.8% mIoU gains over PointSeaNet† with Area-5 as test scene.

# Cells	# PointSeaConv	# DAG levels	# params(M)	OA(%)
3	2	2	4.15	93.7
6	2	2	6.75	**94.2**
9	2	2	9.36	94.1
6	1	2	6.26	93.8
6	2	2	6.75	**94.2**
6	3	2	6.95	94.0
6	2	1	6.53	93.5
6	2	2	6.75	**94.2**
6	2	3	8.72	93.9

Table 8. The comparisons of different number of cells, PointSeaConv and DAG levels during the evaluation phase.

Model	e_1	e_2	e_3	e_4	e_5	ϵ-greedy	OA(%)
A	✓						81.3
B	✓	✓					87.1
C	✓		✓				88.9
D	✓	✓	✓				90.2
E	✓	✓	✓	✓			92.9
F	✓	✓	✓	✓	✓		94.0
G	✓	✓	✓	✓	✓	✓	**94.2**

Table 9. The comparisons of choices of epsilon-greedy algorithm and several essential associations during the evaluation phase. The definition of $e_1 \sim e_5$ is shown in Tab. 1.

5. Conclusion

In this work, we present PointSeaConv, a differentiable convolution search paradigm that operates on point clouds. PointSeaConv is capable of creating an optimal convolution to sufficiently learn local structural features in a purely data-driven manner. For this purpose, a dynamic and learnable directed acyclic graph (DAG) is constructed to represent the whole convolution. Then a joint and differentiable optimization framework is developed to search for core convolution and external architecture. Meanwhile, by incorporating epsilon-greedy algorithm into convolution search, the discretization error is sharply alleviated during the search process, resulting in remarkably better performance.

6. Acknowledge

This research was supported by the National Key Research and Development Program of China under Grant No.2018AAA0100400, and the National Natural Science Foundation of China under Grants 62076242, 62071466, 61976208, 91838303 and 61972394.

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 2

[2] I. Armeni, O. Sener, A.R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, 2016. 7

[3] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *ACM TOG*, 37(4):1–12, 2018. 2, 3, 6, 7, 8

[4] G. Bender, P.J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018. 2

[5] K. Bi, L. Xie, X. Chen, L. Wei, and Q. Tian. Gold-nas: Gradual, one-level, differentiable. *arXiv preprint arXiv:2007.03331*, 2020. 5, 6

[6] A. Brock, T. Lim, J.M. Ritchie, and N. Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017. 2

[7] H. Cai, L. Zhu, and S. Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 2

[8] L. Chen, X. Li, D. Fan, K. Wang, S. Lu, and M. Cheng. Lsanet: Feature learning on point sets by local spatial aware layer. *arXiv preprint arXiv:1905.05442*, 2019. 7

[9] X. Chen, L. Xie, J. Wu, and Q. Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019. 6

[10] Y. Chen, G. Meng, Q. Zhang, S. Xiang, C. Huang, L. Mu, and X. Wang. Renas: Reinforced evolutionary neural architecture search. In *CVPR*, 2019. 2

[11] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun. Detnas: Backbone search for object detection. In *NeurIPS*, 2019. 3

[12] B. Graham, M. Engelcke, and L. Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[13] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE TPAMI*, 2020. 1

[14] P. Hermosilla, T. Ritschel, P. Vázquez, À. Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM TOG*, 37(6):1–12, 2018. 2, 8

[15] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020. 2, 3, 7

[16] B.S. Hua, M.K. Tran, and S.K. Yeung. Pointwise convolutional neural networks. In *CVPR*, 2018. 3, 5, 6

[17] L. Jiang, H. Zhao, S. Liu, X. Shen, C. Fu, and J. Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *ICCV*, 2019. 7

[18] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018. 7

[19] T.N Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 3, 4

[20] R. Klokov and V. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *ICCV*, 2017. 2, 6, 7

[21] A. Komarichev, Z. Zhong, and J. Hua. A-cnn: Annularly convolutional neural networks on point clouds. In *CVPR*, 2019. 3

[22] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1

[23] S. Lan, R. Yu, G. Yu, and L.S. Davis. Modeling local geometric structure of 3d point clouds using geo-cnn. In *CVPR*, 2019. 6

[24] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 7

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 3

[26] H. Lei, N. Akhtar, and A. Mian. Octree guided cnn with spherical kernels for 3d point clouds. In *CVPR*, 2019. 2

[27] G. Li, M. Muller, A. Thabet, and B. Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, 2019. 3

[28] G. Li, G. Qian, I.C. Delgadillo, M. Muller, A. Thabet, and B. Ghanem. Sgas: Sequential greedy architecture search. In *CVPR*, 2020. 3, 4, 6, 8

[29] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018. 2, 3, 6, 7

[30] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *ECCV*, 2018. 5

[31] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2, 5, 6

[32] Y. Liu, B. Fan, G. Meng, J. Lu, S. Xiang, and C. Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *ICCV*, 2019. 6, 7, 8

[33] Y. Liu, B. Fan, S. Xiang, and C. Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 1, 2, 3, 5, 6, 7, 8

[34] Z. Liu, H. Hu, Y. Cao, Z. Zhang, and X. Tong. A closer look at local aggregation operators in point cloud analysis. In *ECCV*, 2020. 3

[35] J. Mao, X. Wang, and H. Li. Interpolated convolutional networks for 3d point cloud understanding. In *ICCV*, 2019. 6

[36] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015. 2

[37] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 1

[38] G. Pan, J. Wang, R. Ying, and P. Liu. 3dti-net: Learn inner transform invariant 3d geometry features using dynamic gcn. *arXiv preprint arXiv:1812.06254*, 2018. 3

[39] L. Pan, C. Chew, and G.H. Lee. Pointatrousgraph: Deep hierarchical encoder-decoder with point atrous convolution for unorganized 3d points. In *ICRA*, 2020. 7

[40] J. Peng, M. Sun, Z. Zhang, T. Tan, and J. Yan. Efficient neural architecture transformation search in channel-level for object detection. In *NeurIPS*, 2019. 3

[41] H. Pham, M.Y. Guan, B. Zoph, Q.V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 5

[42] C.R. Qi, O. Litany, K. He, and L.J. Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019. 1

[43] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 2, 3, 6, 7, 8

[44] C.R. Qi, L. Yi, H. Su, and L.J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 1, 2, 3, 5, 6, 7, 8

[45] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1

[46] G. Riegler, A. Osman Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017. 2

[47] M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017. 3

[48] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[49] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M. Yang, and J. Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *CVPR*, 2018. 2, 7

[50] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015. 1, 2

[51] X. Sun, Z. Lian, and J. Xiao. Srinet: Learning strictly rotation-invariant representations for point cloud classification and segmentation. In *ACM MM*, 2019. 2

[52] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*, 2020. 3

[53] M. Tatarchenko, J. Park, V. Koltun, and Q. Zhou. Tangent convolutions for dense prediction in 3d. In *CVPR*, 2018. 2, 3

[54] H. Thomas, C.R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L.J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 2, 3, 7

[55] Y. Tian, C. Liu, L. Xie, J. Jiao, and Q. Ye. Discretization-aware architecture search. *arXiv preprint arXiv:2007.03154*, 2020. 6

[56] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *ECCV*, 2010. 1

[57] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 3, 4

[58] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, and J.M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 38(5):1–12, 2019. 1, 2, 3, 4, 5, 6, 7

[59] W. Wu, Z. Qi, and L. Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019. 2

[60] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 6

[61] L. Xie and A. Yuille. Genetic cnn. In *ICCV*, 2017. 2

[62] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 2018. 3

[63] Y. Xu, L. Xie, X. Zhang, X. Chen, G. Qi, Q. Tian, and H. Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *arXiv preprint arXiv:1907.05737*, 2019. 5

[64] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, 2020. 6

[65] Z. Yang and L. Wang. Learning relationships for multi-view 3d object recognition. In *ICCV*, 2019. 2

[66] L. Yi, V.G. Kim, D. Ceylan, I.C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM TOG*, 35(6):1–12, 2016. 7

[67] L. Yi, H. Su, X. Guo, and L.J. Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, 2017. 7

[68] Q. Yu, D. Yang, H. Roth, Y. Bai, Y. Zhang, A.L. Yuille, and D. Xu. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In *CVPR*, 2020. 3

[69] W. Zeng and T. Gevers. 3dcontextnet: Kd tree guided hierarchical learning of point clouds using local and global contextual cues. In *ECCV*, 2018. 2

[70] H. Zhao, L. Jiang, C. Fu, and J. Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 2, 5, 7

[71] Y. Zhou and O Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 1

[72] B. Zoph and Q.V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2

[73] B. Zoph, V. Vasudevan, J. Shlens, and Q.V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 2, 5