Fused Sampling and Grouping with Search Space Reduction for Efficient Point Cloud Acceleration

Hyunsung Yoon
Pohang University of Science and Technology
Pohang, Republic of Korea
hyunsung.yoon@postech.ac.kr

Jae-Joon Kim
Seoul National University
Seoul, Republic of Korea
kimjaejoon@snu.ac.kr

ABSTRACT

Recently, point-based deep neural networks (DNN) have demonstrated remarkable ability in analyzing point cloud data. However, challenges arise in sampling and grouping layers, particularly in terms of time and energy consumption due to the iterative access and computation of point cloud data for local feature extraction. In this paper, we introduce a Morton code-based data structure which stores point data with the shared upper bits together, enabling sequential access to the points within a specific voxel. We also propose a fused sampling and grouping approach with a reduced search space, which reuses the point data and the calculated distances for the farthest voxel and its neighbors. Additionally, a dedicated hardware architecture is introduced to maximize the efficiency of the proposed optimization technique. Experimental results show that our approach effectively reduces the number of distance calculations and data accesses with negligible accuracy loss, without requiring retraining of the network model.

CCS CONCEPTS

• **Hardware** → *Emerging architectures*; • **Computer systems organization** → *Neural networks*; • **Computing methodologies** → 3D imaging.

KEYWORDS

Point cloud, farthest point sampling, ball query, interpolation, hardware accelerator, co-optimization

ACM Reference Format:
Hyunsung Yoon and Jae-Joon Kim. 2024. Fused Sampling and Grouping with Search Space Reduction for Efficient Point Cloud Acceleration. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3655940

1 INTRODUCTION

Point cloud is a commonly used data format which represents 3D information in the form of points having coordinates and some additional aspects. With the rising prominence of three-dimensional (3D) applications such as autonomous driving, augment reality, and virtual reality, there is a growing emphasis on the deep neural networks (DNN) which extract features from 3D data.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3655940

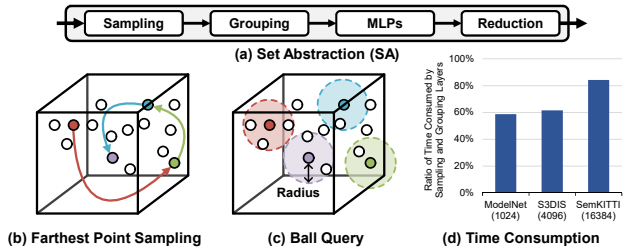

Figure 1: Farthest point sampling and ball query are employed in sampling and grouping layers of set abstraction. They consume more than 50% of the inference time on the general-purpose hardware, RTX-3090.

While DNNs have demonstrated remarkable performance in image feature extraction, earlier works faced challenges when directly applying them to derive 3D information from point cloud, resulting in transformation onto another format[6]. This complication arises from the unordered characteristic of point cloud, hindering the straightforward application of regularized kernels, such as convolution, which are traditionally effective in structured data formats.

PointNet++ [7], a well-known point-based DNN model, addresses the challenge of capturing local information within point cloud data through set abstraction (Fig.1a). Farthest point sampling (Fig.1b) and ball query (Fig.1c) are commonly employed for sampling and grouping layers. Farthest point sampling repeatedly selects a point which is distant from the previous centroids, ensuring uniform coverage. Ball query identifies neighbors within the radius boundary of the sampled points. MLPs and reduction layers play a role of deriving feature data from this region.

While sampling and grouping successfully address the challenge of feature extraction and are applied to later works[8], a new issue emerges. The repetitive memory access and distance computations required for the entire points during the sampling and grouping process result in these operations occupying more than 50% of the inference time on general-purpose hardware, as shown in Fig.1d.

With the enhanced performance of computing elements used for feature computation, the acceleration of sampling and grouping layers becomes indispensable for the efficient processing of point cloud DNNs. To address the overhead from iterative memory access and calculation during bottleneck operations, this paper introduces a Morton code-based data structure designed to exploit the spatial locality of point cloud, utilizing a two-level information. With the proposed data structure, we also propose fused sampling and grouping with reduced search space. This scheme determines the search space using a voxel-level distance, and maximize reuse of point data and calculated point-level distances of selected search space through integration of sampling and grouping. Additionally,

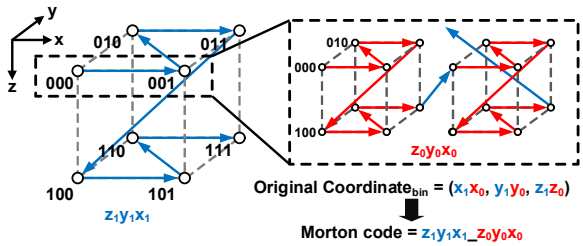

Figure 2: Morton encoding maps 3D information into 1D by interleaving bits of integer coordinate.

a dedicated hardware design supports search space reduction. The reduced search window, including neighbor voxels, has a negligible effect on accuracy, up to 0.4%, without requiring retraining of the model. Our scheme effectively reduces the number of operations and memory accesses, leveraging a relatively structured data access pattern.

2 BACKGROUND

2.1 Morton Code on Point Cloud Processing

Morton code, also know as Z-order curve, is a method that maps multi-dimensional data into a 1D format with a specific order, playing a crucial role in preserving the spatial locality of point cloud data. Morton encoding achieves this by interleaving each bit of integer coordinates, as illustrated in Fig.2. For example, if the original coordinates of a point are represented as (001, 110, 101) in binary, Morton code for this point becomes 110_010_101. This encoding can be visualized as a hierarchy of voxels. The significance lies in the fact that nearby points tend to have similar Morton codes, facilitating the structured organization of point cloud data.

Several works have been introduced to utilize the spatial locality achieved through Morton code for point cloud processing. For instance, in [12], a Morton code-assisted point cloud compression approach is proposed for edge devices. This method identifies similar points based on geometric relations. Another approach, EdgePC [11] reordered point cloud data after Morton encoding to approximate sampling and grouping results with fewer operations on edge devices. While approximation with sorted Morton code demonstrated improvements in latency, there was a trade-off with accuracy degradation even after retraining the model.

2.2 Accelerators for Point-based DNN

With the increasing attention on point cloud DNNs, a variety of hardware accelerators and corresponding software optimization techniques have been introduced. Some of these approaches are focusing on the process of finding geometric relations, such as sampling and grouping, which are well-known performance bottlenecks. Concurrently, there are architectures designed for general acceleration of point cloud processing.

Mesorasi [4] proposed delayed-aggregation to reduce the overhead of feature computation. PointAcc [5] introduced a hardware architecture designed for general acceleration of operations related to point cloud DNNs. While the proposed architecture demonstrated notable efficiency due to custom design and data flow, operations on the hardware require consideration of all input points, leaving a chance of further algorithmic optimization. Point-X [13] accelerates

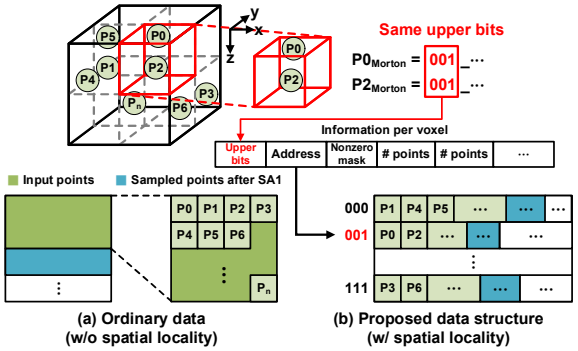

Figure 3: Proposed data structure using the upper bits of Morton code to divide 3D space. Grouped point data posses spatial locality.

graph construction by clustering nearby points and introduces a network-on-chip (NoC) and data flow optimized for clustered data. However, its application is limited to graph-based DNN models. Crescent [3] reduces DRAM traffic during neighbor search operations using a two-level tree data structure. However, there is a limitation such that the proposed tree structure, based on the k-d tree, requires tree construction at every layer, and retraining is necessary to maintain accuracy. MARS [10] mainly discusses the inefficiency of the systolic array employed as a matrix multiplication unit in PointAcc and proposes a reconfigurable systolic array. It also introduces grid-based group filtering to reduce the number of operations. While MARS reduces the number of distance updates, it still needs to compare distances of all points to find the next centroid. TiPU [14] introduces parallel sampling with a grid-based approach, but the proposed approach is optimized for a limited dataset. Another limitation is that, since the window size of sampling and grouping differs, the proposed distance reusing scheme addresses only a portion of the desired number of computations.

3 FUSED SAMPLING AND GROUPING WITH REDUCED SEARCH SPACE

To achieve efficient sampling and grouping, we introduce a Morton code-based data structure designed to exploit spatial locality. Our proposed method, called fused sampling and grouping with search space reduction, utilizes this data structure. The primary objective of our scheme is to eliminate the requirements to check all input points throughout the entire sampling and grouping operations. This is achieved while maximizing data reuse with the same search window for both operations.

3.1 Morton-based Data Structure

Morton-encoding projects 3D information into 1D space with a specific order, implying spatial relationships. To enhance efficient data movement and computation using locality, we introduce a data structure based on the Morton code.

Fig.3 describes the proposed data structure, which organizes nearby points together. The arrangement is based on the shared upper bits in the Morton code, gathering input points depending on their own upper bits. While each voxel has a specific order, the points in each space remain unordered without further arrangement, such as sorting, preserving unordered characteristics of the

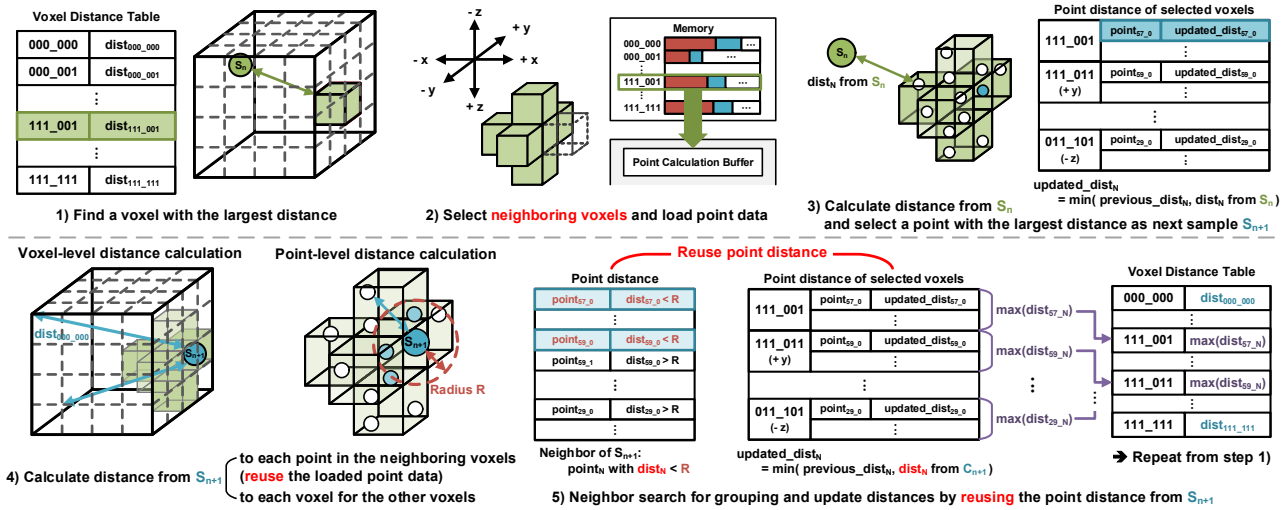

Figure 4: Processing order of the proposed fused sampling and grouping with search space reduction. After sampling with the reduced search space (step 1–3), point distance calculation occurs by reusing the fetched point data (step 4). These point distances are shared for two operations of the neighbor search and voxel distance update (step 5).

original data. This approach allows for capturing local information in various directions, not limited to a particular direction caused by 1D projection.

The proposed data structure consists of two-levels information: voxel-level information and point-level information. Voxel-level metadata includes the address where corresponding points data starts and an abstract of points in each voxel sharing the upper bits. A nonzero mask is a bit-mask that indicates the existence of point at individual levels. If the voxel contains points, the point data is accessed sequentially using the number of points at each level.

Considering the original coordinates are stored in floating-point format, it is necessary to transform the coordinate information to unsigned integers. This transformation involves making the range of coordinates positive by subtracting the minimum range value of the 3D space. Subsequently, division by the resolution of the data, representing the minimum distance between points, approximates the position to integers, ensuring the retention of positional information. As all points undergo subtraction and division, it is noteworthy that performing MLP on the normalized coordinates become straightforward without loss, accomplished simply by multiplying the resolution to the corresponding weights.

The utilization of Morton code-based data structure presents several advantages beyond serial memory access for specific point groups. By leveraging the upper bits of each point, the proposed method enables the storage of newly sampled points in the same manner without requiring additional processes, necessitating only a single-time encoding before the inference starts. From a hardware perspective, the replacement of floating-point computations for spatial relationships with simpler integer operations offers efficiency gains in area and energy consumption.

3.2 Fused Sampling and Grouping

Search space reduction stands out as a prominent method for efficient point cloud processing. Considering that farthest point sampling involves finding the farthest elements from all previous centorids, an insightful approach is to identify the next centroid within the farthest voxel. Based on this insight, we propose a fused sampling and grouping method based on the proposed data structure. This method strategically reduces the search space for both sampling and neighbor search from the entire points to those within particular voxels while maximizing data reuse.

Fig.4 illustrates the processing steps of the fused sampling and grouping method. The process begins with the selection of a voxel with the largest distance from the voxel distance table (Step.1). Following this, the point data of the selected voxel and its neighbors are loaded, and the sampling step ensues, following the same procedure as the original method. This includes distance calculation from the centroid to the fetched points and the subsequent update of distances (Step.2, 3). To select the next candidate voxel, two types of distances are computed from the new centroid: point-level distance by reusing fetched data of neighboring voxels and voxel-level distance for the other voxels (Step.4). While voxel-level distances are compared with the previous voxel-level distances, point-level distances are compared with the previous ones. Simultaneously, neighbor search occurs by reusing point-level distances for comparison with the radius. The maximum distance in each selected voxel is updated in the voxel-distance table for the next voxel selection (Step.5). If the next candidate voxel is the same as the previous one, step 2 and 3 can be skipped since the distance from the previous centroid is already updated in the table.

The use of both types of distance is a crucial part of the propsoed method. When determining the voxel in which the next farthest point is located, the naive approach with the proposed data structure would involve utilizing distances between voxels in the same procedure used in the original point-level operation. However, a challenge arises in that, every voxel would have a distance of zero before finding a sufficient number of centroids, primarily because the number of voxels containing any points is usually smaller than the number of required centroids. While the distance between the centroid and voxel can address this situation, it introduces a new issue that the voxel of new centroid provides information about how far the new sample is from representative point of the voxel.

This can lead to a selection of voxels having points far from a specific point, rather than from the input points. In contrast, utilizing point-level distance provides the actual relationship between points, ensuring a better quality of decision.

The extension of the search window to include neighbors is an enhancement for better candidate selection. Voxel-level positions do not provide precise information about the points inside, leading to the possibility that points near the surface of a voxel might be closer to a previous centroid in another voxel than to points in the same voxel, resulting in a large distance. When such points are selected as centroids, there is a risk of harming the uniformity of sampled points, leading to potential accuracy loss. Moreover, the grouping operation may fail to identify these points even though they are in the actual neighborhood. To address this, we expand the search space in both forward and backward directions in each coordinate. Although the window size is set at 7, it is relatively small, considering that the number of entire voxels becomes 64 in the case of using only 2-bits per coordinate to divide space.

We implemented the proposed sampling and grouping in the earlier layers with a large number of input points to ensure the maintenance of network accuracy. In cases where the number of points is smaller than the number of voxels, the reduced search window may not guarantee the required number of neighbor points. In such instances, the sampling and grouping occur in the original manner, with the data structure maintained. Note that this optimization has a negligible effect on the overall efficiency of the proposed scheme, as the overhead of sampling and grouping mainly stems from the earlier layers with a large number of point, where the proposed scheme can be applied.

3.3 Expansion to Feature Propagation

The proposed search space reduction is not confined solely to the set abstraction layer but can also be applied to the feature propagation layer, which involves finding the nearest centroids of every skip-connected points. Considering that the number of sampled points required for interpolation is relatively small, we can reduce the search space in a manner similar to that used in the set abstraction. The proposed data structure offers more optimization opportunities at this stage. Sequential storage of input and output points of sampling within the same voxel allows easy data fetch for a specific region using the voxel-level information. Additionally, handling voxels with Morton-order enables the reuse of fetched point data for the next search iteration. In short, the benefits of the proposed scheme extend beyond the set abstraction, enhancing overall efficiency across various layers.

4 HARDWARE IMPLEMENTATION

We designed a hardware architecture supporting the reduction of search space (Fig.5). The accelerator consists of three main units, each responsible for handling voxel-level operations, point-level operations, and feature computation. Since the proposed data structure is based on the integer format, most components, except the feature computation part, support integer operations. This design choice results in a more area-efficient implementation compared to floating-point units supporting the same computations.

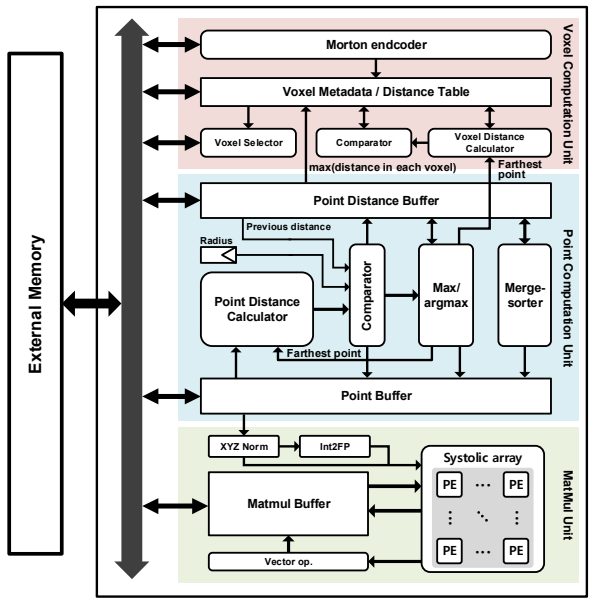

Figure 5: Hardware architecture of the proposed point cloud accelerator.

Voxel Computation Unit. Voxel computation unit oversees voxel-level operations. At the beginning of the inference, the Morton encoder transforms the coordinate information of the point cloud into Morton code and writes it back to the external memory based on the proposed data structure, simultaneously updating voxel metadata. During sampling and grouping, the voxel distance calculator computes the voxel-level distances from the newly selected sample, and comparator array determines voxel-level distance to be updated. The voxel selector identifies a voxel with the farthest distance and its neighbors, issuing instructions to handle these voxels. During the reduced interpolation, which involves neighbor search based on the Morton order of voxels, only the selection of neighboring voxels takes place.

Point Computation Unit. Point computation unit is dedicated to handling point-level operations. Each point distance calculator computes distance between two points in the integer domain. The subsequent comparators can simultaneously compare these distances with the squared radius value and the previous distances. The minimum distances are then forwarded to the Max/argmax unit, which identifies the maximum distance for voxel-distance updating and selects the farthest point to be sampled. Simultaneously, the indices of points with distances smaller than the radius are returned to the point buffer for feature aggregation. In cases where k-nearest neighbor search is necessary instead of ball query, a merger-sorter is implemented.

MatMul Unit. Matmul unit includes a typical systolic array, vector operation module, XYZ norm module, and Int2FP module. XYZ norm module normalizes the coordinate information of the neighbor group through integer subtraction. As the normalized data is in the integer format and the systolic array is designed for the floating-point format, conversion to the floating-point format occurs at the Int2FP modules. The simple architecture of integer operators and Int2FP modules allows for the entire area of both modules to be less than 1% of the systolic array.

Table 1: Evaluated Neural Network Model

Model	Task	Dataset	# Input point
PointNet++[7]	Classification	MdoelNet40 [9]	1K
	Segmentation	S3DIS [1]	4K
		SemanticKITTI [2]	16K

Table 2: Network Accuracy

Dataset	Baseline	Fused SnG	+ Reduced ITP
ModelNet40	92.4	92.2	–
S3DIS	54.3	54.3	54.1
SemanticKITTI	58.1	57.8	57.7

5 EXPERIMENTAL RESULTS

5.1 Experimental Setup

To evaluate the validity of our method, we used PointNet++ [7], a representative point-based model, across three different datasets, as shown in Table 1. To analyze algorithmic impact of our proposed scheme, we initially trained the original PointNet++ using a PyTorch implementation. Subsequently, we replaced the sampling and grouping layers with our proposed scheme without retraining. The maximum upper bits for voxel division were set to 9 bits.

We compared the hardware efficiency of the proposed design with a GPU, NVIDIA RTX-3090, and a point-cloud accelerator, PointAcc. Time and energy consumption of RTX-3090 were evaluated using NVIDIA-smi and Nsight while running the inference of benchmark models. For custom hardware, both our proposed design and PointAcc, we developed cycle simulators and implemented them in Verilog HDL. Both designs were synthesized at 1 GHz in a 28nm CMOS technology using a Synopsys Design Compiler. We used the same design parameters for both accelerators; a 64×64 systolic array, HBM2 for external memory, and 776 KB SRAM.

5.2 Network Accuracy

To validate that the reduced search space can provide enough local information for DNNs, we evaluated the model by reducing the search window with the proposed method (Table 2). The results show that our scheme exhibits good coverage across both the entire input space for sampling and local regions for grouping with an accuracy loss of less than 0.3%. Furthermore, the same search window provides a sufficient number of centroids for interpolation with minimal degradation of mIoU, less than 0.2%. Note that these results are achieved without retraining, unlike the prior works that experience accuracy degradation even after retraining.

5.3 Reduction in Operation

To analyze the reduction in computational load, we compared the number of distance calculations between the original implementation and the proposed method. Fig.6 shows the ratio of the number of distance calculation during the inference of both classification (ModelNet40) and segmentation (S3DIS). We utilized upper 6 bits for ModelNet40, which involves small number of points.

Thanks to the reduced search window, the number of distance calculations required for sampling decreases by 1.7–4.3×. The reuse of distances calculated for sampling eliminates computations required for the ball query, resulting in a further reduction up to 8.3×

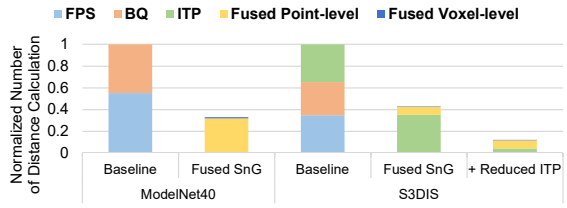

Figure 6: Normalized number of operation.

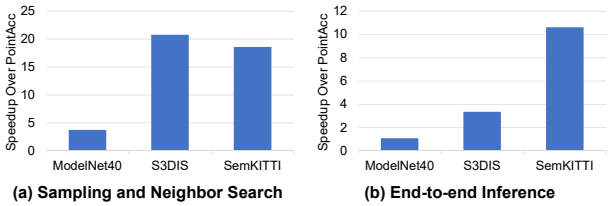

Figure 7: Speedup of the proposed design compared to the point cloud accelerator, PointAcc.

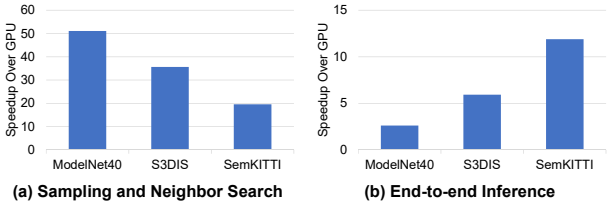

Figure 8: Speedup of the proposed design compared to RTX-3090 GPU.

in computational burden of the entire set abstraction. Furthermore, leveraging spatial locality for the interpolation skips operation for distant centroids, requiring 8.9× less computation. In total, the computation burden for segmentation decreases up to 8.3×. It is noteworthy that reduction trend of data access is similar to that of computation.

5.4 Performance Evaluation

Compared to the point cloud accelerator, PointAcc, the proposed design completes sampling and neighbor search 3.7–20.7× faster, thanks to the reduced number of operations (Fig.7a). Since the small upper bits are set for ModelNet40 due to size of the dataset, the gain is relatively modest compared to the other datasets with a large number of point. In the case of end-to-end inference, the performance gain becomes more pronounced as the number of input point increases, resulting in a speedup of up to 10.6× (Fig.7b). This is because PointAcc consumes significant time on sampling and grouping to sort the entire points, while the reduced search space of our method has an effect of reducing the input size for sorting operations.

Fig.8 shows the speedup of the proposed approach over a Nvidia RTX-3090 GPU. As GPU can handle neighbor search operations, such as ball query and interpolation, in parallel, the performance gain decreases as the number of input point increases, ranging from 51.1× to 10.6×. In contrast, the trend is completely different for end-to-end inference, with a decreased gain of 2.6–11.9×. This is because our architecture employs typical 4K systolic array, which has limited computing ability. While the ratio of time consumed by

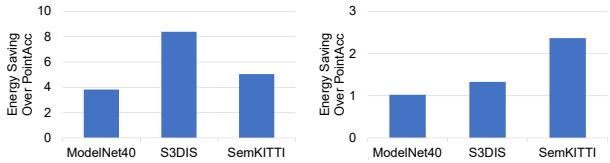

Figure 9: Energy savings of the proposed design over PointAcc.

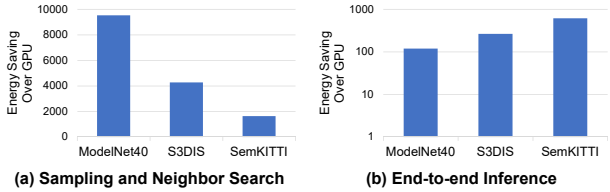

Figure 10: Energy savings of the proposed design over RTX-3090 GPU.

sampling and grouping follows the number of points in the GPU, the time for matrix multiplication becomes dominant in our design.

5.5 Energy Savings

The reduced number of operations and the implementation of integer units in the proposed design leads to 3.8–8.4 × less energy consumption for sampling and grouping operations compared to PointAcc (Fig.9a). As sampling and grouping operation consumes more energy for datasets with a large number of points, the energy savings for end-to-end inference scale up with the number of points, with a gain of 1.0–2.4× (Fig.9b). It is noteworthy that the gain could be further enhanced when handling an even larger number of points. In this experiment, we set a buffer size for point data large enough to hold the largest benchmark. However, in the case that the on-chip buffer cannot store the entire point data, conventional designs need to access the external memory to get the entire data every iteration. In contrast, our method requires particular point data in the selected voxels, not all points. Therefore, our scheme can achieve greater energy saving when handling larger datasets.

Fig.10 shows the energy savings of the proposed design over GPU. The energy consumption for sampling and grouping operation decreases by 1639–9533×, showing a similar trend to the performance improvement. While the inefficiency of the computing unit reduces the gain, the proposed design still achieves a more energy-efficient end-to-end inference, ranging from 120× to 616×.

6 CONCLUSION

Point-based DNNs are widely adopted for the analysis of 3D information from point cloud due to their outstanding performance. However, they have a challenge of significant time consumption in sampling and grouping layers. To mitigate this overhead, we introduced a data structure based on Morton code to exploit spatial locality. With this data structure, we proposed a fused sampling and grouping with search space reduction to reduce the number of computations and maximize data reuse. The dedicated hardware was designed to leverage the proposed approach. The evaluation results show that the proposed method effectively reduces the number of distance calculation up to 8.3× with minimal accuracy loss, less than 0.4%, without retraining. As a result, our scheme achieves efficient sampling and grouping in both time and energy efficiency compared to the previous point cloud accelerator, up to 20.7× and 8.4×, respectively.

ACKNOWLEDGMENTS

This work was supported in part by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2023-00208606, NeuroHub+: Scheduler and Simulator for General In-Memory Neural Network Accelerators, No. 2022-0-00266, Development of Ultra-Low Power Low-Bit Precision Mixed-mode SRAM PIM, IITP-2023-RS-2023-00256081: artificial intelligence semiconductor support program to nurture the best talents, No. 2021-0-01343: Artificial Intelligence Graduate School Program (Seoul National University)), and ISRC at Seoul National University. The EDA tool was supported by the IC Design Education Center(IDEC). (Corresponding Author: Jae-Joon Kim).

REFERENCES

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 2016. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1534–1543.

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. 2019. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9297–9307.

[3] Yu Feng, Gunnar Hammonds, Yiming Gan, and Yuhao Zhu. 2022. Crescent: taming memory irregularities for accelerating deep point cloud analytics. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*. 962–977.

[4] Yu Feng, Boyuan Tian, Tiancheng Xu, Paul Whatmough, and Yuhao Zhu. 2020. Mesorasi: Architecture support for point cloud analytics via delayed-aggregation. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1037–1050.

[5] Yujun Lin, Zhekai Zhang, Haotian Tang, Hanrui Wang, and Song Han. 2021. Pointacc: Efficient point cloud accelerator. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 449–461.

[6] Daniel Maturana and Sebastian Scherer. 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 922–928.

[7] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* 30 (2017).

[8] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. 2022. PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies. *arXiv preprint arXiv:2206.04670* (2022).

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

[10] Xinhao Yang, Tianyu Fu, Guohao Dai, Shulin Zeng, Kai Zhong, Ke Hong, and Yu Wang. 2023. An Efficient Accelerator for Point-based and Voxel-based Point Cloud Neural Networks. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[11] Ziyu Ying, Sandeepa Bhuyan, Yan Kang, Yingtian Zhang, Mahmut T Kandemir, and Chita R Das. 2023. EdgePC: Efficient Deep Learning Analytics for Point Clouds on Edge Devices. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 1–14.

[12] Ziyu Ying, Shulin Zhao, Sandeepa Bhuyan, Cyan Subhra Mishra, Mahmut T Kandemir, and Chita R Das. 2022. Pushing Point Cloud Compression to the Edge. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 282–299.

[13] Jie-Fang Zhang and Zhengya Zhang. 2021. Point-x: A spatial-locality-aware architecture for energy-efficient graph-based point-cloud deep learning. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 1078–1090.

[14] Jiapei Zheng, Hao Jiang, Xinkai Nie, Zhangcheng Huang, Chixiao Chen, and Qi Liu. 2023. TiPU: A Spatial-Locality-Aware Near-Memory Tile Processing Unit for 3D Point Cloud Neural Network. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.