AUTOMATED MOBILE ATTENTION KPCONV NETWORKS VIA A WIDE & DEEP PREDICTOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Kernel Point Convolution (KPConv) achieves cutting-edge performance on 3D point cloud applications. Unfortunately, the large size of KPConv network limits its usage in mobile scenarios. In addition, we observe that KPConv ignores the kernel relationship and treats each kernel point equally when formulating neighbor-kernel correlation via Euclidean distance. This leads to a weak representation power. To mitigate the above issues, we propose a module named Mobile Attention Kernel Point Convolution (MAKPConv) to improve the efficiency and quality of KPConv. MAKPConv employs a depthwise kernel to reduce resource consumption, and re-calibrates the contribution of kernel points towards each neighbor point via Neighbor-Kernel attention to improve representation power. Furthermore, we capitalize Inverted Residual Bottleneck (IRB) to craft a design space and employ a predictor-based Neural Architecture Search (NAS) approach to automate the design of efficient 3D networks based on MAKPConv. To fully exploit the immense design space via an accurate predictor, we identify the importance of carrying feature engineering on searchable features to improve neural architecture representations, and propose a Wide & Deep Predictor to unify dense and sparse neural architecture representations for lower error in performance prediction. Experimental evaluations show that our NAS-crafted MAKPConv network uses 96% fewer parameters on 3D point cloud classification and segmentation benchmarks with better performance. Compared with state-of-the-art NAS-crafted model SPVNAS, our NAS-crafted MAKPConv network achieves ∼1% better mIOU with 83% fewer parameters and 78% fewer Multiply-Accumulates.

1 INTRODUCTION

3D applications such as object recognition (Wu et al., 2015), shape segmentation (Yu et al., 2004), and indoor-scene segmentation (Armeni et al., 2016) are growing popularity thanks to the rise of 3D acquisition technologies and the increasing amount of geometric data collected from 3D sensors. Deep learning models are becoming one of the most common and promising ways to represent 3D data, especially 3D point cloud data, thanks to its high performance (Zhou et al., 2020; Zhao et al., 2020) and good efficiency (Cortinhal et al., 2020; Hu et al., 2020a).

Starting from PointNet (Qi et al., 2017a), point-based methods are becoming one of the most popular choices to extract features and make predictions for 3D input point clouds. Such an approach typically learns an individual representation for each data point and aggregates different representations to formulate a signature for all global points, usually via end-to-end point convolutions (Xu et al., 2018; Atzmon et al., 2018). Kernel point convolution (KPConv) (Thomas et al., 2019) is the first work that maps the input points to the influence of kernel points via linear correlation, and then further convolves the mapped features over the kernel points to compute weighted features.

Given a point y_i, the kernel function g over neighbor points y_i in KPConv carries $g(y_i) = \sum_{k<K} h(y_i, \tilde{x}_k)@W_k$ to learn representations via K kernel points, with the neighbor-kernel correlation $h(y_i, x_k)$. However, KPConv results in very large model sizes for 3D point cloud recognition due to inefficient local feature aggregation via point convolution over kernel weights W_k. As a result, KPConv takes up to ∼200G Multiply-Accumulates (MACs) to perform inference on a single scene at the cost of ∼200ms on a TITAN XP GPU. Such resource consumption and inference cost are too high for real-time execution in mobile scenarios. In addition, the above Neighbor-Kernel cor-

relation function has a strong hidden assumption: the contribution of all kernel points are measured equally towards the neighbor points. As a result of ignoring cross-kernel information from other kernel points to accommodate the joint evolution of different kernel points, such an assumption fails to enhance representation learning on important kernel points and thus yields sub-optimal results.

In this paper, we propose a novel 3D attention module, named Mobile Attention Kernel Point Convolution (MAKPConv), to address the above issues. MAKPConv employs two key design elements to improve the performance and efficiency. First, we propose a depthwise kernel to split feature extraction on point-wise features from channel-wise features during point convolution. Such design allows significant savings in both parameter count and amount of computation, without performance degradation. Second, we introduce Neighbor-Kernel attention (NK attention) to enable the kernel-wise information exchange between different kernel points via learnable attention networks. The joint evolution of all kernel points encourages representation learning on more important kernel points while suppressing less important ones, thus benefits from a better learning adaptability of kernel point contributions towards neighbor point distributions in KPConv networks.

Inspired by Sandler et al. (2018), we capitalize Inverted Residual Bottleneck (IRB) to encapsulate MAKPConv module into MAKPConv block, and craft MAKPConv design space to explore a wider range of choices in 3D models. As 3D architectures are usually high-dimensional, unstructured and unordered, tailoring a good neural architecture to address such complex point cloud data requires a delicate and precise evaluation protocol. This is difficult to achieve with weight-sharing one-shot NAS (e.g., SPVNAS (Tang et al., 2020)) because co-adaptation between sub-networks may degrade the quality of evaluation and lead to sub-optimal results. Thus, we pioneer the usage of predictor-based Neural Architecture Search (NAS) (Wen et al., 2020) that utilizes a sample-based mechanism to reduce evaluation noise and seek better models on 3D point cloud. Specifically, we train a performance predictor based on exploitable samples, and probe the whole MAKPConv design space to obtain the best NAS model based on predictive performance. We observe that existing predictor-based methods adopt a fixed scheme (e.g., one-hot encoding) to encode different types of features, leading to sub-optimal neural architecture representations with inaccurate priors. This is because real-valued features (e.g., block width) and categorical features (e.g., optional NK attention) have different priors and thus they should not be encoded in the same way. To address this issue, we formulate dense representations for features that have quasi-linear priors on model performance (e.g., depth, width), and formulate sparse representation for features that have non-linear sophisticated priors on model performance (e.g., optional attention). Intuited by Cheng et al. (2016), we devise a Wide & Deep predictor to interact dense and sparse representations such that the combination of memorization and generalization reduces the prediction error.

By utilizing the Wide & Deep predictor to perform NAS under the MAKPConv design space, our results on various benchmarks show promising results of the automatically discovered models in terms of both performance and efficiency. In particular, our MAKPConv networks outperform the original KPConv on both classification and semantic segmentation benchmarks with 92%$\sim$93% fewer parameters, thanks to the efficient design of MAKPConv block. Empowered by our Wide & Deep predictor driven NAS, the NAS-crafted networks outperform the state-of-the-art NAS work, SPVNAS, by $\sim$1 mIOU with 83% smaller size and 78% fewer MACs.

We highlight the contribution of this paper as follows:

- We propose Mobile Attention Kernel Point Convolution (MAKPConv) module that combines depthwise kernel with Neighbor-Kernel attention to achieve efficient and high-performing architectures dedicated to learning on 3D point clouds.
- We identify the importance of carrying feature engineering on neural architecture representations to improve predictor-based NAS. Specifically, we encode different searchable features with dense and/or sparse representations given their priors, and utilize a Wide & Deep neural predictor to interact these representations for better accuracy prediction.

2 Related Work

3D Point Convolution Networks. Point convolution networks (Xu et al., 2018; Hu et al., 2020a; Lang et al., 2020) extends convolutional operations towards irregular point set grids to benefit from learnable radial function and spherical harmonics in 3D point clouds. Later work (Thomas et al.,

2019) performs point convolution over radius neighbors and aggregates convolved results to produce learned representation without additional intermediate operators. Recent work on point convolutions improves point sampling efficiency (Hu et al., 2020a; Lang et al., 2020), local feature aggregations (Zhao et al., 2020; Lin et al., 2020), manual architecture fabrications (Hu et al., 2020b; Lei et al., 2020a), and/or improved training protocols (Goyal et al., 2021) to seek higher model performance and efficiency. However, these works neither simplify the structure of point networks to improve model efficiency, nor specify an accurate neighbor-kernel correlation. Our work extends kernel point convolution (Thomas et al., 2019) and addresses the above caveats: we improve the efficiency of local feature aggregation via a depthwise kernel on point convolution, and further introduce Neighbor-Kernel attention to enhance the representation power of a depthwise kernel.

Predictor-based Neural Architecture Search. Predictor-based NAS (Wen et al., 2020; Dudziak et al., 2020) is a popularized method that trains a performance predictor on limited exploitable 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 (Dudziak et al., 2020) and augmenting architecture samples (Liu et al., 2021), yet they overlook the importance of carrying feature engineering on neural architecture features. Our work identifies the importance of improving neural architecture representation, and assigns different priors to a wide set of searchable features and yield dense representations and sparse representations. Our work further designs a Wide & Deep predictor to leverage and interact these representations for better prediction quality on unseen architectures.

3 Mobile Attention Kernel Point Convolution

We start with the formulation of KPConv (Thomas et al., 2019). Given a center point $X \in \mathcal{R}^3$ and its corresponding features $\hat{F} \in \mathcal{R}^{D_{in}}$, a kernel point convolution (KPConv) function carries all N neighbor points $X_n = \{x' : ||x' - x|| < r\} \in R^{N \times 3}$ and their features $\bar{F} \in \mathbb{R}^{N \times D_{in}}$ within radius r over a set of K kernel points $\tilde{X} \in \mathcal{R}^{K \times 3}$ and accompanying weights W_k. Specifically, a correlation function: $h : \mathbb{R}^{N \times D_{in}} \rightarrow \mathbb{R}^{K \times D_{in}}$ maps the unordered neighbor point features $\bar{F}$ to a set of features F on kernel points. Usually, the correlation function is represented as $H \in \mathbb{R}^{N \times K}$ and applied via $F = H\bar{F}$. The correlation function is followed by a kernel $g : \mathbb{R}^{K \times D_{in}} \rightarrow \mathbb{R}^{D_{out}}$ that convolves these features to obtain the aggregated output F'. Here, D_{in} denotes the dimension of input, and D_{out} denotes the dimension of output. The original KPConv overlooks an important opportunity to improve the efficiency by reducing the redundancy of kernel g that is due to the simultaneously performed point-wise and channel-wise feature extractions. In addition, the original KPConv adopts a linear function as correlation: it treats the impact of each kernel point equally and ignores the adaptation of neighbor points towards kernel dispositions. For example, the center kernel point usually attracts more centralized neighbor points with a higher density, and edge kernel points perform critical jobs in recognizing boundary neighbor points with a lower density in the radius sampling neighborhood. Thus, the original KPConv may have sub-optimal performance and efficiency on 3D applications.

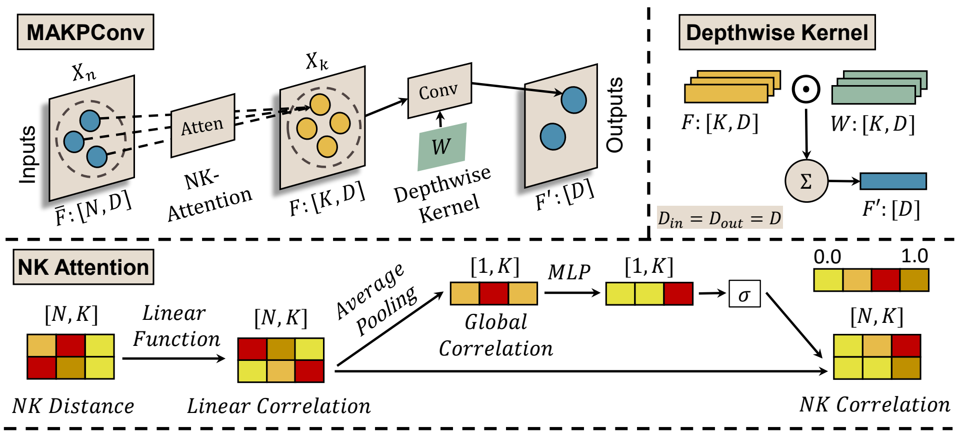

Figure 1: MAKPConv module combines depthwise kernel with Neighbor-Kernel attention to craft more efficient and high-performing 3D networks. Darker color indicates greater kernel contribution.

To overcome the above limitations, we propose Mobile Attention Kernel Point Convolution (MAKPConv) module as an in-place improvement of KPConv block to seek higher quality and efficiency. Figure 1 provides an overview of MAKPConv. Specifically, we design a depthwise kernel to address the computation cost: A depthwise kernel $\hat{g}$ carries element-wise multiplication in place of matrix multiplication between aggregated features and kernel weights, thus achieves memory and computation reduction. Considering a depthwise kernel has limited representation power, we introduce Neighbor-Kernel attention (NK attention) as an improved correlation function $\hat{h}$ to emphasize representation learning on important depthwise kernels and fully utilize the capacity of kernel points. This compensates for the reduced representation power: the joint evolution of kernel points via learnable attention encourages effective representation learning on important depthwise kernels and best exploit their potential. As a result, the fusion of depthwise kernel and NK attention into MAKPConv block has the potential to achieve high-performing yet efficient 3D models.

3.1 Depthwise Kernel

In original kernel function of KPConv, a regular kernel takes an input $F \in \mathbb{R}^{K \times D_{in}}$ over K kernel points and uses a weight kernel $W \in \mathbb{R}^{K \times D_{in} \times D_{out}}$ to simultaneously perform point-wise transformation and channel-wise transformation. A regular kernel g produces an output $F' \in R^{D_{out}}$:

$$F'_d = g(F, W) = \sum_{k \in K} \sum_{d \in D_{in}} F_{k,i} \cdot W_{k,i,d}, \tag{1}$$

where k is a kernel index, and d is a channel index. A regular kernel leads to $K \times D_{in} \times D_{out}$ parameters and $D_{in} \times D_{out} \times K$ MACs during inference, and leads to large computation and memory for the deep layers of the 3D architectures. To favor model efficiency, we design a depthwise kernel $\hat{g}$ that only performs point-wise feature extractions over input point clouds, see Figure 1(a). The proposed depthwise kernel adopts learnable weights $W \in \mathbb{R}^{K \times D_{in}}$, extracts point-wise information on each channel, and obtains an output $F' \in \mathbb{R}^{D_{out}}$ as follows:

$$F'_d = \hat{g}(F, W) = \sum_{k \in K} F_{k,d} \cdot W_{k,d}, \tag{2}$$

where k is a kernel index, and d is a channel index. Here, $D_{out} = D_{in} = D$ as a depthwise kernel does not operate on channel-wise features. As a result, a depthwise kernel point convolution only leads to $K \times D_{in}$ parameters and only $D_{in} \times K$ MACs during inference, and reduces $D_{out}\times$ parameters and MACs during convolution compared to a regular kernel. The parameter saving from a depthwise kernel is significant when D_{out} is large, especially in deeper layers of the network.

3.2 Neighbor-Kernel Attention

Although depthwise kernel $\hat{g}$ adds to model efficiency, such design lacks representation power to fit complex point cloud structures due to inactive channel-wise communication. To compensate for the loss, we propose Neighbor-Kernel attention (NK attention) $\hat{h} : \mathbb{R}^{N \times D_{in}} \to \mathbb{R}^{K \times D_{in}}$ that re-weights the contribution of kernel points by learning a joint representation $\hat{H} \in \mathbb{R}^{N \times K}$ on Neighbor-Kernel distances (NK distance) via an attention network, see the lower part of Figure 1. Such representation, when being applied to neighbor point features $\bar{F}$, can fully harness the capacity provided by different kernel points, thus provide better aggregated features F for the depthwise kernel.

NK attention first adopts a linear function to transform NK distance into linear NK correlation, and applies average pooling to obtain global neighborhood correlation. This global information captures rich information over the statistics of the locations of neighbor points and thus gives a high-level abstraction on the contribution of each kernel point towards the global neighborhood. Then, the global neighborhood correlation is fed into a 2-layer attention network to enable cross-kernel interaction and learn a good representation to quantify the contribution of kernel points, followed by a sigmoid gating to normalize the learned representation. Unlike Squeeze-and-excitation (Hu et al., 2018) , NK attention adopts a full-rank transformation to preserve the information needed towards every kernel with marginal parameter cost. Finally, the learned representation is scaled to the original NK linear correlation, followed by a bypass (i.e., addition) of the original linear NK correlation for easier optimization. As a result, NK attention grasps a richer representation $\hat{H}$ that incorporates the joint evolution of kernel points and encourages representation learning on important depthwise kernels, This leads to concrete performance improvement on 3D point-cloud benchmarks.

4 NERUAL ARCHITECTURE SEARCH WITH WIDE & DEEP PREDICTOR

In this section, we introduce a predictor-based Neural Architecture Search (NAS) driven by a novel Wide & Deep predictor, see Figure 2. MAKPConv module provides efficient motifs for processing point-wise features. To incorporate channel-wise feature extraction into MAKPConv module, we exploit the design of Inverted Residual Bottleneck (IRB) (Sandler et al., 2018) and sandwich MAKPConv module with 2 linear transformation to construct MAKPConv block. To design efficient yet high performing 3D models, we utilize MAKPConv block as an architecture motif and exploit predictor-based neural architectures with a wide set of features such as positional organizations, structural settings, and accessories. As formulating neural architecture representations from these features is the key in predictor-based NAS, we identify the importance of carrying engineering on searchable features. Thus, we assign different priors to these features to formulate dense architecture representations and sparse architecture representations, and propose a novel Wide & Deep neural predictor to effectively interact different sets of neural architecture representations for accurate predictions. Wide & Deep predictor trades-off generalization and memorization via dense deep models and sparse wide embeddings, thus reduce prediction MSE on unseen architectures.

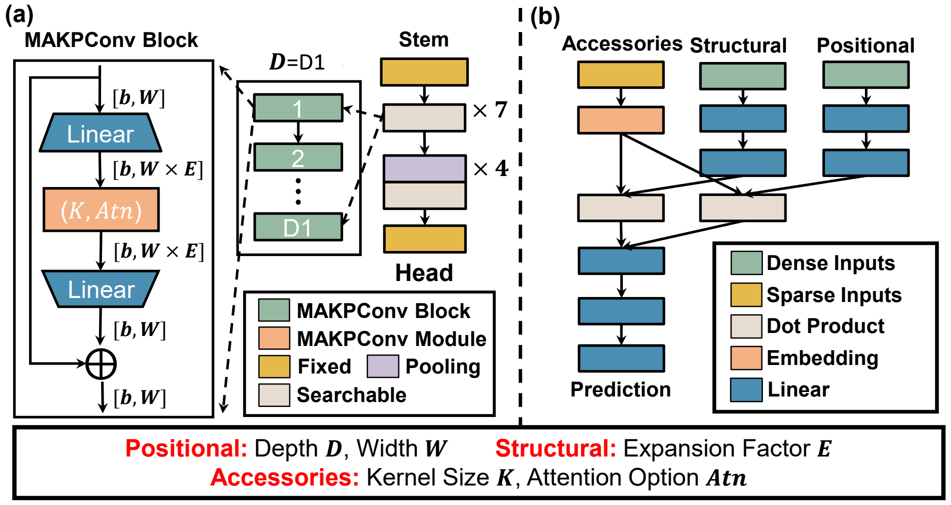

Figure 2: (a) Design space based on MAKPConv blocks. (b) Wide & Deep predictor.

4.1 MAKPCONV BLOCK

Inspired by Sandler et al. (2018), we sandwich MAKPConv by 2 linear transformation layers to derive MAKPConv block as a promising architecture motif to carry efficient and high performing 3D point cloud learning. It has a similar structure as Inverted Residual Bottleneck (IRB) and is described by four key parameters: expansion factor E, kernel size K, output width W, and optional choice of NK attention Atn. We follow the positional setting of MobileNetV2 to manually craft a MAKPConv network as a hand-crafted baseline to verify its performance and efficiency. We compare the performance and size between hand-crafted MAKPConv network and KPConv under the native KPConv protocol in Table 1. Compared with KPConv, the hand-crafted MAKPConv achieves on-par Overall Accuracy (OA) with at least 92% smaller size on ModelNet40, and achieves 0.4% higher mIOU with 93% smaller size on SemanticKITTI, even without NK attention. Such advantage is attributed to the efficient depthwise kernel. With NK attention, MAKPConv yields $\sim$0.4% OA gain on classification and $\sim$0.5% mIOU gain on semantic segmentation with marginal increase in parameter count. These results suggest that re-weighing the contribution of kernel points via learnable attention network enhances depthwise kernels thus improves performance.

Table 1: Performance of hand-crafted MAKPConv vs KPConv on ModelNet40 and SemanticKITTI.

Architecture	Cls. Params (M)	Seg. Params (M)	ModelNet40 OA (%)	SemanticKITTI mIOU (%)
KPConv (Our impl.)	14.9	14.8	92.3	59.2
MAKPConv (w.o. attention)	1.24	0.97	91.8	59.6
MAKPConv	1.25	0.98	92.2	60.1

4.2 MAKPConv Design Space

We utilize MAKPConv blocks to construct a flexible network design space. Specifically, we divide the overall 3D network into a stem KPConv for initial feature extraction, a head MLP for carrying fine-grained classification/segmentation, and several searchable stages. In each searchable stage, we jointly search for the optimal configuration of MAKPConv blocks, including the positional organizations, structural settings, and accessories. We elaborate these searchable features below.

Positional Organizations. The block width and depth of MAKPConv has a direct impact on the latency of MAKPConv models by affecting both computation-bounded operations (e.g., linear transformation) and memory-bounded operations (e.g., indexing). As deeper and wider architectures lead to better performance in 3D point cloud classification and segmentation following the Pareto frontiers, positional organizations have a positive quasi-linear impact towards performance and take continuous values on any positive integers.

Structural Settings. The expansion ratio affects the inner-structure of MAKPConv block that has a positive quasi-linear correlation with performance. However, co-adaption of expansion factor towards block width may induce challenges when learning a mapping rule to accurately predict architectures in the search space. Structural settings take continuous values on any positive integers.

Accessories. We mark the search of optional attentions and kernel size as accessories in the MAKPConv search space. Accessories have sophisticated impact on model performance. For example, as kernel point convolution utilize different dispositions for different number of kernels, it is unclear whether the change in the number of kernel points affects the quality of kernel disposition and the model performance. Both accessories can only take categorical values.

Considering the high variance in 3D point cloud tasks and the robustness to noisy evaluations, we adopt predictor-based NAS to explore MAKPConv design space. However, MAKPConv design space is prohibitively large: a total of 11 stages and 7 searchable features within each stage leads a total search space size of 1.8×10^{19} possible architectures. Such a large and complex search space calls for a high-performing predictor-based NAS to accurately model the design space.

4.3 Wide & Deep Predictor

Neural architecture representation is the key in the performance of predictor-based NAS due to its strong connection with predictor performance. Specifically, better neural architecture representation links searchable features towards priors that are more reflective to the final performance, and thus increases the chance to discover high-performing NAS models.

To improve neural architecture representations, We focus on carrying feature engineering to distinguish searchable features with different priors. Motivated by the aforementioned relationship between searchable features and the model performance, we assign different priors to a wide set of searchable features and craft dense and sparse neural architecture representations for corresponding features. Specifically, we formulate dense neural architecture representation for searchable features that can take continuous values (e.g., positional organizations, structural settings), and formulate sparse neural architecture representation for searchable features that can only take discrete values (e.g., accessories). With dense and sparse neural architecture representations, we follow the idea of Wide and Deep Models (Cheng et al., 2016) to design a Wide & Deep predictor that trades-off memorization and generalization in performance prediction, see Figure 2. Wide & Deep model adopts a separate MLP to obtain high-level dense outputs based on each of the 2 dense representations, and adopts an embedding table to convert sparse representations to continuous sparse outputs. To interact dense and sparse outputs, we leverage the DotProduct block in DLRM (Naumov et al., 2019) to fuse learned dense representations and sparse representations. Specifically, given a dense representation $D \in R^{B\times dim}$ and a sparse representation $S \in R^{B\times N\times dim}$, DotProduct carries cross-representation communication by carrying the following transformation:

$$X = [vec^{-1}_{B,1,dim}(D); S], DotProduct(D, S) = Triu(XX^T) \tag{3}$$

where dim represents the dimension of both dense and sparse representations, N denotes the number of sparse inputs, and B denotes the batch size. Eq. (3) first vectorizes the dense tensor D to have the same dimension as the sparse tensor S, and extracts the upper triangular matrix upon self matrix factorization. We carry 2 DotProducts on 2 separate pairs of dense and sparse representations to encourage the learning of independent interactions. This is because positional organizations and

Table 2: Wide & Deep Predictor achieves best prediction quality over SemanticKITTI.

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

structural settings may not necessarily have the same influence over the choice over accessories. After fusing dense and sparse representations via DotProduct, we utilize a MLP to learn predictions based on the fused representations in the Wide & Deep predictor.

4.4 Training and Evaluation of the Wide & Deep Predictor

Architecture Sampling. We randomly sample $\sim$1K architectures on both ModelNet40 and SemanticKITTI. Following the common NAS pipeline, we split the training dataset of ModelNet40/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) is the training objective of the Wide & Deep Predictor on 1K sampled architectures. To ensure a consistent comparison of predictor performance over multiple benchmarks, we carry optimization on predictors with normalized performance ranging from 0 to 1. Since the Wide & Deep Predictor can perform gradient-based optimization, we incorporate predictor pretraining on MACs (Dai et al., 2021) and rank loss (Dudziak et al., 2020).

We conduct extensive experiments on various top-performing NAS predictors collected from NASBench-301 (Siems et al., 2020). To justify that interacting dense and sparse representations benefits predictor performance, we establish "Wide" as a MLP Predictor that accepts sparse neural architecture representations transformed from categorical encoding of features, and establish "Deep" as a MLP predictor that accepts dense neural architecture representations transformed from real-valued encoding features. These two predictors also leverage rank loss (Dudziak et al., 2020) and predictor pretraining (Dai et al., 2021), thus are utilized as strong baselines for gradient-based NAS predictors to justify the significance of dense-sparse feature interaction. We list the supported training heuristics and compare the performance (i.e., MSE and Kendall τ) of Wide & Deep Predictor over various NAS predictors under 10 different runtimes. Compared with all non-gradient predictors (Chen & Guestrin, 2016; Ke et al., 2017; Duan et al., 2020), Wide & Deep predictor achieves lower MSE and higher Kendall τ in accuracy prediction, thanks to the potential of ranking-aware training and predictor pretraining via gradient-based optimization. Compared to gradient-based predictors using either dense or sparse features, Wide & Deep predictor achieves better prediction quality thanks to better architecture representation. Such improvement in predictor precision eventually yields the discovery of better NAS models, see the elaborations in Section 6.2.

5 Experiments

In this section, we evaluate MAKPConv over classification (ModelNet40 (Wu et al., 2015)) and semantic segmentation (SemanticKITTI (Behley et al., 2019)) benchmark.

5.1 Search Configuration

MAKPConv Design Space. Each block can choose from kernel size {5, 7, 13} to perform MAKPConv operation. Each block has up to 3 different choices for block depth, block width, and expansion factors. In addition, each block has the choice of an optional NK attention to balance performance and efficiency. We describe other details of the MAKPConv design space in Appendix.

Regularized Evolution. Following Real et al. (2019), we employ regularized evolution and our Wide & Deep predictor to effectively probe the MAKPConv design space. Given a candidate architecture A with predicted performance $\hat{P}$ 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 balance performance and resource consumption on both ModelNet40 and SemanticKITTI. We use a population of 200 and sample size of 150 to carry regularized evolution with 360 rounds over MAKPConv design space to craft NAS models.

5.2 Evaluation on 3D benchmarks

We use our Wide & Deep predictor to automate the design of MAKPConv blocks and discover promising models on ModelNet40 and SemanticKITTI 3D applications. To ensure a fair comparison with larger models on SemanticKitti, we apply width scaling (Sandler et al., 2018) on the NAS-crafted models and attach a $m\times$ suffix to denote the application of a $m\times$ width multiplier. We compare the performance and efficiency metrics such as parameter count and MACs.

ModelNet40. We follow the RSCNN protocol in SimpleView (Goyal et al., 2021) and show the evaluation results on the testing dataset of ModelNet40 in Table 3. For fair comparison, we report the mean accuracy across 5 runs and observe a 0.2% OA variance. Compared to KP-Conv, the hand-crafted MAKPConv achieves on-par OA with ~92% fewer parameters thanks to the proposed MAKPConv module. Our NAS approach explores more efficient model designs: the NAS-crafted MAKPConv achieves 0.5% higher OA than hand-crafted counterpart while being 55% smaller. Compared with efficient 3D models (Lei et al., 2020b; Goyal et al., 2021; Lin et al., 2020; Liu et al., 2019; Wang et al., 2019), NAS-crafted MAKPConv achieves on-par performance with ~30% fewer parameters. By further scaling up the model, our NAS-crafted MAKPConv outperforms its hand-crafted counterpart by 0.8% OA with a similar parameter count.

Table 3: Overall Accuracy (OA) on ModelNet40. *: Use native protocol instead of SimpleView RSCNN evaluation protocol.

Architecture	**Params (M)**	**OA (%)**
SPH3D-GCN* (Lei et al., 2020b)	0.8	92.1
FPConv* (Lin et al., 2020)	2.1	92.5
RSCNN (Liu et al., 2019)	1.3	92.5
DGCNN (Wang et al., 2019)	1.8	92.8
KPConv (Thomas et al., 2019)	14.9	92.9
SimpleView (Goyal et al., 2021)	0.80	93.2
PointNet++ (Qi et al., 2017b)	1.48	93.3
MAKPConv, hand-crafted	1.25	92.6
MAKPConv + NAS	**0.56**	93.1
MAKPConv + NAS, 2×	1.21	**93.4**

SemanticKITTI. Table 4 shows the mIOU comparison on sequence 08 of the SemanticKITTI dataset. Our hand-crafted MAKPConv 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 MAKPConv achieves ~ 1% higher mIOU than the state-of-the-art SPVNAS while saving 83% parameters and 78% MACs. Applying width multiplier on NAS-crafted model further boosts the performance. We elaborate on the latency comparison in Appendix A.7.

Table 4: mIOU on sequence 08 **(validation split)** of SemanticKITTI. Latency is measured with NVIDIA TITAN X Pascal on a single SemanticKITTI scene containing ~60K points. Here, red numbers denote computation time and blue numbers denote processing time. $^+$: Results from Zhou et al. (2021). *: Results from Tang et al. (2020).

Architecture	**Method**	**Params (M)**	**MACs (G)**	**Latency (ms)**	**mIOU (%)**
RandLANet (Hu et al., 2020a)	Point-based	1.24	-	103	57.1
KPConv-rigid (our impl.)	Point-based	14.8	60.9	221 (164 + 57)	59.2
PolarNet (Zhang et al., 2020)	Projection-based	13.6	135.0*	**62***	58.2$^+$
SalsaNext (Cortinhal et al., 2020)	Projection-based	6.7	62.8*	71*	59.0$^+$
MinkowskiNet (Choy et al., 2019)	Voxel-based	5.5	28.5*	294*	58.9
MAKPConv, hand-crafted	Point-based	**0.97**	**4.7**	160 (103 + 57)	**60.1**
SPVNAS (Tang et al., 2020)	Voxel-based	3.3	20.0	**158**	61.5
		7.0	34.7	**175**	63.5
MAKPConv + NAS	Point-based	**0.57**	**4.4**	169 (112 + 57)	62.4
MAKPConv + NAS, 2×		1.36	11.0	206 (149 + 57)	**64.1**

6 Ablation Studies

In this section, we first demonstrate how NK attention re-calibrates the contribution of each kernel point by visualizing the relationship between NK attention and NK linear correlation. Then, we discuss the relationship between predictor MSE reduction and quality improvement of the discovered NAS models to demonstrate the importance of predictor design in predictor-based NAS.

6.1 Kernel Contribution via NK Attention

Empirical evaluations reflect that NK attention leads to 0.4% OA gain and 0.5% mIOU gain on 3D object classification and semantic segmentation. To verify the source of improvement of MAKPConv over its non-attention counterpart and justify the re-calibration of kernel points, we plot the learned NK attention on 7 kernel points in Figure 3 and identify the contribution of each kernel via the slope. We observe that the learned attention for each kernel point has a strong relationship with kernel dispositions (i.e., 1-4-2 groups among symmetrical axis) in the Octahedron. The two vertices that are the most distant from the center point shows the most contribution towards neighbor points. This is because these kernel points are involved in addressing the boundary effects of input point clouds for more robust recognition results. We also observe that the 4 vertices connecting the square plane has the least contribution, illustrating that such high concentration of kernels causes redundancy and suggests a future direction in optimizing kernel location to fully capitalize the kernel points to improve representation power. The center kernel point admits most information from centralized neighbor points, thus co-adapts to global neighbor information and yields an moderate contribution towards neighbor points.

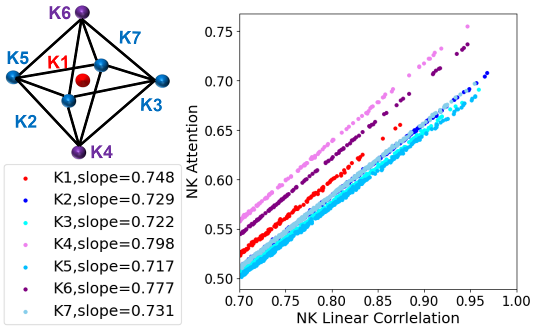

Figure 3: Visualization of learned NK attention matches the octahedron disposition of kernels.

6.2 Effectiveness of Wide & Deep Predictor

We show that even marginal MSE reduction (i.e., 0.07×10^{-2}) leads to significant quality improvement of NAS-crafted models, which justifies the importance of feature and architecture engineering in predictors. We establish 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 top-performing Wide NN predictor in Table 5. 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 ModelNet40 and SemanticKITTI in Table 5.

Table 5: 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	**ModelNet40 OA (%)**				**SemanticKITTI mIOU (%)**			
	mean	**std**	**max**	**min**	**mean**	**std**	**max**	**min**
Random	90.96	0.47	91.5	90.2	53.54	1.19	54.68	51.96
Predictor-based + Wide	91.14	0.23	91.5	90.9	54.90	0.49	55.28	54.2
Predictor-based + Wide & Deep	**91.22**	0.16	91.4	91.1	**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 Wide & Deep predictor can outperform NAS-crafted models via Wide predictor by a large margin (i.e., up to 1.2% mIOU on SemanticKITTI). The above results emphasize the need to improve predictor-based NAS via a delicate neural predictor design.

7 Conclusion

In this work, we present MAKPConv module to improve both efficiency and performance of KPConv. MAKPConv employs depthwise KPConv that separates point-wise feature extraction from channel-wise feature extraction to reduce computation cost and memory consumption, and NK attention that re-weighs the contribution of different kernel points via interactions of neighbor-kernel correlation through an learnable attention network. We further incorporate the IRB design to construct MAKPConv block and propose a new predictor-based NAS driven Wide & Deep predictor to automate the model design. Experiments and ablation study over 3D classification and 3D semantic segmentation benchmarks verify the effectiveness of MAKPConv.

8 Reproducibility Statement.

We make tremendous efforts to ensure reproducibility of our proposed method. Specifically, we mention the detailed configuration of predictor-based NAS in Section 5.1, introduce the design of hand-crafted MAKPConv design in Appendix A.1/A.2, describe the detailed configuration of Wide & Deep predictor in Appendix A.3, show the transferability of searched architecture in Appendix A.4, cover the training protocols in Appendix A.5, and use Appendix A.6 to justify orthogonal contribution of our improving performance predictor and other line of research in improving search algorithms. In addition, we use Appendix A.7 to give an analysis on the latency breakdown of searched models and points an direction on further improving the efficiency of MAKPConv kernel to seek higher efficiency in 3D point cloud applications. Code that covers other details of this work is available via this link.

References

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*, pp. 1534–1543, 2016.

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

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*, pp. 9297–9307, 2019.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

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*, pp. 7–10, 2016.

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*, pp. 3075–3084, 2019.

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.

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*, pp. 16276–16285, 2021.

Tony Duan, Avati Anand, Daisy Yi Ding, Khanh K Thai, Sanjay Basu, Andrew Ng, and Alejandro Schuler. Ngboost: Natural gradient boosting for probabilistic prediction. In *International Conference on Machine Learning*, pp. 2690–2700. PMLR, 2020.

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.

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)*, pp. 14504–14513, June 2021.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

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.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

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*, pp. 11108–11117, 2020a.

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*, pp. 222–239. Springer, 2020b.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

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*, pp. 7578–7588, 2020.

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*, pp. 11611–11620, 2020a.

Huan Lei, Naveed Akhtar, and Ajmal Mian. Spherical kernel for efficient graph convolution on 3d point clouds. *IEEE transactions on pattern analysis and machine intelligence*, 2020b.

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*, pp. 4293–4302, 2020.

Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019.

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

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

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv e-prints*, pp. arXiv–1906, 2019.

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*, pp. 652–660, 2017a.

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*, pp. 5099–5108, 2017b.

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)*, pp. 1757–1767, June 2021.

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)*, pp. 15477–15487, October 2021.

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, pp. 4780–4789, 2019.

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*, pp. 4510–4520, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

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*, pp. 685–702. Springer, 2020.

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*, pp. 6411–6420, 2019.

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.

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*, pp. 660–676. Springer, 2020.

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*, pp. 1912–1920, 2015.

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)*, pp. 87–102, 2018.

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)*, pp. 403–417, 2018.

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*, pp. 644–651. 2004.

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)*, pp. 399–408. IEEE, 2018.

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*, pp. 9601–9610, 2020.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020.

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.

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*, pp. 13194–13203, 2021.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A Appendix

In this appendix, we elaborate the detailed structure of hand-crafted MAKPConv via MAKPConv blocks. We also verify the transferability of NAS discovered architecture on S3DIS (Armeni et al., 2016) indoor scene segmentation benchmark. Furthermore, we discuss detailed hyperparameter settings for training and evaluating 3D architectures on ModelNet40, S3DIS, and SemanticKITTI benchmarks, and visualize the optimization curve in the search process for both vanilla predictor-based NAS (i.e., via Wide Predictor) and our improved predictor-based NAS (i.e., via Wide & Deep predictor). We also discuss the latency breakdown of our searched models and propose some future work in improving the efficiency of MAKPConv series models.

A.1 Hand-crafted MAKPConv network

Table 6: Structure of hand-crafted MAKPConv. "1/2" strides means upsampling by 2×. Note that stage 1∼7 is utilized to perform classification, and stage 8∼11 is utilized to perform semantic segmentation.

Stage	**Depth**	**Stride**	**In Width**	**Out Width**	**E**	**K**
1	1	1	16	16	1	7
2	2	2	16	24	3	7
3	3	2	24	32	3	7
4	4	2	32	64	3	7
5	3	1	64	96	3	7
6	3	2	96	160	3	7
7	1	1	160	320	3	7
8	1	1/2	416	160	3	7
9	1	1/2	160	96	3	7
10	1	1/2	96	64	3	7
11	1	1/2	64	32	3	7

Table 6 illustrates the detailed architecture of our hand-crafted MAKPConv. The hand-crafted MAKPConv contains a total of 17 MAKPConv blocks for classiciation, and 4 additional MAKPConv blocks for semantic segmentation. The positional setting (i.e., depth, block width) of hand-crafted MAKPConv strictly follows MobileNetV2, which is considered as the state-of-the-art hand-crafted model with remarkable efficiency. For structural settings, we employ an expansion factor of 3 for all MAKPConv layers. In addition, we leverage a kernel size of 7 for each MAKPConv block for design efficiency. As a result, MAKPConv serves as a strong baseline to compare with when evaluating the performance of NAS-crafted MAKPConv models.

A.2 MAKPConv Design Space

We present the detailed settings of the MAKPConv design space in Table 7. Despite the search of block depth, block width, expansion factor and kernel size, we also incorporate the search of optional attention into MAKPConv design space to trade-off the overhead in Neighbor-Kernel Attention and its performance gain.

Table 7: Configuration of the MAKPConv design space. "1/2" strides means upsampling by 2×.

Hierarchy	**Stage**	**Strides**	**Depth**	**Kernel Size** (K)	**Expansion (E)**	**Width**
Classification Backbone	**1**	1	1	5, 7, 13	1	16
	2	2	2, 3	5, 7, 13	2, 3, 4	16, 24
	3	2	2, 3, 4	5, 7, 13	2, 3, 4	24, 32
	4	2	3, 4, 5	5, 7, 13	2, 3, 4	24, 32, 40
	5	1	2, 3, 4	5, 7, 13	2, 3, 4	40, 56, 72
	6	2	3, 4, 5	5, 7, 13	2, 3, 4	64, 80, 96
	7	1	1	5, 7, 13	2, 3, 4	160
Segmentation Head	**8**	1/2	1	5, 7, 13	2, 3	64, 80, 96
	9	1/2	1	5, 7, 13	2, 3	40, 56, 72
	10	1/2	1	5, 7, 13	2, 3	24, 32, 40
	11	1/2	1	5, 7, 13	2, 3	16, 24

A.3 Design of Wide & Deep predictor

We configure the architectural hyperparameters of our wide & deep predictor via intuition without delicate architecture engineering. Specifically, we employ a 3-layer MLP with 64-128-256 layers as the dense architecture to extract dense neural architecture representations for both positional organizations and structural settings. The overall architecture that is responsible for processing fused features after dense-sparse interaction is a 2-layer MLP with 256-256 layers. We utilize ReLU as the activation function and apply Dropout of 0.5 before the final regression head to mitigate overfitting. To ensure fair comparison, both the Wide predictor and the Deep predictor also adopt a 2-layer MLP with 256-128 units to keep the same maximum projection dimension as the Wide & Deep predictor.

A.4 ARCHITECTURE TRANSFERABILITY ON S3DIS

We further verify the transferability of NAS-crafted MAKPConv architectures 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$\sim$Area 6 splits, see Table 8. 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 9. Though our NAS-crafted model is not optimized in S3DIS dataset, it shows a significant margin on S3DIS 6-fold cross-validation benchmark compared with existing approaches such as Qiu et al. (2021); Zhao et al. (2020) and achieves 74.5 mIOU, a new state-of-the-art result using only 1.35M parameters. On Area 5 of the S3DIS dataset, we observe that: (1) Our NAS-crafted MAKPConv achieves 1.8% higher mIOU than original KPConv with 11$\times$ parameter efficiency. (2) Our NAS-crafted MAKPConv achieves competitive performance over different classes on S3DIS indoor semantic segmentation and demonstrate $\sim$1% higher mIOU over 9 out of 13 classes compared to existing approaches. Thus, our NAS-crafted models have a good transferability towards alternative benchmarks, and our NAS approach can be further incorporated into other 3D point cloud tasks such as nuScenes (Caesar et al., 2019), KITTI (Geiger et al., 2013) to improve the current state-of-the-art models.

Table 8: 6-fold cross-validation results on S3DIS.

Architecture	**mIOU (%)**	**mAcc(%)**	**OA(%)**	**Params (M)**
RandLANet (Hu et al., 2020a)	70.0	82.0	88.0	1.24
KPConv (Thomas et al., 2019)	70.6	79.1	-	14.8
RPNet (Ran et al., 2021)	70.8	-	-	-
SCF-Net (Fan et al., 2021)	71.6	82.7	88.4	-
BAAF-Net (Qiu et al., 2021)	72.2	**83.1**	88.9	**1.23**
PointTransformer (Zhao et al., 2020)	73.5	81.9	90.2	4.9
MAKPConv + NAS, 2$\times$	**74.4**	82.1	**90.3**	1.35

Table 9: mIoU per class on S3DIS Area-5.

Methods	**mIoU**	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book.	sofa	board	clut.
Pointnet (Qi et al., 2017a)	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 (Zhang et al., 2018)	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 (Ye et al., 2018)	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 (Thomas et al., 2019)	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
MAKPConv + NAS, 2$\times$	**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**

A.5 HYPERPARAMETER SETTINGS

The implementation of our work is based upon the official PyTorch repository of KPConv (Thomas et al., 2019). We follow the exact evaluation protocol in this repo to produce our validation results. For training, we demonstrate the detailed hyperparameter settings on 3 different benchmarks (i.e., ModelNet40 (Wu et al., 2015), SemanticKITTI (Behley et al., 2019) and S3DIS (Armeni et al., 2016)) in the following context.

ModelNet40. A single training batch in ModelNet40 contains 16 sub-sampled point clouds. We adopt a similar training pipeline following the original KPConv (Thomas et al., 2019) paper yet incorporates SimpleView (Goyal et al., 2021) RSCNN training protocol for fair comparison. Specifically, the SimpleView-RSCNN protocol adopts random scaling & translation for data augmentation, employ cross-entropy loss to optimize, and utilize a voting scheme on the best test model to best exploit the potential of searched model. We train our NAS-crafted model for 250 epochs with an initial learning rate of 0.02 and cosine learning rate schedule (Loshchilov & Hutter, 2016). To combat overfitting, we also adopt a dropout rate of 0.5, a L2 weight decay of 3e-4 and employ early stopping to select the best model for testing.

S3DIS. A single training batch in S3DIS contains 8 sub-sampled point clouds. We use a downsample rate of $dl0 = 0.04m$ following the original KPConv paper (Thomas et al., 2019). Specifically, we train our best model for 250 epochs with an initial learning rate of 0.04 and cosine learning rate schedule (Loshchilov & Hutter, 2016). To combat overfitting, We also adopt a L2 weight decay of 3e-4 and the default data augmentation in KPConv paper.

SemanticKITTI. A single training batch in SemanticKITTI contains 10 sub-sampled point clouds. On SemanticKITTI, we measure Multiply-Accumulates (MACs) over a scene with an average of $\sim$12.3K points, which gives similar MAC count for KPConv architecture as is shown in (Tang et al., 2020). We use a downsample rate of $dl0 = 0.06m$ following the original KPConv paper (Thomas et al., 2019). Specifically, we train our best model for 250 epochs with an initial learning rate of 0.04 and cosine learning rate schedule (Loshchilov & Hutter, 2016). To combat overfitting, We also adopt a L2 weight decay of 3e-4 and the default data augmentation in KPConv paper.

A.6 Optimization curve on Different Predictors

In this section, we demonstrate that the improvements of the proposed predictors has no interference over the improvements in search algorithms. More specifically, our contribution towards leveraging a better neural architecture representation is orthogonal to the contribution in introducing a better search algorithm. We apply 3 popular search algorithms: Random Search, Regularized Evolution (Real et al., 2019) and Reinforcement-Learning via Proximal Policy Optimization (PPO) (Zoph & Le, 2016; Schulman et al., 2017) on both Wide predictor and Wide & Deep predictor, and plot the normalized performance for visualization, see Figure 4. Despite the greater complexity of Wide & Deep predictor that may potentially formulate a curvy performance surface, we observe that the superiority of EA on Wide predictor is successfully preserved on Wide & Deep predictor. This indicates that the improvement of Wide & Deep predictor is consistent with the evaluation of different search algorithm, and sets up a new direction of improving the design of neural predictors to enhance predictor-based NAS.

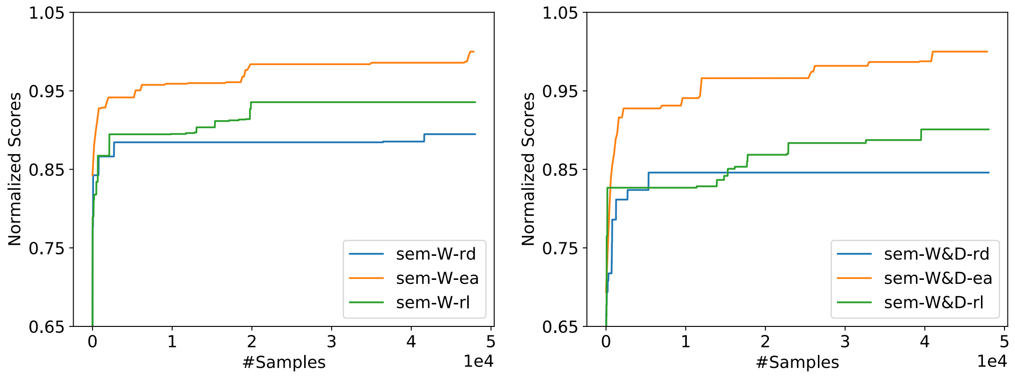

Figure 4: Left: Normalized prediction scores using Wide predictor (donated as W), Right: Normalized prediction scores using Wide & Deep predictor (donated as W&D).

A.7 Latency Analysis of MAKPConv Kernel

In Table 4, we notice that our NAS-crafted MAKPConv model is 24% 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 MAKPConv models, especially compared to voxel-based methods. This is because voxel-based methods (e.g., SPVNAS (Tang et al., 2020)) benefit from 1) reduced overhead in point-based neighboring mechanisms 2) mature software-hardware co-design that enables inference with high throughput. To dive deep into this issue, we profile the latency breakdown of the NAS-crafted MAKPConv model and demonstrate the profiling result in Table 10.

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`)

Table 10: Operator-level latency breakdown of our searched model. We list the critical operators here and categorize less important operator as 'Other'. In column **Type**, **C** denotes computation-bounded operation and **M** denotes memory-bounded operation.

Operator	**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

and point-wise local neighborhood movement (i.e., `aten` :: `sub`) contribute to $\sim$50% of the total inference cost. Although radius sampling still consumes moderate cost in MAKPConv, there is still a need to leverage design automation algorithm (e.g., our proposed MAKPConv algorithm) to perform optimization on the overall architecture on MAKPConv and seek better inference efficiency by improving computation-bounded operations.

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

- Efficient radius sampling implementation. The current implementation lies in CPU and is not optimal for parallelization. A possible CUDA version of the radius sampling can speedup the inference speed of MAKPConv networks.
- The current MAKPConv does not utilize a fused version of convolution that aggregates the feature outputs on each kernel point efficiently. Instead, the output features on each kernel point is firstly computed via `aten` :: `bmm` and then aggregated via `aten` :: `sum`.
- The current process of moving points towards their local neighborhoods controlled by $aten :: sub$ is far from efficient and consumes $\sim$ 20% latency. Thus, a specialized hardware for efficient batch points movement is potentially beneficial for efficient inference on point-based networks.
- As lower radius leads to more efficient sampling, we consider incorporating the choice of radius for each MAKPConv layer into the design space and harness such option to discover more efficient point-based 3D networks.

As such, with heavy optimization on MAKPConv design, we can reduce the uncertainty and/or complexity in profiling MAKPConv blocks and thus push forward latency-aware search for better hardware specialization. We leave this as our future work to improve the design efficiency of 3D architectures on processing point cloud data.