Training-free NAS for 3D Point Cloud Processing

Ping Zhao, Panyue Chen, and Guanming Liu

Tongji University, Shanghai, China
{zhaoping,chenpanyue,2130776}@tongji.edu.cn

Abstract. Deep neural networks for 3D point cloud processing have exhibited superior performance on many tasks. However, the structure and computational complexity of existing networks are relatively fixed, which makes it difficult for them to be flexibly applied to devices with different computational constraints. Instead of manually designing the network structure for each specific device, in this paper, we propose a novel training-free neural architecture search algorithm which can quickly sample network structures that satisfy the computational constraints of various devices. Specifically, we design a cell-based search space that contains a large number of latent network structures. The computational complexity of these structures varies within a wide range to meet the needs of different devices. We also propose a multi-objective evolutionary search algorithm. This algorithm scores the candidate network structures in the search space based on multiple training-free proxies, encourages high-scoring networks to evolve, and gradually eliminates low-scoring networks, so as to search for the optimal network structure. Because the calculation of training-free proxies is very efficient, the whole algorithm can be completed in a short time. Experiments on 3D point cloud classification and part segmentation demonstrate the effectiveness of our method[1].

Keywords: 3D Point Cloud Processing · Training-free Proxies · Neural Architecture Search.

1 Introduction

Deep neural networks (DNNs) for 3D point cloud processing have achieved superior performance on many tasks and have received more and more extensive attention. However, most of the existing DNNs rely on researchers to manually design the network structures, which places high demands on the researchers' experience and expertise. In addition, the structure and computational complexity of existing networks are relatively fixed, which makes it difficult for them to be flexibly applied to devices with different computational constraints. For example, complex networks [20,30] in academia often cannot be deployed on mobile devices. This restriction creates a split between academia and industry, limiting the development of downstream tasks such as autonomous driving [12,27]

[1] Codes will be available.

Table 1. Two categories of existing training-free proxies. We classify them according to the properties that they can reflect.

Pruning-Based Proxies for Network Trainability	**Gradnorm** [1]: Euclidean norm of the parameter gradients.
	Synflow [23]: absolute Hadamard product of the parameter gradients and parameters.
Linear-Region-Based Proxies for Network Expressivity	**Naswot** [16]: Hamming distance of binary activation patterns between mini-batch samples.
	Zen-score [13]: gradients of deep layer feature maps to mini-batch samples.

and robotics [22]. To address the above issues, some studies [9,8,14] proposed to use one-shot neural architecture search (NAS) to quickly sample structures. However, these methods require careful training of a relatively large supernet and suffer from the degenerate search-evaluation correlation problem [8], i.e., the performance of the sampled network at search time does not match its performance at evaluation time. This problem further limits the development of one-shot NAS methods.

In this paper, we focus on bridging the gap between academia and industry for different needs of deep neural networks. Specifically, we propose a novel training-free neural architecture search algorithm which can quickly sample network structures that satisfy the computational constraints of various devices. We first design a cell-based search space in which the network structure has dynamic layer number and feature dimensions. The computational complexity of structures in this search space varies within a wide range to meet the needs of different devices, e.g., we can sample very complex structures for cloud device, relative simple structures for mobile device. Then we propose a multi-objective evolutionary search algorithm. This algorithm scores the candidate network structures in the search space based on multiple training-free proxies, encourages high-scoring networks to evolve, and gradually eliminates low-scoring networks, so as to search for the optimal network structure.

Training-free proxies are numerical metrics that can evaluate the performance of a neural network before training. Previous studies [1,13,16] usually use a single proxy to guide the search. However, a single proxy can only reflect a single property[2] of the network structure, cannot fully reflect different properties of a complex network structure. Based on such single proxy, we can only encounter a small part of network structures in the search space. Therefore, in this paper, we propose to use multiple training-free proxies jointly. The objective of our search algorithm is to find network structures that perform well across multiple training-free proxies. We believe that in this way similar to ensemble learning, the

[2] A network has various properties, e.g., trainability (how effective a network can be optimized via gradient descent), expressivity (how complex the function a network can represent).

search algorithm can sample more potential structures from the search space. To jointly use proxies that reflect various properties, we divide them into different categories, as shown in Table 1.

To sum up, there are two stages in our method. In the search stage, based on multiple training-free proxies, we use an evolutionary search algorithm to find the potential network structures under a specific computational constraints in the designed search space. In the evaluation stage. We train these structures from scratch on different tasks to obtain their final performance. Experiments on 3D point cloud classification and part segmentation show that, compared with previous methods, our method can find better network structures under different computational constraints, which makes our method widely applicable to various devices. Furthermore, compared to traditional NAS methods, our search stage only takes hours, which further reduces the usage bottleneck of our method.

Our contributions are as follows:

1. We design a novel search space adapted to training-free proxies for 3D point cloud processing neural networks. This search space allows us to explore potential structures with varying amounts of computation;
2. We propose a multi-objective evolutionary search algorithm, which enables us to search with multiple proxies in an ensemble learning manner.
3. Our method largely bridges the gap between academia and industry, experiments on 3D point cloud classification and part segmentation demonstrate the effectiveness of our method.

2 Related Works

2.1 Point Cloud Processing

PointNet [18] first proposes to use a multi-layer perceptron with shared weights to process each point and use a symmetric aggregation function to obtain the global feature. PointNet++ [19] uses ball query for each point to obtain features of its neighbor points. PointCNN [11] aligns the points in a certain order by predicting the transformation matrix of the local point set. PCNN [2] proposes a parameterized continuous convolution operation. DGCNN [28] uses features from k-Nearest Neighbors (k-NN) and proposes an edge convolution operator for feature extraction. AdaptConv [35] proposes to jointly use feature relationship and coordinate relationship to achieve efficient graph convolution. These studies have proved the effectiveness of deep neural networks in 3D point cloud processing. However, they have relatively fixed network structures, which makes them difficult to be flexibly applied to devices with different computational constraints.

2.2 Training-free Neural Architecture Search

Training-free neural architecture search algorithms first come from network pruning. Some studies [7,26,23] try to prune networks in the initialization stage. SNIP

[7] proposes an importance metric to approximate the change of network loss after a specific parameter is removed. SynFlow [23] proposes `synflow` metric that can avoid layer collapse during pruning. Abdelfattah et al. [1] extend these indicators to neural architecture search and use them to evaluate the performance of the whole neural network. Other studies [16,13] are motivated by recent theory advances in deep networks and can also be calculated before network training. NASWOT [16] calculates Hamming distance of binary activation patterns between mini-batch samples to estimate linear regions of RELU networks. Zen-NAS [13] proves that the expressivity of the network can be efficiently measured by its expected Gaussian complexity. These works have achieved competitive performance. However, few of them use multiple training-free proxies jointly. In addition, they are all applied to 2D image classification and have not been verified on more tasks such as 3D point cloud classification and segmentation.

2.3 Neural Architecture Search in 3D Point Cloud Processing

Some studies [9,14,24,24,8,17] apply one-shot NAS to 3D point cloud processing and achieve relatively good performance. LC-NAS [9] implements a latency constraint formulation to trade-off between accuracy and latency in 3D NAS. PolyConv [14] introduces a poly-convolutional feature encoder that comprises multiple feature aggregation functions. SPVNAS [24] implements a sparse point-voxel convolution network to effectively process large-scale 3D scenes. SGAS [8] proposes a greedy algorithm and designs multiple numerical metrics such as edge importance for efficient structure selection. PointSeaNet [17] proposes a differentiable convolution search paradigm to create a group of suitable convolutions for 3D point cloud processing. In contrast to these one-shot NAS studies, our training-free method does not require training a relatively large supernet and does not suffer from the degenerate search-evaluation correlation problem.

3 Methods

3.1 Preliminaries and Notations

To quickly sample network structures that satisfy the computational constraints of various devices, we propose a novel training-free neural architecture search algorithm. In section 3.2, we analyze the existing training-free proxies and classify them based on the properties they can reflect. In section 3.3, we propose a search space for 3D point cloud processing network. In section 3.4, we introduce a novel multi-objective evolutionary search algorithm based on multiple training-free proxies.

Let $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ denote a minibatch point cloud data, $\boldsymbol{x}_i \in \mathbb{R}^{3\times G}$ denote a point cloud with G points, $\boldsymbol{\theta}$ denote network parameters which is initialized by a Gaussian distribution $\mathcal{N}(0,1)$, $\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{\theta})$ denote network loss to the input $\boldsymbol{x}_i$.

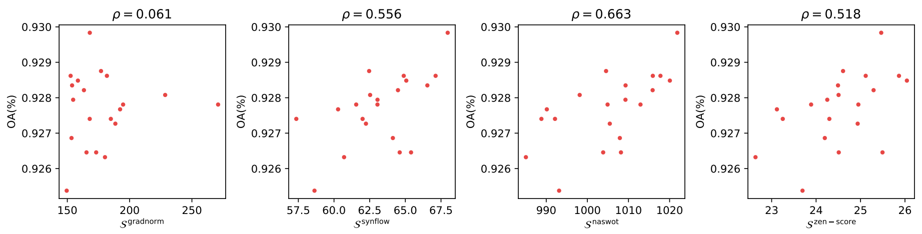

Fig. 1. Correlation between the score of different training-free proxies (horizontal axis) and the overall accuracy (vertical axis). ρ stands for the Pearson coefficient.

3.2 Training-free Proxies

We divide existing training-free proxies into two categories according to the properties that they can reflect: (1) Pruning-based proxies for trainability of neural networks. These kinds of proxies come from network pruning studies and indicate the performance of the neural network by calculating an indicator related to network parameter gradient at initialization; (2) Linear-region-based proxies for the expressivity of neural network. The linear region [4,31] is an essential indicator of the expressivity of the RELU neural network. These kinds of proxies indicate the performance of the network by estimating the number of linear regions of neural networks.

(1) Pruning-Based Proxies for Trainability

Gradnorm [1] is one of the simplest proxies. It sums the Euclidean norm of the parameter gradients after a single backward of minibatch data and uses this proxy to represent the trainability of a network.

$$\texttt{gradnorm} : \mathcal{S}^{\text{gradnorm}} = \sum_{\theta} \left\| \frac{\partial \mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta})}{\partial \theta} \right\| \tag{1}$$

Synflow [23,1] proposes iterative magnitude pruning to avoid layer-collapse in premature pruning. Formally, it is the absolute Hadamard product of the gradient of loss to each parameter and the parameter itself in a single minibatch. Following [1], when calculating `synflow`, we compute a loss which is simply the product of all parameters in the network, and sum up `synflow` of each parameter to represent the trainability of a network.

$$\texttt{synflow} : \mathcal{S}^{\text{synflow}} = \sum_{\theta} \left| \frac{\partial \mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta})}{\partial \theta} \odot \theta \right| \tag{2}$$

(2) Linear-Region-Based Proxies for Expressivity

Naswot [16] estimates the linear regions of RELU networks by calculating the Hamming distance of binary activation patterns between mini-batch samples. Suppose there are M rectified linear units in the network. For a specific input $\boldsymbol{x}_i$, we can arrange its activation patterns into a pattern string $\boldsymbol{p}_i$. In this way, the

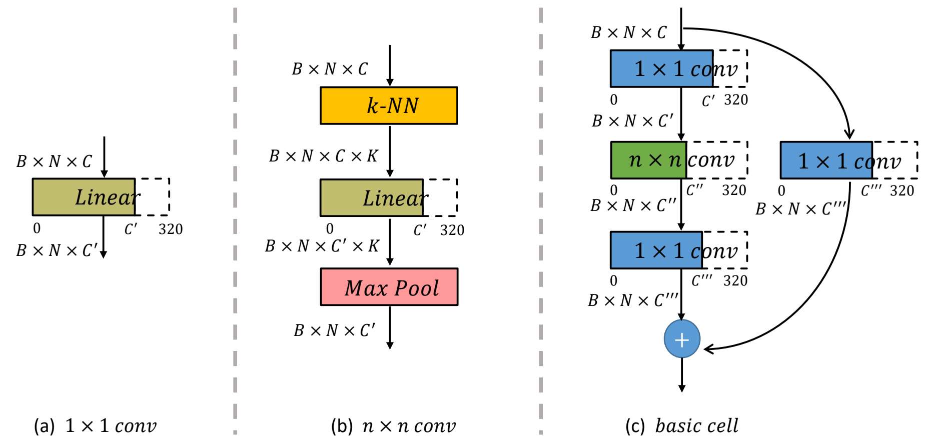

Fig. 2. Illustration of 1×1 convolution (a), $n \times n$ convolution (b) and basic cell (c). 'Linear' means a linear layer shared by all points. Note that the output feature channels of each 'Linear' layer are dynamic (The parts in solid lines mean they are chosen while those in dashed lines are not.), leading to various basic cells.

number of linear regions can be estimated by calculating the Hamming distance d_H of different inputs in a minibatch.

$$\texttt{naswot} : \mathcal{S}^{\text{naswot}} = \log \left| \begin{pmatrix} M - d_H(\boldsymbol{p}_1 - \boldsymbol{p}_1) & \dots & M - d_H(\boldsymbol{p}_1 - \boldsymbol{p}_N) \\ \vdots & \ddots & \vdots \\ M - d_H(\boldsymbol{p}_N - \boldsymbol{p}_1) & \dots & M - d_H(\boldsymbol{p}_N - \boldsymbol{p}_N) \end{pmatrix} \right| \tag{3}$$

Zen-score [13] calculates the change of deep layer feature maps $f(\boldsymbol{X}, \boldsymbol{\theta})$ in the network after adding disturbance to the input $\boldsymbol{X}$. This result is equal to the Gaussian complexity of the neural network, which considers not only the distribution of linear regions but also the Gaussian complexity of the linear classifier in each linear region.

$$\texttt{zen-score} : \mathcal{S}^{\text{zen-score}} = \log \left\| \frac{\partial f(\boldsymbol{X}, \boldsymbol{\theta})}{\partial \boldsymbol{X}} \right\| \tag{4}$$

Through the classification of existing training-free proxies, we can jointly use proxies with different properties for neural architecture search. We believe that NAS can achieve better performance through this ensemble learning-like way. Experiments have also proved this view.

3.3 Point Cloud Search Space

Similar to many classical neural architecture search algorithms [36,34,21,3], we design a cell-based search space, the macro architecture of the network is stacked

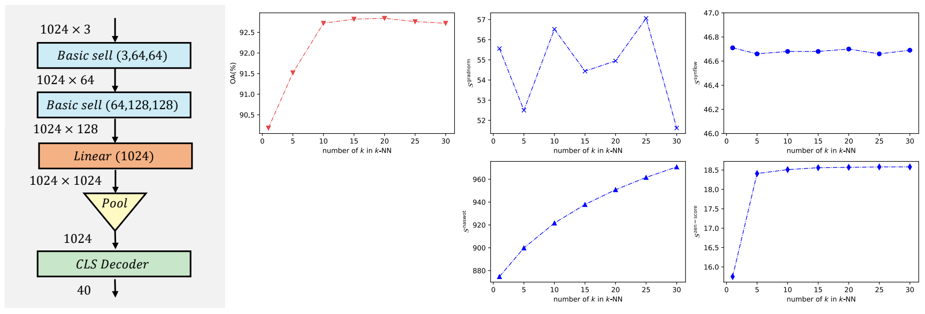

Fig. 3. Exploring the effects of hyperparameter k on network performance and training-free proxies. k stands for the number of neighbors obtaining in k-NN algorithm. The left side shows the network structure used in this experiment, and the right side shows the effects of k to overall accuracy and proxies. Except `naswot`, other proxies are not sensitive to k.

by several basic cells, only the inner architecture of each cell needs to be searched. In our paper, as shown in Figure 2, a **basic cell** is a bottleneck block[5] consisting of two 1×1 convolutions and one $n \times n$ convolution. However, 3D point cloud data is disordered, and it is difficult to use regular convolutions in 2D image processing to process it, so we first define the convolutions for 3D point cloud processing based on previous studies.

1×1 convolution comes from PointNet [18]. Considering the permutation invariance of 3D point cloud data, PointNet first proposes to use a linear layer with shared weights to process each point. This operation well maintains the permutation invariance but can not mine the relationship between points. We take this linear layer with shared weights as a basic component which corresponds to 1×1 convolution in 2D image processing.

$n \times n$ convolution comes from DGCNN [28]. During a $n \times n$ convolution, the k-NN algorithm is firstly used to obtain the features of k neighbors of each point in the point cloud based on the feature distance between points. Then a linear layer with shared weights is used to deal with these features (maintaining permutation invariance). Finally, a max pooling operation is used to aggregate features from neighbors. We take these operations as a basic component which corresponds to the convolution with a size greater than 1×1 in 2D image processing.

It should be noted that in 2D image processing, the receptive field of $n \times n$ convolution is determined by the size of n, while in 3D point cloud processing, the receptive field of $n \times n$ convolution is determined by the size of k in k-NN. In our experiments (Figure 3), we fix the network structure and the number of feature channels, and only change the value of k in the $n \times n$ convolution to observe the change of overall accuracy (training from scratch on ModelNet40 [29] dataset) and proxies, we can find that except `naswot`, other training-free proxies are not sensitive to the value of k. This may be because the parameter amount of $n \times n$ convolution in 3D point cloud processing is completely determined by its

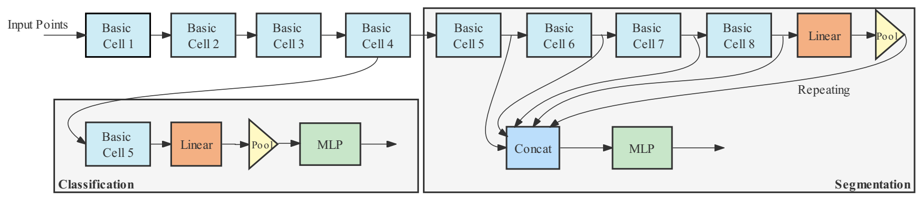

Fig. 4. Examples of the network structure for classification with 5 basic cells and the network structure for segmentation with 8 basic cells, respectively. Note that we only search structures of network encoder, and the decoder part is a multi-layer perceptron.

linear layer. This operation is similar to 1×1 convolution in the case of random initialization. Therefore, in this paper, we fix the k value as the empirical value of 20.

As shown in Figure 4, we stack multiple different basic cells to form the encoder of the network, the decoder is a multi-layer perceptron. In the search stage, The number of output feature channels of three convolutions in each cell varies in [32,64..., 320], which means that there are 1000 different cells in our search space.

We randomly sample 20 classification networks with 3 cells in this search space, calculate their values on different training-free proxies, and train them from scratch on ModelNet40 [29] dataset to obtain the overall accuracy. Then we report the correlation between the proxies and the accuracy. As shown in Figure 1, `gradnorm` does not perform well in this search space. Therefore, in the subsequent experiments, we will not consider `gradnorm`. The other three proxies can achieve high correlation coefficients.

3.4 Training-free NAS

We use the standard single-objective evolutionary algorithm in [13] as a baseline. In this paper, we call it SE-NAS (**S**ingle-objective **E**volutionary **N**eural **A**rchitecture **S**earch). It uses one single training-free proxy as objective to rank candidate structures and randomly mutate structures in population. In [13], they only use `zen-score` as objective. We also use `synflow` and `naswot` for comparison. After multiple iterations of SE-NAS, we select the three highest-scoring structures in the population and retrain these structures to report the optimal performance they can achieve.

To use multiple proxies jointly in an ensemble learning way, we design ME-NAS (Algorithm 1): a **m**ulti-objective **e**volutionary **n**eural **a**rchitecture **s**earch algorithm. Different from SE-NAS, since each structure corresponds to multiple proxies, it is difficult for us to compare two structures directly through the value of proxies. Therefore, we introduce the concept of Pareto dominance, that is, when the values of a structure $\mathcal{A}$ on all proxies are better than those of structure $\mathcal{B}$, then structure $\mathcal{A}$ dominates structure $\mathcal{B}$, or structure $\mathcal{A}$ is better

Algorithm 1 ME-NAS

```
Require: Search space $\mathcal{S}$, FLOPs constraint $B$, maximal depth $L$, total number of
    iterations $T$, population size $N$, number of new individuals per iteration $n$.
Ensure: Optimal structures.
 1: Randomly generate $N$ structures $\{F_0, \ldots, F_N\}$ and calculate their training-free
    proxies.
 2: Initialize population $\mathcal{P} = \{F_0, \ldots, F_N\}$.
 3: for $t = 1, 2, \cdots, T$ do
 4:     Devide $\mathcal{P}$ into $M$ non-dominated fronts $(\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_M)$ by fast non-dominated
        sort on training-free proxies and calculate crowding-distance in each non-
        dominated front.
 5:     Define new population $\hat{\mathcal{P}}$.
 6:     for $m = 0, 1, \cdots, M$ do
 7:         if $|\hat{\mathcal{P}}| + |\mathcal{R}_m| > N$ then
 8:             break.
 9:         end if
10:         Add all individuals in $\mathcal{R}_m$ to $\hat{\mathcal{P}}$.
11:     end for
12:     Fill $\hat{\mathcal{P}}$ to $|\hat{\mathcal{P}}| = N$ by adding individuals from $\mathcal{R}_m$ according to crowding-
        distance.
13:     $\mathcal{P} = \hat{\mathcal{P}}$.
14:     Repeat steps 4 once.
15:     Select $n$ individuals and MUTATE (Algorithm 2) them to generate $n$ new indi-
        viduals.
16:     Calculate training-free proxies of new individuals and add them to $\mathcal{P}$.
17: end for
18: Return all architectures in $\mathcal{R}_0$.
```

than structure $\mathcal{B}$. Through this method, we can divide all structures in the population into multiple non-dominated fronts, the structure of the upper front dominates the structure of the lower front, and there is no dominance relationship between the structures of the same front. When the population evolves, we select the structure to be mutated by its front. The higher the level, the more likely it is to mutate. Finally, ME-NAS returns multiple top-front structures (usually 2 to 4), which we individually retrain and report the optimal performance they can achieve.

4 Experiments

4.1 Implementation Details

Our experiments are mainly carried out on the 3D point cloud classification and part segmentation task. First, we compare SE-NAS and ME-NAS under different computation constraints and compare our results with existing methods on 3D point cloud classification. Then, we give the quantitative and qualitative results of 3D point cloud part segmentation.

Algorithm 2 MUTATE

```
Require: Structure $F_t$, Cell number $M$ of $F_t$, Search space $\mathcal{S}$.
Ensure: Randomly mutated architecture $\hat{F}_t$.
1: Uniformly select an integer $i$ from $[0, M]$.
2: if $i = M$ then
3:     Generate a new cell $\hat{b}$ from $\mathcal{S}$.
4:     Add $\hat{b}$ to the end of $F_t$.
5: else
6:     Uniformly alternate channel width of $i$-th cell in $F_t$ within some range or even
       delete this cell.
7: end if
8: Return the mutated architecture $\hat{F}_t$.
```

During neural architecture search, we set the maximum number of cells to 5 and 8 for classification and part segmentation, respectively. The number of input channels of each cell is determined by the previous cell, and the number of middle-layer channels and output channels are selected from $[32, 64, ..., 320]$. The population size of the evolutionary algorithm is 50, and the number of iterations is 30000. To show that our algorithm can be flexibly applied to various devices, we set several different computation constraints in the search process. For classification tasks, we set the computation constraint to 0.5G MACs (Tiny), 1G (Small), 2G (Middle), 3G (Large), 4G (Huge). For segmentation tasks, we set the computation constraint to 4G (Tiny), 8G (Small), 12G (Middle), 16G (Large), 20G (Huge).

Our classification experiments are carried out on the ModelNet40 [29] dataset. This dataset contains 12311 meshed CAD models from 40 categories, where 9843 models are used for training and 2468 models for testing. We randomly sample 1024 points from each object as network inputs and only use the $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ coordinates of the sampled points as input. The training strategy is similar to [35]. It should be noted that the randomness of 3D point cloud classification is relatively large, so in all classification experiments, we train the same structure three times with different seeds and report the average accuracy.

Table 2. SE-NAS and ME-NAS with different training-free proxies under different computational constraints.

Method	`synflow`	`naswot`	`zen-score`	Tiny	Small	Middle	Large	Huge
SE-NAS	✓			92.82	93.01	93.05	93.29	93.31
SE-NAS		✓		92.92	93.06	93.22	93.33	93.46
SE-NAS [13]			✓	92.85	92.99	93.12	93.24	93.37
ME-NAS	✓	✓		93.03	**93.20**	**93.40**	**93.53**	93.66
ME-NAS		✓	✓	93.00	93.07	93.27	93.38	93.49
ME-NAS	✓		✓	93.02	93.17	93.37	93.50	93.68
ME-NAS	✓	✓	✓	**93.08**	93.19	93.38	**93.53**	**93.71**

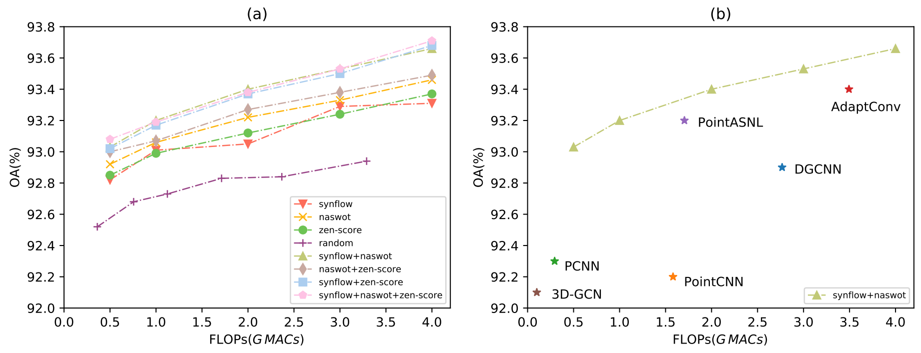

Fig. 5. Exploring the relationships between computational constraints during NAS (horizontal axis) and network performance (vertical axis) on the ModelNet40 dataset. (a) shows the comparisons of SE-NAS and ME-NAS with different training-free proxies. `random` means randomly selecting structures. (b) shows the comparisons of ME-NAS (synflow+naswot) and other studies.

Our part segmentation experiments are carried out on the ShapeNetPart [33] dataset. This dataset contains 16,881 shapes from 16 categories, with 14,006 for training and 2,874 for testing. Each point is annotated with one label from 50 parts, and each point cloud contains 2-6 parts. We randomly sample 2048 points from each shape for segmentation. The training strategy is similar to [35].

4.2 3D Point Cloud Classification

SE-NAS and ME-NAS under Different Computational Constraints We report the results of SE-NAS and ME-NAS under different computational constraints and different training-free proxies in Table 2 and Figure 5 (a), where `random` means randomly selecting structures within the computational constraints. As we can see, all training-free proxies can achieve better results than `random`, and jointly using multiple proxies that reflect different properties in an ensemble learning way can further improve performance.

Comparisons with Other Works
We compare the results[3] reported by ME-NAS (synflow+naswot) with other studies in 3D point cloud classification. In Figure 5 (b), we draw the FLOPs-OA diagram[4]. In Table 3, we provide more comparison results. Experiments show that with a large computational constraints, the performance of the proposed ME-NAS surpasses existing manual methods and one-shot NAS methods, with a small computational constraints, ME-NAS can also achieve relatively good performance.

[3] Note that we do not use the voting strategy during testing in all experiments.
[4] We just test a few networks' FLOPs, cause some papers are not source-opened.

Table 3. Classification results of ME-NAS (synflow+naswot) and other studies on the ModelNet40 dataset. ‘Design’ means how this work is designed, ‘MD’ means manual design. ‘OS’ means one-shot NAS methods, ‘ZS’ means zero-shot NAS methods.

Method	Design	Input	#points	OA(%)
PointNet [18]	MD	xyz	1k	89.2
PointNet++ [19]	MD	xyz, normal	5k	91.9
3D-GCN [15]	MD	xyz, normal	1k	92.1
PointCNN [11]	MD	xyz	1k	92.2
PCNN [2]	MD	xyz	1k	92.3
LC-NAS [9]	OS	xyz	1k	92.8
DGCNN [28]	MD	xyz	1k	92.9
KPConv [25]	MD	xyz	6.8k	92.9
PointASNL [32]	MD	xyz, normal	1k	93.2
SGAS [8]	OS	xyz	1k	93.2
AdaptConv [35]	MD	xyz	1k	93.4
PolyConv [14]	OS	xyz	1k	93.5
ME-NAS (synflow+naswot)-Tiny	ZS	xyz	1k	93.03
ME-NAS (synflow+naswot)-Small	ZS	xyz	1k	93.20
ME-NAS (synflow+naswot)-Middle	ZS	xyz	1k	93.40
ME-NAS (synflow+naswot)-Large	ZS	xyz	1k	93.53
ME-NAS (synflow+naswot)-Huge	ZS	xyz	1k	93.66

4.3 3D Point Cloud Part Segmentation

Comparisions with Other Works
3D point cloud part segmentation is a more challenging task, we compare the results reported by ME-NAS (synflow+naswot) with other studies on the ShapeNet-Part dataset in Table 4, the proposed ME-NAS (synflow+naswot) can also surpass existing methods.

Visualization
Finally, we report the visualization results of ME-NAS (synflow+naswot)-Tiny on the ShapeNetPart dataset. As shown in Figure 6, our model can achieve results similar to ground truth annotations on multiple categories.

5 Conclusions

In this paper, we propose a novel training-free neural architecture search algorithm for 3D point cloud processing. Our method can flexibly adjust the computation constraint of the searched structures to meet the needs of various devices. Experiments on 3D point cloud classification and part segmentation show that our method can achieve competitive results under different computation constraints, which will promote the development of downstream tasks. Futhermore, our method is highly scalable, future research can add new cells to the search space or come up with new training-free proxies for a better performance.

Table 4. Part segmentation results of ME-NAS (synflow+naswot) and other studies on the ShapeNetPart dataset evaluated as the mean class IoU (mcIoU) and mean instance IoU (mIoU).

Method	mcIoU	mIoU	air plane	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor bike	mug	pistol	rocket	skate board	table
Kd-Net [6]	77.4	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	87.4	86.7	78.1	51.8	69.9	80.3
PointNet [18]	80.4	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++ [19]	81.9	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
SO-Net [10]	81.0	84.9	82.8	77.8	88.0	77.3	90.6	73.5	90.7	83.9	82.8	94.8	69.1	94.2	80.9	53.1	72.9	83.0
DGCNN [28]	82.3	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
PCNN [2]	-	85.1	82.4	80.1	85.5	79.5	90.8	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
PointCNN [11]	-	86.1	84.1	86.4	86.0	80.8	90.6	79.7	92.3	88.4	85.3	96.1	77.2	95.3	84.2	64.2	80.0	83.0
PointASNL [32]	-	86.1	84.1	84.7	87.9	79.7	92.2	73.7	91.0	87.2	84.2	95.8	74.4	95.2	81.0	63.0	76.3	83.2
3D-GCN [15]	82.1	85.1	83.1	84.0	86.6	77.5	90.3	74.1	90.9	86.4	83.8	95.6	66.8	94.8	81.3	59.6	75.7	82.8
AdaptConv [35]	83.4	86.4	84.8	81.2	85.7	79.7	91.2	80.9	91.9	88.6	84.8	96.2	70.7	94.9	82.3	61.0	75.9	84.2
ME-NAS (synflow+naswot)-Tiny	83.1	85.9	84.1	84.4	88.0	79.2	91.1	74.7	91.5	88.2	85.7	96.0	69.8	94.0	82.0	59.5	76.2	83.3
ME-NAS (synflow+naswot)-Small	83.5	86.1	84.5	85.3	88.6	80.2	91.0	76.1	92.2	88.4	85.7	95.4	71.0	94.3	82.4	60.5	76.2	82.9
ME-NAS (synflow+naswot)-Middle	83.7	86.3	85.0	85.6	87.3	80.6	91.2	74.3	92.1	88.6	86.0	96.1	72.1	94.2	82.3	61.2	74.6	83.1
ME-NAS (synflow+naswot)-Large	83.9	86.5	85.1	86.0	88.3	80.4	91.2	77.6	91.3	88.6	85.4	96.1	71.6	94.6	82.2	62.1	75.4	83.6
ME-NAS (synflow+naswot)-Huge	84.1	86.6	85.3	86.2	88.6	80.5	91.5	77.8	91.7	88.7	86.0	96.0	71.8	94.4	82.3	62.6	75.4	83.3

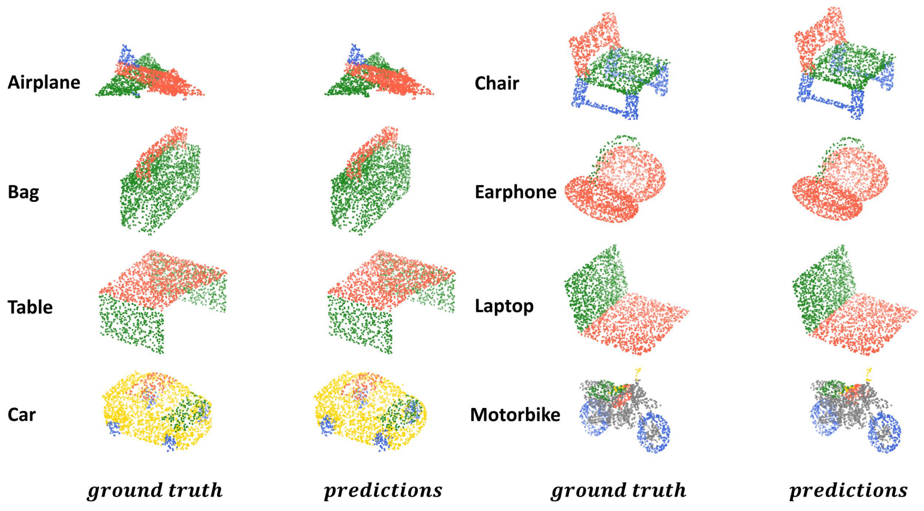

Fig. 6. Segmentation results on the ShapeNetPart dataset. We select 8 categories for visualization.

Acknowledgements The work is partially supported by the National Nature Science Foundation of China (No. 61976160, 61906137, 61976158, 62076184, 62076182) and Shanghai Science and Technology Plan Project (No.21DZ1204800) and Technology research plan project of Ministry of Public and Security (Grant No.2020JSYJD01).

References

1. Abdelfattah, M.S., Mehrotra, A., Dudziak, Ł., Lane, N.D.: Zero-cost proxies for lightweight nas. arXiv preprint arXiv:2101.08134 (2021)

2. Atzmon, M., Maron, H., Lipman, Y.: Point convolutional neural networks by extension operators. arXiv preprint arXiv:1803.10091 (2018)
3. Elsken, T., Metzen, J.H., Hutter, F.: Efficient multi-objective neural architecture search via lamarckian evolution. arXiv preprint arXiv:1804.09081 (2018)
4. Hanin, B., Rolnick, D.: Complexity of linear regions in deep networks. In: International Conference on Machine Learning. pp. 2596–2604. PMLR (2019)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
6. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 863–872 (2017)
7. Lee, N., Ajanthan, T., Torr, P.H.: Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:1810.02340 (2018)
8. Li, G., Qian, G., Delgadillo, I.C., Muller, M., Thabet, A., Ghanem, B.: Sgas: Sequential greedy architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1620–1630 (2020)
9. Li, G., Xu, M., Giancola, S., Thabet, A., Ghanem, B.: Lc-nas: Latency constrained neural architecture search for point cloud networks. arXiv preprint arXiv:2008.10309 (2020)
10. Li, J., Chen, B.M., Lee, G.H.: So-net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9397–9406 (2018)
11. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. vol. 31 (2018)
12. Liang, M., Yang, B., Wang, S., Urtasun, R.: Deep continuous fusion for multi-sensor 3d object detection. In: Proceedings of the European Conference on Computer Vision. pp. 641–656 (2018)
13. Lin, M., Wang, P., Sun, Z., Chen, H., Sun, X., Qian, Q., Li, H., Jin, R.: Zen-nas: A zero-shot nas for high-performance image recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 347–356 (2021)
14. Lin, X., Chen, K., Jia, K.: Object point cloud classification via poly-convolutional architecture search. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 807–815 (2021)
15. Lin, Z.H., Huang, S.Y., Wang, Y.C.F.: Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1800–1809 (2020)
16. Mellor, J., Turner, J., Storkey, A., Crowley, E.J.: Neural architecture search without training. In: International Conference on Machine Learning. pp. 7588–7598 (2021)
17. Nie, X., Liu, Y., Chen, S., Chang, J., Huo, C., Meng, G., Tian, Q., Hu, W., Pan, C.: Differentiable convolution search for point cloud processing. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
18. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)
19. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. vol. 30 (2017)

20. Ran, H., Zhuo, W., Liu, J., Lu, L.: Learning inner-group relations on point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15477–15487 (2021)
21. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Aging evolution for image classifier architecture search. In: AAAI Conference on Artificial Intelligence. vol. 3 (2019)
22. Rusu, R.B., Marton, Z.C., Blodow, N., Dolha, M., Beetz, M.: Towards 3d point cloud based object maps for household environments. Robotics and Autonomous Systems **56**(11), 927–941 (2008)
23. Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow. vol. 33, pp. 6377–6389 (2020)
24. Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S.: Searching efficient 3d architectures with sparse point-voxel convolution. In: European Conference on Computer Vision (2020)
25. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6411–6420 (2019)
26. Wang, C., Zhang, G., Grosse, R.: Picking winning tickets before training by preserving gradient flow. arXiv preprint arXiv:2002.07376 (2020)
27. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2589–2597 (2018)
28. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics **38**(5), 1–12 (2019)
29. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1912–1920 (2015)
30. Xiang, T., Zhang, C., Song, Y., Yu, J., Cai, W.: Walk in the cloud: Learning curves for point clouds shape analysis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 915–924 (2021)
31. Xiong, H., Huang, L., Yu, M., Liu, L., Zhu, F., Shao, L.: On the number of linear regions of convolutional neural networks. In: International Conference on Machine Learning. pp. 10514–10523. PMLR (2020)
32. Yan, X., Zheng, C., Li, Z., Wang, S., Cui, S.: Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5589–5598 (2020)
33. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics **35**(6), 1–12 (2016)
34. Zhong, Z., Yan, J., Wu, W., Shao, J., Liu, C.L.: Practical block-wise neural network architecture generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2423–2432 (2018)
35. Zhou, H., Feng, Y., Fang, M., Wei, M., Qin, J., Lu, T.: Adaptive graph convolution for point cloud analysis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4965–4974 (2021)
36. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8697–8710 (2018)