PIDS: Joint Point Interaction-Dimension Search for 3D Point Cloud

Tunhou Zhang[1], Mingyuan Ma[1], Feng Yan[2], Hai Li[1], Yiran Chen[1]
[1]ECE Department, Duke University, Durham, NC 27708
[2]Department of Computer Science, University of Houston, Houston, TX 77204
[1] {tunhou.zhang,mingyuan.ma,hai.li,yiran.chen}@duke.edu,
[2] fyan5@central.uh.edu

Abstract

The interaction and dimension of points are two important axes in designing point operators to serve hierarchical 3D models. Yet, these two axes are heterogeneous and challenging to fully explore. Existing works craft point operator under a single axis and reuse the crafted operator in all parts of 3D models. This overlooks the opportunity to better combine point interactions and dimensions by exploiting varying geometry/density of 3D point clouds. In this work, we establish PIDS, a novel paradigm to jointly explore point interactions and point dimensions to serve semantic segmentation on point cloud data. We establish a large search space to jointly consider versatile point interactions and point dimensions. This supports point operators with various geometry/density considerations. The enlarged search space with heterogeneous search components calls for a better ranking of candidate models. To achieve this, we improve the search space exploration by leveraging predictor-based Neural Architecture Search (NAS), and enhance the quality of prediction by assigning unique encoding to heterogeneous search components based on their priors. We thoroughly evaluate the networks crafted by PIDS on two semantic segmentation benchmarks, showing $\sim$ *1% mIOU improvement on SemanticKITTI and S3DIS over state-of-the-art 3D models.*

1. Introduction

The rise of 3D acquisition technologies and the increasing amount of geometric data collected from 3D sensors drive the booming of 3D point cloud applications, such as object recognition [37], shape segmentation [42], and indoor-scene segmentation [1]. Deep Neural Networks (DNNs) plays a critical role in processing 3D point cloud data in an end-to-end fashion [25]. The recent development of neural architecture engineering drives better performance [47, 46] and higher efficiency [15] for 3D point cloud models.

Table 1. Comparison of different methods in seeking optimal 3D point operators. Methods marked with $*$ leverage Neural Architecture Search with design automation.

Method	Interaction?	Dimension?	Reuse?	S3DIS mIOU (%)
KPConv [34]	✓		✓	70.6
PointTransformer [46]	✓		✓	73.5
PointSeaNet [28]*	✓		✓	71.9
PIDS (Ours)*	✓	✓		74.4

A hierarchical 3D model for point cloud is composed of several point operators. Researchers have identified two heterogeneous axes of a point operator: the interaction of points in a point cloud (point interaction), and the size dimension of the point operator (point dimension). Starting from PointNet [25, 26], point-based models employ various strategies to design a point interaction to extract features and make dense predictions for semantic segmentation on 3D point cloud. Later works propose Multi-layer Perceptron [25, 26], point convolutions [38, 2], graph neural networks [35], attention mechanism [46, 13, 44], kernel point convolutions [34], voxelization [7], and 2D projections [8, 12] to improve the performance-efficiency tradeoff on 3D point cloud, with manually tuned point dimensions in different hierarchies of a 3D model. More recent works leverage Neural Architecture Search (NAS) to explore a wider range of 3D model choices, such as point dimensions in voxelized point interactions [33] and the structural wiring of convolution-based point interactions [24].

Yet, existing approaches on crafting 3D operators have several limitations, due to the varying distribution of 3D point cloud. First, existing works only craft a single type of point interaction and reuse it within all point operators of a 3D model, see Table 1. The reuse of point interactions can limit the performance of the crafted 3D operators, due to the varying geometric/density distribution of points. Second, existing approaches optimize point interactions and point dimensions separately and only seek an optimal solution on a single axis. Such an approach misses the opportunity to discover a better combination of point interactions and dimensions, and thus limits the efficiency of the crafted

point operators. Overall, the crafted models usually have sub-optimal performance versus efficiency trade-off on processing 3D point clouds.

In this paper, we propose a joint interaction-dimension search to resolve the above limitations. We envision that existing NAS approaches may have several challenges in 3D point cloud. First, even with full design automation powered by NAS, it is challenging to jointly explore point interactions and point dimensions simultaneously and precisely on 3D point clouds with varying distribution. Thus, existing weight-sharing approaches such as differentiable NAS [20] and one-shot NAS [4, 41] may be infeasible, as it is difficult to learn shared weights over point operators, given the fast-changing 3D point cloud inputs. Second, heterogeneous search components add to the complexity of architecture-performance landscape, making it more difficult to accurately map architectures towards their ground-truth performance. The above challenges call for a more scalable and accurate NAS that innovates automatic model design on serving 3D applications.

To this end, we propose PIDS (Joint **P**oint **I**nteraction-**D**imension **S**earch for 3D Point Cloud), a novel paradigm that leverages NAS [49, 4, 36] to joint explore point interactions and point dimensions. In PIDS, we craft a massive search space containing point interactions with versatile geometry/density in 3D point cloud and positional size-based point dimensions. For point interactions, we backbone the search space on kernel point convolution [34], and introduce high-order point interactions to fit geometric dispositions and density distributions within varying 3D points. For point dimensions, we incorporate the design motif from Inverted Residual Bottleneck (IRB [30]) to enable more efficient size-based search components (i.e., width, depth, and expansion factor). We guide PIDS with predictor-based NAS [36]. We train a neural predictor to map architecture-performance pairs in PIDS search space. To improve the quality of search, we innovate the design of a **D**ense-**S**parse (DS) predictor to encode unique priors to point interactions and point dimensions. DS predictor uses sparse embedding features to encode categorical choice of point interactions, and uses dense features to represent continuous choice of point dimensions. The dense features and sparse features are then interacted to compute cross-term product, yielding a superior neural architecture representation that improves performance prediction.

The joint interaction-dimension search in PIDS enables to discover efficient 3D models, and the DS predictor of PIDS allows a better ranking of candidate models (i.e., up to 0.03 Kendall τ improvement and 2.6% higher mIOU on semantic segmentation over prior state-of-the-art predictors) to improve the quality of search. As a result, PIDS crafts high-performing and efficient 3D architectures on various semantic segmentation benchmarks, such as SemanticKITTI [3] and S3DIS [1]. On S3DIS, PIDS network outperforms the state-of-the-art hand-crafted architecture by $\sim 0.9\%$ higher mIOU and is $3.6\times$ parameter-efficient. On SemanticKITTI, PIDS network achieves 0.6% higher mIOU, saves $7.2\times$ parameters, and reduces $7.4\times$ MACs over state-of-the-art NAS-crafted architecture.

We highlight the contributions of this paper as follows:

- We propose a new paradigm, PIDS, to jointly explore point interactions and dimensions in a vast search space on 3D point cloud. PIDS optimizes both axes of the point operator and seeks 3D models to strike a balance in performance-efficiency trade-off.
- We leverage predictor-based NAS to accurately model candidate networks in the PIDS search space. We further enhance the quality of performance prediction by proposing Dense-Sparse predictor that encodes unique priors on heterogeneous search components and computes cross-term product to gather a better architecture representation.
- The best model discovered by PIDS achieves the state-of-the-art mIOU on SemanticKITTI and S3DIS over existing models, with higher efficiency.

2. Related Work

Deep Learning for 3D Point-Cloud. Point interactions are often achieved via point-based networks [38, 15, 17, 34] that carries feature aggregations towards irregular point set grids to benefit from learnable radial function and spherical harmonics in 3D point clouds. Recent point-based literature on point interactions improves point sampling efficiency [15, 17], local feature aggregations [46, 19], manual architecture fabrications [16, 18], improved training protocols [12], and/or better feature selection [39, 6] to seek higher model performance and efficiency. However, most of these works manually design a building block, while overlooking the varying geometry and density distributions (i.e., heterogeneity) of 3D point clouds. In addition, these manual efforts may not be able to explore a wider search space of the point interactions and miss the opportunity to find innovative architecture patterns. The above limitations in the size of search space (i.e., point dimensions) may negatively impact the quality and efficiency of the designed models.

Predictor-based NAS. Predictor-based NAS [36, 10] is a popular method that trains a performance predictor on limited samples. The trained predictor serves as a surrogate model of the ground-truth performance and is utilized to guide architecture search on the full design space. Existing achievements on predictor-based NAS lies in improving sample efficiency [10] and augmenting architecture samples [21] of the performance predictors, yet does not emphasize on improving the neural architecture representations to reduce prediction error. In the design space for 3D models, existing approaches may be subject to weak prediction per-

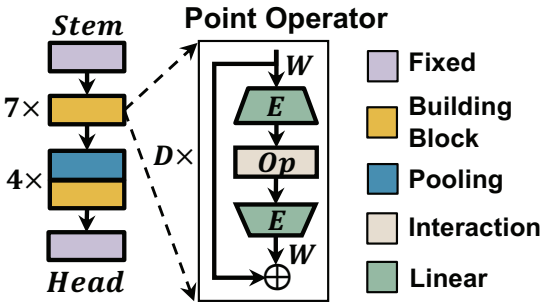

Figure 1. Overview of PIDS search space. Within each point operator, Op denotes the search component for point interactions. D: Depth, W: Width and E: Expansion Factor denote the search component for point dimensions.

formance without the correct neural architecture representations given versatile heterogeneous search components, requiring a deep study on the encoding of priors on different heterogeneous search components.

3. PIDS Search Space

In this section, we formally present PIDS, a new paradigm that jointly searches point interactions and point dimension. Figure 1 demonstrates the joint interaction-dimension search space in PIDS. A 3D model for semantic segmentation uses an encoder-decoder structure with 11 searchable stages, with 7 stages in backbone encoder model and 4 stages in segmentation decoder model. The searchable stages are sandwiched by fixed stem/head layer to correctly handle inputs/outputs. Each stage is composed of several point operators as basic building blocks. Each point operator has configurable point interactions (e.g., orders of interaction) and point dimensions (e.g., width) in the search space. Intuited by the design motifs from KPConv [34], PIDS proposes high-order point interactions and seeks the optimal adaptations towards varying geometry and density in different part of the point cloud. Intuited by Inverted Residual Bottleneck (IRB) [30], PIDS expands the exploration scope for point dimensions (i.e., depth, width, and expansion factor within a point operator) to discover more flexible model choices.

3.1. High-order Point Interactions

In a 3D point cloud, a point interaction is defined as a symmetric function over a center point X and its N neighboring points X_N. Given a center point $X \in \mathcal{R}^3$ and its corresponding features with D dimensions: $F \in \mathcal{R}^D$, point interaction takes all N neighbor points $X_n \in R^{N\times 3}$ with their corresponding features $F \in \mathbb{R}^{N\times D}$ to compute the output features $F' \in \mathbf{R}^{N\times D}$ via a learnable transformation f parameterized by θ, illustrated as follows:

$$F' = f(F, X, X_n; \theta). \tag{1}$$

We use the intuitions from kernel point convolution [34] to build point interactions. This is because in a kernel point convolution, the neighbor points X_n is efficiently achieved by performing radius sampling, taking up only ∼30% of the total computation cost. Thus, optimizing the point interactions and point dimensions shows promising improvement on the overall 3D model. Based on kernel point convolution, we introduce our first-order and second-order point interactions. A stacking of the first/second point interactions lead to high order point interactions in the 3D model.

First-order Point Interaction. A first-order point interaction assigns a unique kernel $\mathcal{K}$ with K kernel points that carry weights $W_k \in \mathbb{R}^{K\times D}$ of all feature dimensions. A first-order point interaction adopts a linear correlation function $h_l : \mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{K\times D}$ that maps the unordered neighbor point features F to a set of features $\bar{\mathbf{F}} = [\bar{F}_1, \bar{F}_2, ..., \bar{F}_K]$ on K kernel points. Specifically, a linear correlation function $h^{(l)}$ measures the contribution of a neighbor point $x_i \in X_n$ towards a kernel point $\hat{x}_k$ and outputs a linear correlation $\mathcal{H}^{(l)} \in \mathbb{R}^{N\times K}$ as follows:

$$\mathcal{H}^{(l)}_{i,k} = h^{(l)}(x_i, \hat{x}_k) = \max(0, 1 - \frac{||x_i - \hat{x}_k||_2}{\delta}), \tag{2}$$

Where δ is a hyperparameter that indicates the influence of a kernel over neighboring points. The first-order interaction aggregates the kernel features $\bar{\mathbf{F}}$ via:

$$f_{1st}(F_i) = \sum_k \mathcal{H}^{(l)} \hat{F}_k W_k \tag{3}$$

$$= \sum_{x_i \in X_n} h^{(l)}(x_i, \hat{x}_k) F_i W_k, \forall x_i \in X_n, \tag{4}$$

where $\hat{F}_k \in \mathbb{R}^{\mathbb{D}}$ indicates a D-dimensional feature mapped to the kernel points, F_i denotes an original D-dimensional feature maintained in local neighborhood points. Eq. 3,4 establish a direct interaction between a center point and its surrounding neighbor points by an explicit form of linear combination, carried by a specific kernel point W_k that simultaneously operate on all D features. Different from the original kernel point convolution, first-order point interaction carries element-wise multiplication between weighted

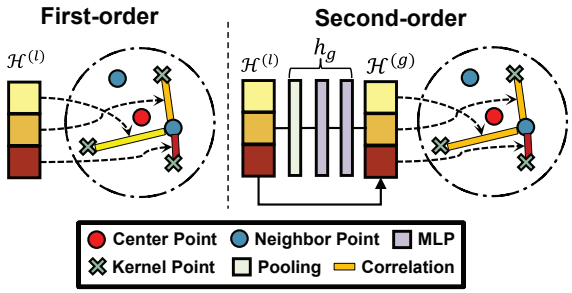

Figure 2. First-order and second-order point interaction. The first order point interaction locally models a center point towards a geometric kernel, and the second-order point interaction additionally utilizes global neighbors to capture points with varying density.

features (i.e., $H^{(l)}\hat{F}_k$) and kernel weights (i.e., W_k), without considering the channel-wise features. This saves $D\times$ parameters and Multiply-Accumulates compared to the original kernel point convolution, yielding higher efficiency in 3D models.

In KPConv, kernel point convolution uses a fixed kernel size (i.e., 15) within all parts of the model, indicating a fixed geometric kernel disposition $\hat{x}_k$ for points in different hierarchical levels of the point cloud. Our first-order interaction extends the scope of kernel dispositions to allow versatile choices in kernel dispositions. Given different kernel dispositions , the first-order point interaction has the ability to capture different geometric properties within the mast 3D point cloud, leading to higher quality of learned representations with fast-changing point cloud geometries.

Second-order Point Interaction. Note that in first-order point interaction, the linear correlation $\mathcal{H}^{(l)}$ that measures the importance of a kernel point towards a neighbor point purely uses the Euclidean distance between them. Such importance measurement overlooks the contribution of other neighbors, thus may not be accurate under point clouds with varying densities. For example, farther kernel points might be more important to neighbor points in a sparse point cloud, where each center point has only a few neighbor points within a sparse distribution of point clouds.

To cover the varying geometric and dense disposition of kernels, we introduce a density-aware correlation $H^{(g)}$ as an alternative to the linear correlation H_l. The density-aware correlation leverage the global neighborhood information of center points and make adaptations to the varying density of these points. Specifically, the density-aware linear correlation $H^{(g)}$ takes linear correlation $H^{(l)}$ as input, and uses gating function h_g to re-calibrate the kernel-neighbor importance as follows:

$$H^{(g)} = h_g(\mathcal{H}^{(l)}) = \sigma(\text{MLP}(\text{Pool}(\mathcal{H}^{(l)}))) \odot \mathcal{H}^{(l)}, \quad (5)$$

Here, σ is the sigmoid function, MLP denotes a 2-layer MLP layer with learnable weights, $Pool$ denotes the global average pooling operator that averages the information in the neighbor dimension of $\mathcal{H}^{(l)}$. Based on the density-aware linear correlation $H^{(g)}$, we establish the second-order interaction as follows:

$$f_{2nd}(F_i) = \sum_k \frac{1}{2}(\mathcal{H}^{(l)} + \mathcal{H}^{(g)})\hat{F}_k W_k, \forall x_i \in X_n. \quad (6)$$

As $H^{(g)}$ is a function of $H^{(l)}$, we employ a summation of $H^{(g)}$ and $H^{(l)}$ for in second-order interaction for easier optimization, following the spirit of residual learning [14]. The second-order interaction considers the relative position of neighbor points given a fixed kernel disposition, thus can quickly adapt to 3D point clouds with varying density. However, second-order point interaction uses slightly more resources due to use of a gating function h_g, thus should be used sparingly in the design of 3D models for efficiency.

Table 2. Evaluation of IRB Design on SemanticKITTI.

Architecture	Params (M)	MACs (G)	SemanticKITTI mIOU (%)
KPConv (Our impl.)	14.8	60.9	59.2
PIDS (first-order)	0.97	4.6	59.6
PIDS (second-order)	0.98	4.7	60.1

3.2. Efficient Point Dimensions

As point interactions only performs point-wise feature extraction, we sandwich high-order point interactions with 2 Fully-connected layer to form a PIDS point operator, see Figure 1. This enable channel-wise feature interactions in 3D point clouds. Intuited by the design of Inverted Residual Bottleneck (IRB) [30], we apply an expansion factor on the intermediate point interactions to enrich the representation in low-cost point interaction operations. This IRB design of point operator also provides a wider range of search components for point dimensions.

Next, we validate the effectiveness of the IRB design that combines point interactions and point dimensions in the PIDS point operator. We follow the layer organization of MobileNet-V2 [30] to construct a hand-crafted 3D model. We evaluate the crafted 3D model with first-order/second-order point interaction, and compare its performance with KPConv in Table 2. Even without search, the hand-crafted 3D model with the first-order interaction achieves achieves 0.4% higher mIOU with $12\times$ smaller size on SemanticKITTI compared to KPConv. The hand-crafted 3D model with second-order interaction pushes the mIOU improvement $\sim$ 0.5% higher, with only marginal increase in parameter count. The above empirical evaluations suggest two key insights in designing 3D models: 1) relaxing the choice of kernel dispositions in first-order point interaction provides more opportunities to adapt to geometric distribution of 3D points, and 2) carrying second-order point interaction that leverages global neighborhood information to measure kernel importance makes better adaptations to the varying density of points.

3.3. Search Components

In PIDS search space, we jointly search for the best combination of point interactions and point dimensions. We illustrate the two search components as follows:

- **Point Interactions.** We search for the type of interactions (i.e., first-order/second-order) and the choice of geometric kernel dispositions (i.e., 5-point Tetrahedron, 7-point Octahedron, and 13-point Icosahedron dispositions) contained within the point interactions. The choice within point interactions are categorical discrete options.
- **Point Dimensions.** We search for the width (i.e., number of features in each point operator), depth (i.e., the number of stacked of point operators) and expansion factor (i.e., ratio of point interactions to FC operators) of a point oper-

ator. The choice within point dimensions are continuous floating-point numbers.

Unlike NAS for image-based models that emphasize the search of size-based homogeneous components, the choice of point interactions and the choice of point dimensions are heterogeneous. In addition, with a total of 11 stages, the overall search space size contains up to 1.8×10^{19} possible architectures, making the architecture exploration under moderate search cost more challenging. This calls for a precise NAS method to precisely explore the joint point interaction-dimension search space.

4. Joint Point Interaction-Dimension Search

We employ predictor-based NAS to jointly explore the best combination of point interactions and point dimensions in 3D models for point cloud. Specifically, we sample a number of architectures from the joint search space, and use their architecture-performance pairs to train a performance predictor. Unfortunately, the large cardinality of the joint interaction-dimension search space raise challenges in accurately modeling the search space. In this section, we analyze the priors on search components and propose Dense-Sparse (DS) predictor, a novel surrogate model that encode unique priors to different heterogeneous search features to improve the quality of performance prediction. As a result, DS predictor improves ranking of 3D models within the joint interaction-dimension search space.

4.1. Dense-Sparse Predictor

We start with analyzing the priors on different search components in PIDS interaction-dimension search space. We note that point interactions settings are sparsely distributed within the search space. For example, the choice of kernel dispositions in first/second point interaction follows discrete geometry shapes with no explicit correlation that can be achieved via linear interpolation. Thus, the choice of point interactions possesses categorical priors which can be implicitly learned by embeddings. Alternatively, the choice of point dimensions encodes a linear/quasi-linear relationship towards the model performance, as is reflected in the compound scaling scheme in EfficientNet [32]. Thus, the choice of point interactions contains continuous priors that can be modelled in deep representations. As a result, using a vanilla predictor to model these multi-modal features with different priors may lead to sub-optimal results.

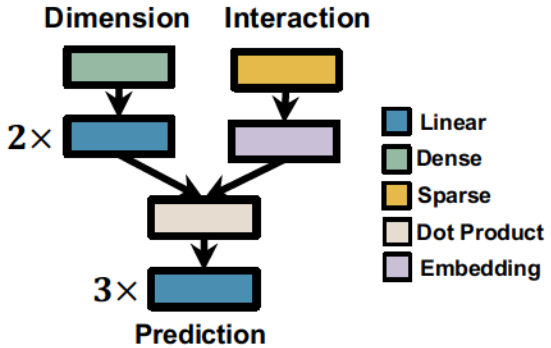

Figure 3. Dense-Sparse predictor computes a cross-term product on heterogeneous search components to improve the prediction quality of joint interaction-dimension search.

Motivated by Wide & Deep learning [5] that combines memorization and generalization for multi-modality inputs, we propose a novel Dense-Sparse predictor to encode unique priors on point interactions/dimensions. This improves the neural architecture representation of candidates in PIDS search space and thus, improves the quality of prediction and ranking of candidate models. We craft dense/sparse neural architecture representations for point interactions/dimensions, and leverage a dot product to compute the cross-term relationship between point interactions and point dimensions. Specifically, given a dense representation $X_d \in R^{B\times dim}$ and a sparse representation $X_s \in R^{B\times N\times dim}$, a dot product carries cross-term feature communication by carrying the following transformation:

$$Z = [vec^{-1}_{B,1,dim}(X_d); X_s], \tag{7}$$

$$DP(Z) = Triu(ZZ^T), \tag{8}$$

Where DP denotes the dot product operation, and $Triu$ denotes the upper triangular matrix. Figure 3 demonstrates the architecture DS predictor. DS predictor adopts a separate MLP to learn deep representations on the continuous dense representations (i.e., point dimensions), and adopts an embedding table to learn the categorical choice of sparse representations (i.e., point interactions). Thanks to the enhanced encoded priors and their cross-term relationship, DS predictor yields a more accurate modeling of the PIDS search space. Next, we elaborate the training and evaluation of our DS Predictor.

Architecture Sampling. We randomly sample ∼1K architectures on SemanticKITTI. Following common settings in NAS, we split the training dataset of SemanticKITTI into mini-train and mini-val. We train the sampled architectures on mini-train until convergence, and evaluate these architectures on mini-val to obtain architecture performance.

Predictor Training. Mean-Squared Error (MSE) lossis the training objective of the DS predictor on 1K sampled architectures. To ensure a consistent comparison of predictor performance over multiple benchmarks, we carry optimization on predictors with normalized performance. Since Dense/Sparse/Dense-Sparse predictor can perform gradient-based optimization, we incorporate predictor pre-training on Multiply-Accumulates (MACs) prediction [9], and use pair-wise margin rank loss [10] to improve ranking.

We compare DS predictor with top-performing predictor designs collected from NASBench-301 [31], see Ta-

Table 3. Wide & Deep Predictor achieves best prediction quality over SemanticKITTI.

Predictor	Rank Loss?	Priors	Pre-training?	MSE (10^{-2})	Kendall τ
Random Forest		Continuous		3.75±0.30	0.240±0.044
GPR		Continuous		4.29±0.41	0.144±0.042
XGBoost		Continuous		4.24±0.40	0.210±0.037
NGBoost		Continuous		3.90±0.33	0.255±0.047
LGBoost		Continuous		4.03±0.36	0.236±0.033
Dense	✓	Continuous	✓	3.07±0.33	0.379±0.050
Sparse	✓	Discrete	✓	2.87±0.21	0.400±0.047
Dense-Sparse	✓	Continuous + Discrete	✓	2.80±0.23	0.408±0.044

ble 3. Since sparse embedding is not feasible in Bayesian models (e.g., GPR) and tree-based methods (e.g., Random Forest), all categorical features are treated as continuous features during the training process. By enhancing search priors and computing cross-term feature relationship via a dot-product, DS predictor achieves better prediction quality and ranking on PIDS search space: DS predictor achieves up to 0.172 higher Kendall τ compared to prior arts, and 0.008 higher Kendall τ over Dense predictor that converts all search components to continuous features.

4.2. Evolutionary Search on DS Predictor

With a trained DS predictor, we follow regularized evolution [29] to effectively probe the PIDS search space to carry a joint search on point interactions and point dimensions. Given a candidate architecture A with performance prediction $\hat{P}$ and MACs M, we use $S(A) = \hat{P} - \beta \times \log M$ as our search objective, and empirically set β to 0.5 to trade-off performance and resource. We introduce a single mutation of an architecture genotype with the following sequence of actions in one of the stages:

- Change the kernel disposition in point operators.
- Change the order of interaction in point operators.
- Change the width/expansion factor in point operators.
- Change the depth of stacked point operators.

We use a population of 200 and a sample size of 150 to carry regularized evolution with 360 rounds over PIDS search space to craft NAS models.

5. Experiments

In this section, we evaluate PIDS over two semantic segmentation benchmarks, SemanticKITTI [3] and S3DIS [1]. Specifically, we carry an end-to-end search on SemanticKITTI to identify the best 3D architecture in PIDS design space, and transfer it to S3DIS.

5.1. Hyperparameter Settings

We describe the detailed hyperparameter settings on SemanticKITTI and S3DIS as follows.

SemanticKITTI. A single training batch contains 10 sub-sampled point clouds. On SemanticKITTI, we measure Multiply-Accumulates (MACs) over a scene with an average of ∼12.3K points, which gives a similar MAC count for KPConv architecture as is shown in [33]. We use a downsample rate of $0.06m$, and train our best model for 250 epochs with an initial learning rate of 0.04 and cosine learning rate schedule [22]. We adopt an L2 weight decay of 3e-4 and default data augmentation [34].

S3DIS. A single training batch contains 8 sub-sampled point clouds. We use a downsample rate of $0.04m$ following the original KPConv paper [34]. Specifically, we train our best model for 250 epochs with an initial learning rate of 0.04 and cosine learning rate schedule [22]. We adopt a L2 weight decay of 3e-4 and default data augmentation [34].

5.2. Evaluation on SemanticKITTI

We evaluate the top-performing models discovered by PIDS. We apply width scaling [30] on the NAS-crafted models and attach a $m\times$ suffix to denote the application of a $m\times$ width multiplier. This ensures a fair comparison with existing state-of-the-art models of similar sizes. We compare the performance and efficiency metrics such as parameter count and MACs.

Performance Evaluation. Table 4 shows the mIOU comparison on sequence 08 of the SemanticKITTI dataset. Our hand-crafted PIDS model outperforms point-based, projection-based, and voxel-based methods by at least 0.9%, 1.1%, and 1.2% mIOU respectively, with significant parameter and MAC reduction. The NAS-crafted PIDS model achieves $\sim$ 1% higher mIOU than the state-of-the-art SPVNAS while saving 5.8$\times$ parameters and 4.5$\times$ MACs. Applying a 2$\times$ width multiplier on NAS-crafted model further boosts the performance, demonstrating a good scalability of discovered 3D models.

Latency Analysis of Best PIDS Model. In Table 4, we notice that our NAS-crafted PIDS model is 4.2 $\times$ faster than original KPConv model despite of its higher mIOU. However, we also observe that the latency reduction is less significant compared to MAC reduction of PIDS models, especially compared to voxel-based methods. This is because voxel-based methods (e.g., SPVNAS [33]) benefit from 1) reduced overhead in point-based neighboring mechanisms 2) mature software-hardware co-design that enables inference with high throughput.

Table 4. mIOU on sequence 08 **(validation split)** of SemanticKITTI. Latency is measured with NVIDIA TITAN X Pascal on a single scene containing ∼60K points. Here, red/blue numbers denote computation/processing time. $^{+}$: Results from [48]. *: Results from [33].

Architecture	Method	Params (M)	MACs (G)	Latency (ms)	mIOU (%)
RandLANet [15]	Point-based	1.24	-	103	57.1
KPConv-rigid (our impl.)	Point-based	14.8	60.9	221 (164 + 57)	59.2
PolarNet [45]	Projection-based	13.6	135.0*	**62***	58.2^{+}
SalsaNext [8]	Projection-based	6.7	62.8*	71*	59.0^{+}
MinkowskiNet [7]	Voxel-based	5.5	28.5*	294*	58.9
PIDS (second-order)	Point-based	**0.97**	**4.7**	160 (103 + 57)	**60.1**
SPVNAS [33]	Voxel-based	3.3	20.0	**158**	61.5
		7.0	34.7	**175**	63.5
PIDS (NAS)	Point-based	**0.57**	**4.4**	169 (112 + 57)	62.4
PIDS (NAS, 2×)		1.36	11.0	206 (149 + 57)	**64.1**

Table 5. Operation-level latency breakdown of our searched PIDS model. We list the critical operations here and categorize less important operation as 'Other'. In column **Type**, **C/M** denotes computation-bounded/memory-bounded operation.

Operation	Type	Latency (ms)	Latency (%)
Preprocessing (CPU)	-	57.015	33.73
`aten :: sub`	C	23.396	13.84
`aten :: bmm`	C	20.236	11.97
`aten :: gather`	M	18.115	10.72
`aten :: mul`	C	12.487	7.39
`aten :: sum`	C	9.887	5.85
`aten :: addmm`	C	7.292	4.31
`aten :: threshold_`	C	4.444	2.63
`aten :: copy_`	M	2.912	1.72
`aten :: sqrt`	C	2.256	1.33
Other	-	11.00	6.51

To dive deep into this issue, we profile the latency breakdown of the NAS-crafted PIDS and demonstrate the profiling result in Table 5. Here, the preprocessing operations (including radius sampling, grid sub-sampling etc.) consumes 33.7% of the inference cost, neighbor gathering contributes to the memory-bounded operation (i.e., `aten :: gather`), which takes 10% of the total inference latency. Computation-bounded parts in KPConv such as depthwise convolution (i.e., `aten :: mul` and `aten :: sum`), MLPs (i.e., `aten :: bmm`) and point-wise local neighborhood movement (i.e., `aten :: sub`) contribute to ∼50% of the total inference cost.

Besides the optimization of architecture, we also envision some potential improvements that harness the hardware-software co-design to improve the efficiency of PIDS blocks, such as yet not limited to:

- Enable efficient radius sampling implementation to allow parallelization between different CPU cores.
- Utilize a fused version of convolution (*i.e., `aten :: bmm` and `aten :: sum`) that aggregates the feature outputs on each kernel point efficiently.
- Incorporate the choice of radius for each point operator into the design space and harness such options to discover more efficient point-based 3D models.

Table 6. 6-fold cross-validation results on S3DIS.

Architecture	mIOU (%)	mAcc (%)	OA (%)	Params (M)
InterpCNN [23]	66.7	-	88.7	-
RandLANet [15]	70.0	82.0	88.0	1.24
KPConv [34]	70.6	79.1	-	14.8
RPNet [28]	70.8	-	-	-
SCF-Net [11]	71.6	82.7	88.4	-
BAAF-Net [27]	72.2	**83.1**	88.9	**1.23**
PointTransformer [46]	73.5	81.9	90.2	4.9
PointSeaNet [28]	71.9	-	90.3	6.75
PIDS (NAS, 2×)	**74.4**	82.1	**90.3**	1.35

5.3. Architecture Transferability on S3DIS

We further verify the transferability of NAS-crafted PIDS models on S3DIS to evaluate its performance on indoor scene segmentation. Following the established protocols in 3D point cloud segmentation, we report both 6-fold cross-validation mIOU across all Area 1∼Area 6 splits, see Table 6. By convention, we also report the validation mIOU and mIOU per class on Area 5 to compare with existing methods on 3D point cloud, see Table 7.

Though our NAS-crafted PIDS model is not optimized on S3DIS, it shows a significant margin on the S3DIS 6-fold cross-validation benchmark compared with existing approaches such as [27, 46] and achieves 74.5% mIOU, a new state-of-the-art result using only 1.35M parameters (i.e., 3.6× fewer than the runner-up architecture, Point-Transformer [46]). On Area 5 of the S3DIS dataset, we observe that: (1) NAS-crafted PIDS achieves 1.8% higher mIOU than original KPConv with 11× parameter efficiency. (2) NAS-crafted PIDS achieves competitive performance over different classes on S3DIS indoor semantic segmentation and demonstrates at least ∼1% higher mIOU over 9 out of 13 classes compared to existing approaches.

6. Ablation Studies

In this section, we first visualize how second-order point interaction re-calibrates the contribution of each neighbor point to suit heterogeneous point clouds with varying den-

Table 7. mIoU per class on S3DIS Area-5.

Methods	mIoU	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book.	sofa	board	clut.
Pointnet [25]	41.1	88.8	97.3	69.8	0.1	3.9	46.3	10.8	52.6	58.9	40.3	5.9	26.4	33.2
Eff 3D Conv [43]	51.8	79.8	93.9	69.0	**0.2**	28.3	38.5	48.3	71.1	73.6	48.7	59.2	29.3	33.1
RNN Fusion [40]	57.3	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
KPConv [34]	65.4	92.6	97.3	81.4	0.0	16.5	**54.5**	69.5	90.1	80.2	**74.6**	**66.4**	63.7	58.1
PIDS(NAS, 2×)	**67.2**	**93.6**	**98.3**	**81.6**	0.0	**32.2**	51.5	**73.2**	**90.7**	**82.5**	73.3	64.7	**71.6**	**60.0**

sity, reflected by the learned coefficients over the geometry disposition of kernels. Then, we discuss the relationship between MSE reduction of neural predictors, and the quality improvement of discovered NAS models.

6.1. Second-order Point Interactions: Visualization

To verify the source of improvement from point interactions, we plot the dynamics of learned point interactions on a 7-point Octahedron kernel, see Figure 4. Here, we use the slope to denote the strength of the second-order interaction $\mathcal{H}^{(g)}$ over first-order interaction $H^{(l)}$, indicating the impact of neighbor points towards linear correlation captured by the gating function $h^{(g)}$. The learned second-order point interaction demonstrates a strong geometric harmony with Octahedron kernel dispositions (i.e., 1-4-2 groups among symmetrical axis): the two vertices that are the most distant from the center point show the most significant interaction (i.e., $\mathcal{H}^{(g)}$) towards global neighborhood information $\mathcal{H}^{(l)}$. This supports more robust recognition results, as these kernel points are involved in addressing the boundary effects of sparse and fast-changing point clouds. The 4 vertices connecting the square plane have the least significant interaction, suggesting a smoother learning on a denser point distributions to capture the features of point majorities. The center kernel point admits most information from centralized points, and thus co-adapts to global neighbor information and yields a moderate interaction strength.

6.2. Effectiveness of DS Predictor

We show that even marginal Kendall τ improvement (i.e., 0.008) in DS predictor leads to significant quality improvement of NAS-crafted models. We establish a random search baseline by selecting the top-5 models from the outcome of 1K random samples, and establish predictor-based NAS baseline by selecting top-5 models using the aforementioned runner-up predictor: Sparse predictor in Table 8. All models are evaluated on classification and segmentation benchmarks using the same 100-epoch training pipeline. We report the mean and standard deviation of performance on SemanticKITTI in Table 8.

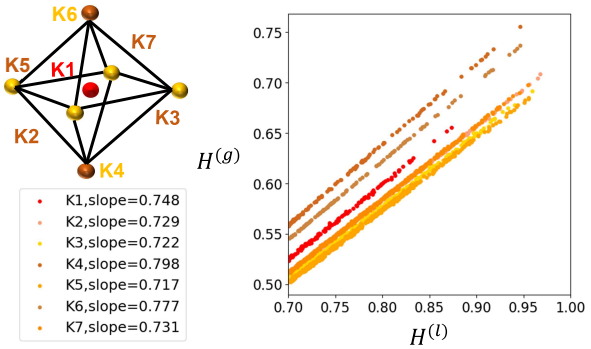

Figure 4. Visualization of the second-order point interaction on 7-point Octahedron kernel dispositions.

Table 8. Comparison of various predictor-based NAS methods with Random Search baseline. Results are derived from the top-5 NAS models discovered by each method.

Search Method	SemanticKITTI mIOU (%)			
	mean	std	max	min
Random	53.54	1.19	54.68	51.96
Sparse predictor	54.90	0.49	55.28	54.2
Dense-Sparse predictor	**56.10**	0.85	56.83	54.87

Results demonstrate that: (1) without introducing extra architecture samples, predictor-based NAS can outperform random search by up to 1.4% mIOU on SemanticKITTI. (2) Even with 0.07×10^{-2} lower MSE, the NAS-crafted models via DS predictor can outperform NAS-crafted models via Sparse predictor by a large margin (i.e., up to 1.2% mIOU on SemanticKITTI). This shows the significance of improving the design of predictor and enabling superior neural architecture representations for predictor-based NAS.

7. Conclusion

In this work, we present a new paradigm, Joint **P**oint **I**nteraction-**D**imension **S**earch for 3D Point Cloud (PIDS), to jointly search point interactions and point dimensions on two axes of 3D point operators. PIDS innovates high-order point interactions and efficient point dimensions to suit geometry and density heterogeneity of 3D point cloud data, and establishes a search space to carry joint exploration. PIDS utilizes predictor-based NAS and proposes a novel Dense-Sparse predictor to enhance the quality of predictions and the ranking of candidate networks. Dense-Sparse predictor utilizes enhanced priors to encode heterogeneous search components, and interact discrete/continuous architecture representations through a cross-term dot product. Results on 2 semantic segmentation benchmarks justify the state-of-the-art performance of NAS-crafted PIDS models, and its good transferability of searched models.

Acknowledgement. This project is in part supported by the following grants: NSF-2112562, NSF-1937435, and ARO W911NF-19-2-0107, and CAREER-2048044.

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[2] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018.

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019.

[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[6] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. 2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12547–12556, 2021.

[7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[8] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving. *arXiv preprint arXiv:2003.03653*, 2020.

[9] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using predictor pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16276–16285, 2021.

[10] Łukasz Dudziak, Thomas Chau, Mohamed S Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas D Lane. Brp-nas: Prediction-based nas using gcns. *arXiv preprint arXiv:2007.08668*, 2020.

[11] Siqi Fan, Qiulei Dong, Fenghua Zhu, Yisheng Lv, Peijun Ye, and Fei-Yue Wang. Scf-net: Learning spatial contextual features for large-scale point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14504–14513, June 2021.

[12] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. *arXiv preprint arXiv:2106.05304*, 2021.

[13] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[16] Zeyu Hu, Mingmin Zhen, Xuyang Bai, Hongbo Fu, and Chiew-lan Tai. Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 222–239. Springer, 2020.

[17] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: Differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7578–7588, 2020.

[18] Huan Lei, Naveed Akhtar, and Ajmal Mian. Seggcn: Efficient 3d point cloud segmentation with fuzzy spherical kernel. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11611–11620, 2020.

[19] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. Fpconv: Learning local flattening for point convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2020.

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[21] Yuqiao Liu, Yehui Tang, and Yanan Sun. Homogeneous architecture augmentation for neural predictor. *arXiv e-prints*, pages arXiv–2107, 2021.

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[23] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1578–1587, 2019.

[24] Xing Nie, Yongcheng Liu, Shaohong Chen, Jianlong Chang, Chunlei Huo, Gaofeng Meng, Qi Tian, Weiming Hu, and Chunhong Pan. Differentiable convolution search for point cloud processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7437–7446, 2021.

[25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[26] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[27] Shi Qiu, Saeed Anwar, and Nick Barnes. Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1757–1767, June 2021.

[28] Haoxi Ran, Wei Zhuo, Jun Liu, and Li Lu. Learning inner-group relations on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15477–15487, October 2021.

[29] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[31] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.

[32] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[33] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision*, pages 685–702. Springer, 2020.

[34] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.

[35] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[36] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European Conference on Computer Vision*, pages 660–676. Springer, 2020.

[37] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[38] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[39] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shenghui Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. *arXiv preprint arXiv:2207.04397*, 2022.

[40] Xiaoqing Ye, Jiamao Li, Hexiao Huang, Liang Du, and Xiaolin Zhang. 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 403–417, 2018.

[41] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pages 702–717. Springer, 2020.

[42] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. In *ACM SIGGRAPH 2004 Papers*, pages 644–651. 2004.

[43] Chris Zhang, Wenjie Luo, and Raquel Urtasun. Efficient convolutions for real-time semantic segmentation of 3d point clouds. In *2018 International Conference on 3D Vision (3DV)*, pages 399–408. IEEE, 2018.

[44] Cheng Zhang, Haocheng Wan, Xinyi Shen, and Zizhao Wu. Patchformer: An efficient point transformer with patch attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11799–11808, 2022.

[45] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020.

[46] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

[47] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020.

[48] Zixiang Zhou, Yang Zhang, and Hassan Foroosh. Panoptic-polarnet: Proposal-free lidar point cloud panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13194–13203, 2021.

[49] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.