Attention-Based Point Cloud Sampling

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

M.Sc.

Chengzhi Wu

aus Jiangsu, China

Tag der mündlichen Prüfung:	22.10.2024
Erster Gutachter:	Prof. Dr.-Ing. habil. Jürgen Beyerer
Zweiter Gutachter:	Jun.-Prof. Dr.-Ing. Alina Roitberg

Abstract

Point cloud processing has emerged as a pivotal area of research in 3D computer vision, driven by the increasing demand for accurate and efficient representation of three-dimensional data in various applications, including autonomous driving, robotics, and virtual reality. Despite significant advancements, the task of effectively sampling and processing point clouds to maintain crucial geometric features while optimizing downstream task performance remains a substantial yet underexplored challenge. Traditional point cloud sampling methods have been widely used to address such challenges due to their simplicity and efficiency. However, they often fall short in adapting to varying tasks and maintaining a balance between preserving critical details and ensuring uniformity. Recent progress in learning-based sampling methods has shown significant promise in efficiently reducing point cloud size while achieving good performance on 3D vision tasks. Nevertheless, these generative-based methods primarily focus on creating new point clouds that approximate the original distribution, making it challenging to trace sampled points back to their original locations and discern learned patterns. Motivated by the strengths and limitations of both traditional and learning-based sampling methods, this dissertation proposes a series of innovative point cloud sampling methods—APES, SAMPS, and SAMBLE—that integrate task-oriented learning with mathematical statistics-based direct point selection. Each method builds upon the insights and innovations of its predecessors, progressively advancing the state of point cloud sampling techniques.

The first method introduced in this dissertation is the Attention-based Point cloud Edge Sampling (APES). Inspired by the Canny edge detection algorithm used in image processing, APES adapts this concept to the 3D domain by utilizing attention-based mechanisms to identify and sample edge points in

point clouds. The method pioneers the integration of task-oriented learning with mathematically traceable direct point selection. However, APES also highlights the challenge of maintaining shape uniformity while focusing on edge preservation, necessitating more sophisticated sampling strategies. Building on the foundation laid by APES, the Sparse Attention Map-based Point cloud Sampling (SAMPS) method addresses its predecessor's limitations by introducing a sparse attention map that combines local and global information. This method achieves a more effective trade-off between sampling edge points and maintaining the global uniformity of the point cloud, showing significant improvements especially when only a limited number of points are sampled. Despite these advancements, SAMPS reveals the need for adaptive strategies tailored to the unique characteristics of different point clouds. The final method, Sparse Attention Map and Bin-based Learning (SAMBLE) method, represents a significant advancement by learning shape-specific sampling strategies. Building on the sparse attention map proposed in SAMPS, SAMBLE computes point-wise sampling scores and partitions points into bins to enhance discrimination among different point categories. By learning bin boundaries adaptively and determining bin sampling ratios with additional tokens during attention computation, SAMBLE tailors sampling strategies for each shape, leading to enhanced performance across various downstream tasks.

Both quantitative and qualitative results from extensive experiments demonstrate that these methods contribute to developing more effective and adaptable point cloud sampling strategies, enhancing the overall quality of the sampled point cloud data and achieving better performance on various point cloud downstream tasks. The findings and methodologies presented in this dissertation lay the groundwork for future advancements in point cloud processing, paving the way for further enhanced methodologies and more efficient processing techniques that can potentially revolutionize how point cloud data is handled and utilized in diverse real-world applications.

Kurzfassung

Die Punktwolkenverarbeitung hat sich als ein zentrales Forschungsgebiet in der 3D-Computer Vision etabliert, angetrieben durch die steigende Nachfrage nach präzisen und effizienten Darstellungen von dreidimensionalen Daten in verschiedenen Anwendungen wie autonomes Fahren, Robotik und virtuelle Realität. Trotz erheblicher Fortschritte bleibt die Aufgabe, Punktwolken effektiv zu sampeln und zu verarbeiten, um wesentliche geometrische Merkmale beizubehalten und gleichzeitig die Leistung nachgelagerter Aufgaben zu optimieren, eine bedeutende, jedoch wenig erforschte Herausforderung. Traditionelle Punktwolken-Sampling-Methoden wurden häufig eingesetzt, um solche Herausforderungen zu bewältigen, da sie einfach und effizient sind. Sie scheitern jedoch oft daran, sich an unterschiedliche Aufgaben anzupassen und ein Gleichgewicht zwischen der Erhaltung kritischer Details und der Sicherstellung von Gleichmäßigkeit zu finden. Jüngste Fortschritte in lernbasierten Sampling-Methoden haben vielversprechende Ergebnisse gezeigt, indem sie die Größe von Punktwolken effizient reduzieren und gleichzeitig gute Leistungen bei 3D-Vision-Aufgaben erzielen. Dennoch konzentrieren sich diese generativen Methoden hauptsächlich darauf, neue Punktwolken zu erstellen, die die ursprüngliche Verteilung approximieren, was es schwierig macht, gesampelte Punkte auf ihre ursprünglichen Positionen zurückzuführen und erlernte Muster zu erkennen. Angespornt durch die Stärken und Schwächen sowohl traditioneller als auch lernbasierter Sampling-Methoden, schlägt diese Dissertation eine Reihe innovativer Punktwolken-Sampling-Methoden vor—APES, SAMPS und SAMBLE—die aufgabenorientiertes Lernen mit mathematisch-statistischer direkter Punktselektion integrieren. Jede Methode baut auf den Erkenntnissen und Innovationen ihrer Vorgänger auf und treibt den Stand der Punktwolken-Sampling-Techniken schrittweise voran.

Die erste in dieser Dissertation vorgestellte Methode ist das Attention-based Point cloud Edge Sampling (APES). Inspiriert vom Canny-Kantendetektion Algorithmus, der in der Bildverarbeitung verwendet wird, adaptiert APES dieses Konzept auf den 3D-Bereich, indem aufmerksamkeitbasierte Mechanismen genutzt werden, um Kantenpunkte in Punktwolken zu identifizieren und zu sampeln. Die Methode ist wegweisend in der Integration von aufgabenorientiertem Lernen mit mathematisch nachvollziehbarer direkter Punktselektion. APES verdeutlicht jedoch auch die Herausforderung, die Formgleichmäßigkeit zu wahren, während der Schwerpunkt auf der Kantenbewahrung liegt, was anspruchsvollere Sampling-Strategien erfordert. Aufbauend auf den Grundlagen von APES adressiert die Sparse Attention Map-based Point cloud Sampling (SAMPS)-Methode die Einschränkungen ihres Vorgängers, indem sie eine spärliche Aufmerksamkeitskarte einführt, die lokale und globale Informationen kombiniert. Diese Methode erzielt einen effektiveren Kompromiss zwischen dem Sampling von Kantenpunkten und der Aufrechterhaltung der globalen Gleichmäßigkeit der Punktwolke und zeigt insbesondere bei einer begrenzten Anzahl von gesampelten Punkten erhebliche Verbesserungen. Trotz dieser Fortschritte zeigt SAMPS die Notwendigkeit adaptiver Strategien auf, die an die einzigartigen Eigenschaften verschiedener Punktwolken angepasst sind. Die letzte Methode, die Sparse Attention Map and Bin-based Learning (SAMBLE)-Methode, stellt einen bedeutenden Fortschritt dar, indem sie formenspezifische Sampling-Strategien erlernt. Aufbauend auf der in SAMPS vorgeschlagenen spärlichen Aufmerksamkeitskarte berechnet SAMBLE punktweise Sampling-Scores und partitioniert Punkte in Bins, um die Unterscheidung zwischen verschiedenen Punktkategorien zu verbessern. Durch das adaptive Lernen von Bin-Grenzen und das Bestimmen von Bin-Sampling-Verhältnissen mit zusätzlichen Tokens während der Aufmerksamkeitsberechnung passt SAMBLE die Sampling-Strategien an jede Form an, was zu verbesserten Leistungen bei verschiedenen nachgelagerten Aufgaben führt.

Sowohl quantitative als auch qualitative Ergebnisse aus umfangreichen Experimenten zeigen, dass diese Methoden zur Entwicklung effektiverer und anpassungsfähigerer Punktwolken-Sampling-Strategien beitragen, die Gesamtqualität der gesampelten Punktwolkendaten verbessern und bessere Leistungen bei verschiedenen Punktwolken-nachgelagerten Aufgaben erzielen. Die in

dieser Dissertation präsentierten Erkenntnisse und Methodologien legen den Grundstein für zukünftige Fortschritte in der Punktwolkenverarbeitung und ebnen den Weg für weiter verbesserte Methodologien und effizientere Verarbeitungstechniken, die das Potenzial haben, die Handhabung und Nutzung von Punktwolkendaten in verschiedenen realen Anwendungen zu revolutionieren.

Acknowledgements

Writing this dissertation has been an incredible journey, and I could not have completed it without the support, guidance, and encouragement of many individuals. I would like to take this opportunity to express my heartfelt gratitude to them.

First and foremost, I would like to thank my supervisor, Prof. Jürgen Beyerer, for giving me the chance to work at the Vision and Fusion Laboratory (IES) of the Karlsruhe Institute of Technology (KIT) and encouraging me to grow as a research scientist. His unwavering support, invaluable guidance, and constructive feedback throughout this research have been instrumental in shaping this dissertation.

I am deeply grateful to Dr. Julius Pfrommer, who initiated my research journey here and talked with me regularly, sharing ideas and engaging in academic discussions. I also extend my thanks to all the people I met from his department at the beginning of my work, especially Jania Stompe, Andreas Ebner, and Constanze Hasterok. Your warm welcome gave me a fresh and confident start. Dr. Julius Pfrommer is now the leader of the Cognitive Industrial Systems (KIS) department at Fraunhofer IOSB. I wish him great success in the future, building the department larger and better.

I would like to thank the entire group of IES at KIT, especially Chia-wei Chen and Zeyun Zhong, for their frequent discussions with me and their help in many aspects. Special thanks to Arno Appenzeller and Tim Zander for organizing the Freitagsrunde and the summer seminars. Josephine Rehak and Jonas Vogl deserve my gratitude for helping me revise my German speech script for my presentation at the summer seminars. I also thank Andreas Specker, Maximilian Becker, and Mickael Cormier for their support with the

GPU server, which enabled me to conduct my experiments. Additionally, I appreciate the assistance from the secretaries Gaby Gross and Ingrid Celustek.

I acknowledge the financial support from the Carl-Zeiss Foundation. Their generous funding allowed me to focus entirely on my studies and research. This thesis is the result of close collaboration under the AgiProbot project with numerous great people from various KIT research groups. I extend my thanks to the professors involved, especially Prof. Gisela Lanza, Prof. Tamim Asfour, and Prof. Michael Heizmann, for supervising the project and sharing their ideas during our milestone meetings. Thanks also to the project manager, Constantin Hofmann, and all the team members, especially Alexander Cebulla, Simon Mangold, Jan-Philipp Kaiser, and Jan-Felix Klein, for working on the project together.

I would like to thank the people from my former group at the Institute for Visualization and Data Analysis (IVD) at KIT, especially Prof. Carsten Dachsbacher, Jun.-Prof. Boris Neubert, and the secretary Diana Kheil. Diana, who is also a great friend, frequently chatted with me, providing a warm and memorable experience with all the people there.

Thank you to all the students I supervised for your trust. Special thanks to Junwei Zheng and Hao Fu for having proactive academic discussions with me even after their theses were finished. I am grateful to all the doctors who provided care during my doctoral studies, especially Dr. Tanshiyue Xiong, Dr. Mishel Mama, and their colleagues, for helping me stay healthy. I also thank all the restaurants I visited, especially those Chinese restaurants, for satisfying my taste buds.

I am grateful to all my friends who supported me emotionally. Special thanks to those who often or occasionally joined me for lunch or dinner gatherings, including Kunyang Liu, Xi Huang, Huining Meng, Xianlin Luo, Jiaxin Pan, Ruo Jia, Jianyi Liang, Liang Hong, and many others. Thank you, Hanqing Wang, for sharing your life with me throughout my PhD journey and for all the joyful moments we spent together in Paris. Thank you, Jiayi Chen, Botao Ma, Jian Li, and Ben He, for the unforgettable road trips we took together and for your generous help when I moved into my new apartment. Thank you,

Yin Wu, for inviting me to your piano concerts every year; it was great fun to enjoy them with friends like Chia-wei Chen, Ding Luo, Qian Xu, Zheng Li, and others. Thank you, Youfeng Kuang and Shukai Zhang, for inviting me to your wedding; I hope you two are happy forever! And thank you, Xin Luo and Kaihang Song, for trusting me to take care of your cats during your leave—they are truly adorable!

Thank you to all the old friends I met before coming to Germany for keeping in touch despite the distance. Special thanks to Jiming Qian for welcoming me on my first day in Germany, to Lingling Gao for sharing your moments of joy with me now and then, to Han Wu and Zhaoxiong Ding for chatting with me frequently during early days of my study here in Germany, and to Shuhao Cai and Yuhui Dong for the trip to Tromsø, where we witnessed the magnificent aurora together.

Finally, my deepest gratitude goes to my family. To my parents, Guicai Tang and Hongjun Wu, for their unconditional love, patience, and encouragement throughout my academic journey. To my beloved cousins, including Yinxia Jiang, Linjun Tang, Jiali Zhang, Hui Jiang, Jin Gu, and You Wu, for supporting me like true brothers and sisters. And to my grandparents and all other relatives, for being proud of me and cheering for me back in China. Thank you for the constant support, understanding, and for believing in me even during the toughest times. Thank you for being by my side all the time.

Lastly, I would like to extend my sincere thanks to all the participants in my study, whose cooperation and contributions were crucial to this research. Thank everyone who has directly or indirectly supported me in this endeavor. Your encouragement and belief in me have been a great source of motivation.

Thank you all! :)

Karlsruhe, July 2025 *Chengzhi Wu*

Contents

Notation

This chapter introduces the notation and symbols which are used in this thesis.

General notation

Scalars	italic Roman and Greek letters	x, N, α
Vectors	bold Roman and Greek lowercase letters	$\mathbf{m}, \boldsymbol{\alpha}$
Matrices	bold Roman and Greek uppercase letters	$\mathbf{M}, \boldsymbol{\Theta}$
Sets	calligraphic Roman uppercase letters	$\mathcal{B}$
Methods	alphanumeric Roman uppercase letters	A

Scalars

a	point-wise sampling score
b	bias
d	number of feature dimension
d_k	feature dimension of attention key vectors
e	Euler's number
k	number of selected neighbor points
m	element in a vector or a matrix
n	number of selected cells in a matrix column
n_b	number of bins

r	sampling ratio
y^{gt}	binary ground truth indicator
y^{pred}	predicted per class probability
B	batch size
C	channel number
K_{up}	number of neighbors used for upsampling
N	number of points in raw point cloud
M	number of points in downsampled point cloud
α	attention weight
β	number of points in each bin
γ	momentum update factor
δ	standard deviation
κ	number of points sampled in each bin
μ	negative slope of Leaky Rectified Linear Units
ν	boundray values for bin partitioning
ρ	point sampling probability
τ	temperature parameter in Boltzmann sampling
ω	bin sampling weight
ϵ	a regularization term of a small positive number

Vectors

$\mathbf{b}$	bias vector
$\mathbf{k}$	key vector in attention mechanisms
$\mathbf{p}$	feature vector of a point
$\mathbf{q}$	query vector in attention mechanisms
$\mathbf{r}$	vector of sampling ratios in each bin

$\mathbf{v}$	value vector in attention mechanisms
$\mathbf{w}$	perceptron weight vector
$\mathbf{y}^{gt}$	vector of binary ground truth indicators
$\mathbf{y}^{pred}$	vector of predicted per class probabilities
$\boldsymbol{\alpha}$	vector attention weights
$\boldsymbol{\beta}$	vector of point number in bins
$\boldsymbol{\kappa}$	vector of sampled point number in bins
$\boldsymbol{\nu}_c$	bin boundaries derived from current batch of data
$\boldsymbol{\nu}_t$	bin boundaries applied for current iteration
$\boldsymbol{\omega}$	vector of sampling weights in bins

Matrices and Tensors

$\mathbf{G}$	edge gradient of an image
$\mathbf{G}_x, \mathbf{G}_y$	edge gradient in horizontal and vertical direction
$\mathbf{I}$	tensor of an image
$\mathbf{K}$	Key matrix in attention mechanisms
$\mathbf{M}^g$	global attention map
$\mathbf{M}^s$	sparse attention map
$\mathbf{M}_{\text{pre}}$	attention map before softmax
$\mathbf{M}_{\text{post}}$	attention map after softmax
$\mathbf{Q}$	Query matrix in attention mechanisms
$\mathbf{S}_x, \mathbf{S}_y$	Sobel operator
$\mathbf{W}$	weight matrix of a neural layer
$\mathbf{V}$	Value matrix in attention mechanisms
$\boldsymbol{\Theta}$	edge direction in an image

Sets

$\mathcal{B}$	set of points in a bin
$\mathcal{S}$	a point cloud set
$\mathcal{S}_i$	set of neighbor points around a point
TP	true positives
FP	false positives
FN	false negatives

Functions

$f_{\text{mean}}(\cdot)$	compute mean from a set of values
$f_{\text{std}}(\cdot)$	compute standard deviation from a set of values
$h(\cdot, \cdot)$	compute correlation map between two items
$h^l(\cdot, \cdot)$	compute local correlation map between center and its neighbors
$h^g(\cdot, \cdot)$	compute global correlation map between all elements
$Q(\cdot)$	linear layers applied on the query input
$K(\cdot)$	linear layers applied on the key input
$V(\cdot)$	linear layers applied on the value input
$\mathcal{L}_{\text{CE}}(\cdot, \cdot)$	cross-entropy loss function

1 Introduction

1.1 Motivation

Point clouds serve as a prevalent data representation across diverse domains such as autonomous driving, augmented reality, and robotics. These collections of data points, typically obtained from 3D scanners or LiDAR sensors, capture the spatial geometry of objects and environments with high precision. Given the often substantial volume of data involved, managing and processing these large datasets efficiently poses significant challenges. Consequently, the task of sampling a representative subset of points emerges as a fundamental and crucial aspect in the field of 3D computer vision. Effective sampling techniques not only reduce computational load and storage requirements but also enhance the performance of downstream tasks such as object classification, key-point detection, and semantic segmentation. By focusing on the most informative points, researchers and practitioners can ensure robust and real-time applications in dynamic and complex environments.

Traditional point cloud sampling methods encompass a diverse range of techniques that have been widely explored and utilized. One of the most common approaches is Random Sampling (RS), where points are selected randomly, ensuring simplicity and computational efficiency, albeit sometimes at the expense of uniformity and detail preservation. Voxel-based grid sampling divides the point cloud into regular grid cells or 3D voxels and selects representative points within each cell, achieving a more even distribution. Poisson disk sampling aims to maintain a minimum distance between sampled points, ensuring a more natural distribution and avoiding clustering. Farthest Point Sampling (FPS) [Eld97, Moe03] iteratively selects points that are farthest from the already chosen points, ensuring good coverage of the point cloud. In

addition to preserving essential geometric and structural information, various other geometry-based methods have been developed, leveraging point normals or surface curvatures [Pau03] for accurate representation. Another exemplary method in this context is Inverse Density Importance Sampling (IDIS) [Gro18], which selectively samples points whose distance sum values with neighbors are smaller.

A key characteristic of traditional point cloud sampling methods is that they directly select points from input clouds, allowing each sampled point to be easily traced back to the original point cloud. This traceability, along with their simplicity and efficiency, has made RS and FPS widely used in many current neural network models that require downsampling operations [Qi17b, Yu18, Li18b, Wu19, Yan20a]. However, despite their effectiveness, these traditional sampling methods can be limited in their ability to adapt to varying tasks and situations. Once a sampling method is decided, the sampling process becomes independent of the downstream task. Furthermore, in the modern context where more complex point cloud features are available—such as high-dimensional representations in the latent space facilitated by deep neural networks—traditional sampling methods may struggle to adapt to these sophisticated features.

In recent years, learning-based point cloud sampling methods have gained significant attention due to their ability to leverage data-driven approaches for more intelligent and adaptive sampling. These methods typically employ neural networks to learn optimal sampling strategies from large datasets in a task-oriented manner, allowing them to capture specific task or dataset-related characteristics that traditional methods might miss. The first pioneers include S-Net [Dov19], SampleNet [Lan20], DA-Net [Lin21], etc. They use simple Multi-Layer Perceptrons (MLPs) to generate new point cloud sets of desired sizes as resampled results, supplemented by different post-processing operations. MOPS-Net [Qia20] learns a sampling transformation matrix first, and then generates the sampled point cloud by multiplying it with the original point cloud. By making modifications to the network part of S-Net, several methods have also been proposed, e.g. PST-NET [Wan21] and LighTN [Wan23].

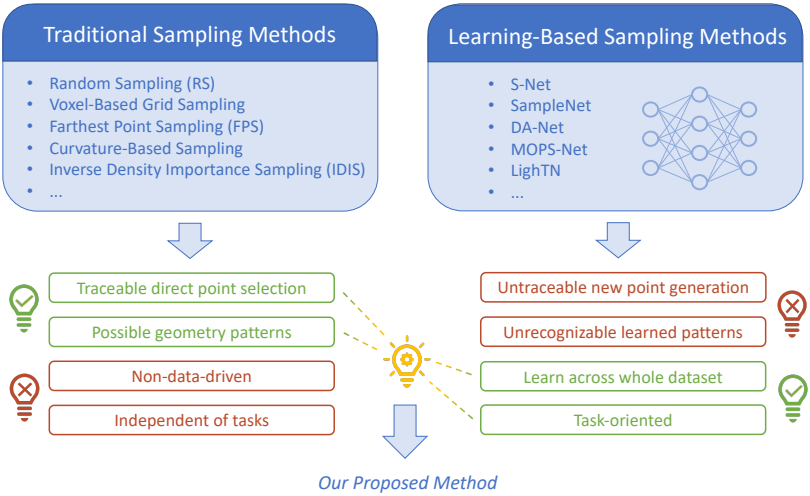


Figure 1.1: Comparison of traditional and learning-based sampling methods, highlighting their pros and cons. Our proposed sampling methods integrate task-oriented learning and mathematical statistics-based direct point selection, aiming to capture the underlying geometry patterns in a learning-based manner.

Compared to traditional sampling methods, Learning-based sampling methods have shown significant promise in effectively reducing the number of points while preserving or even enhancing the performance of various 3D vision tasks. These methods provide a flexible and data-driven approach to sampling, allowing for better adaptation to specific tasks and point cloud characteristics. However, it is important to note that all of these methods are generative-based, with a primary focus on generating new point clouds of reduced sizes that approximate the distribution of the original data. Consequently, they do not directly select points from the input data, making it challenging to trace back a sampled point to its original location within the point cloud. Furthermore, the qualitative results obtained from these methods reveal that their learned underlying patterns or structures are not readily discernible or recognizable.

Motivated by the advantages and disadvantages offered by both types of sampling methods, in this dissertation, we propose a series of point cloud sampling methods that *integrate task-oriented learning and mathematical statistics-based*

direct point selection. In addition, our proposed methods aim to capture and represent the underlying patterns and structures present in the data more effectively. One key to the success of 2D image processing with neural networks is that they can detect primary edges and use them to form shape contours implicitly in the latent space [Yos15]. Inspired by that insight, we first pursue the idea of focusing on salient outline points, i.e. edge points, for the sampling of point cloud subsets for downstream tasks. Broadly speaking, edge detection may be considered a special sampling strategy. Subsequently, given another interesting observation of *when selecting an equal number of neighbors for each point in the input point cloud, points at different positions are chosen as neighbors with varying frequencies*, we propose a more advanced method by considering the frequencies. In this case, multiple point-wise sampling score computation methods are designed and explored, achieving a better trade-off between sampling edge points and preserving global uniformity. Lastly, considering the actual sampling process, both formerly proposed two methods overlook the inherent variations in point distributions across diverse shapes, thereby failing to account for their distinctive characteristics and adopting the same sampling strategy across all point clouds. Therefore, we propose a bin-based method to learn shape-specific sampling strategies for different point cloud shapes, resulting in enhanced performance on point cloud sampling and other downstream tasks.

1.2 Challenges

Point clouds are a crucial data format in 3D computer vision and deep learning, offering detailed spatial information about objects and environments. However, leveraging point clouds as a research target in deep learning introduces a unique set of challenges that span data representation, model architecture, training, and evaluation as the following.

- **Data Scarcity.** There is a relative scarcity of large, publicly available annotated point cloud datasets compared to 2D image datasets. This scarcity limits the ability to leverage supervised learning approaches and hinders the development of high-performing models. Moreover, creating

annotated datasets for point clouds is labor-intensive and time-consuming.

- **Data Preprocessing and Augmentation.** Preprocessing point clouds typically involves normalizing and aligning the data to a common coordinate system, which is crucial for ensuring consistency across different samples. Effective data augmentation techniques are essential for training robust deep learning models. However, creating meaningful augmentations for point clouds, such as rotations, translations, and scaling, while preserving the underlying geometric properties, is non-trivial.
- **Unstructured Nature.** Point clouds are inherently unstructured, consisting of a set of points without any explicit connectivity information. This lack of structure contrasts with the regular grid structure of images, making it difficult to directly apply conventional Convolutional Neural Networks (CNNs) designed for 2D data.
- **Handling Permutation Invariance.** Point clouds are unordered sets, meaning their representation should be invariant to permutations of the points. Designing neural network architectures that respect this permutation invariance is a significant challenge. Traditional CNNs and RNNs are not naturally suited to handle this property.
- **Efficient Feature Extraction.** Extracting meaningful features from point clouds requires specialized model architectures. Techniques such as PointNet and its variants have been developed to address this, but designing efficient and scalable feature extraction mechanisms remains a complex task, especially when the attention mechanism is involved.
- **Standardized Benchmarks and Metrics.** The availability of benchmark datasets and the establishment of standardized evaluation metrics are crucial for ensuring fair and consistent evaluation of various models and approaches. However, defining appropriate metrics that capture the performance nuances in 3D data is complex.

In addition to the aforementioned general challenges in point cloud learning, the development of learning-based sampling methods presents distinct hurdles as a relatively less-explored topic for point clouds. Specifically, there are numerous challenges associated with designing a learning-based sampling method that can effectively preserve specific geometry patterns akin to traditional geometry-based sampling methods. The challenges in proposing such a new point cloud sampling method include the following.

- **Maintaining Geometric Fidelity.** Ensuring that the sampled point cloud accurately represents the geometric features of the original data is critical. This includes preserving details in high-curvature regions and avoiding the loss of important structural information.
- **Balance Between Uniformity and Detail.** Achieving an optimal balance between uniform sampling and capturing fine details is essential. Uniform sampling may overlook critical features, while detail-focused sampling might miss broader structural elements.
- **Shape-Specific Sampling Strategies.** Most existing sampling methods operate independently of the input point cloud shapes. However, incorporating learning-based techniques offers the potential to adapt to diverse input variations, thereby enabling the acquisition of shape-specific sampling strategies.
- **Scalability.** The method should be scalable to handle varying sizes and densities of point clouds without a substantial increase in computational resources. Meanwhile, the method should be able to output any desired sizes of sampled sub-point clouds.
- **Traceable Sampling and Differentiability.** The indexing operation in neural networks naturally disables the gradient backpropagation, which is why most previous learning-based point cloud sampling methods rely on generation-based approaches, resulting in untraceable sampled points. Therefore, the design of a learning-based sampling method that achieves both differentiability for enabling end-to-end training with neural

networks and preserves the traceability of sampled points poses a significant challenge.

- **Task-Oriented Optimization.** Different computer vision tasks (e.g., object classification, key point detection, semantic segmentation) may have different sampling preferences for point cloud sampling. Designing a universal method that optimizes point selection for various downstream tasks while maintaining general applicability is challenging.
- **Evaluation Metrics.** Establishing appropriate metrics to evaluate the effectiveness of the sampling method is crucial. These metrics should comprehensively assess how well the sampled point cloud represents the original data and its suitability for intended applications.
- **Visualization of Learned Features.** Visualizing the features learned by deep learning models from point clouds is not straightforward. Unlike image data, where features can often be visualized as filter activations, point cloud features are harder to interpret. Developing methods to visualize and understand these features is crucial for model validation and improvement.
- **Interpretability and Explainability.** Ensuring that the model's decisions and the importance of selected points are interpretable and explainable can be difficult. Interpretability is crucial for understanding the model's behavior and gaining trust from other researchers who are interested in applying the proposed methods.

1.3 Contributions

The objective of this dissertation is to design point cloud sampling methods that integrate task-oriented learning and mathematical statistics-based direct point selection. Three sampling methods are proposed step by step.

Given the inherent challenge of ensuring meaningful and interpretable learned features within neural network models, as a starting point, we propose to

explore a sampling methodology that incorporates elements from geometry-based techniques. Notably, curvature-based sampling stands out as a widely recognized approach, wherein points located in regions of high curvature are typically deemed significant. These edge points effectively capture the salient outlines of the input point cloud shapes, enhancing their representational fidelity. However, the integration of the process of sampling edge points from 3D point clouds into the training of neural network models presents an intriguing yet challenging endeavor. By revisiting the Canny edge detection algorithm [Can86], which is a widely-recognized classical edge detection method for images, we propose our **A**ttention-based **P**oint cloud **E**dge **S**ampling (termed APES) method for point clouds. It uses the attention mechanism [Vas17] to compute correlation maps and sample edge points whose properties are reflected in these correlation maps. The contributions of APES are summarized as follows:

- A point cloud edge sampling method that combines neural network-based learning and mathematical statistics-based direct point selection. During the training, the backpropagation of the gradient is enabled while the traceability of sampled points is also preserved.
- Two kinds of APES by using two different attention modes. Based on neighbor-to-point (N2P) attention which computes correlation maps between each point and its neighbors, local-based APES is proposed. Based on point-to-point (P2P) attention which computes a correlation map between all points, global-based APES is proposed.
- The intermediate result preserves the point index meaning and is traceable, hence the sampled results can be visualized easily. The qualitative results show that the proposed method successfully achieves the goal of sampling edge points for point cloud shapes.
- The proposed method is scalable to handle varying sizes and densities of point clouds. Meanwhile, the method can downsample the input point cloud into any desired size.

APES introduces an innovative approach to point cloud sampling, with a particular emphasis on the edges of point clouds. However, it excessively focuses rigorously on edge points, resulting in a deficiency in sustaining good global uniformity of the input shapes. Consequently, the interpolation operation becomes impractical during the upsampling process, and the sampling quality of few-point sampling is notably subpar. Therefore, we subsequently propose **S**parse **A**ttention **M**ap-based **P**oint cloud **S**ampling (SAMPS) method to achieve a better trade-off between sampling edge points and preserving global uniformity, overcoming the limitations of APES, and achieving better performance on downstream tasks. The contributions of SAMPS are summarized as follows:

- Sparse Attention Map (SAM) are proposed by merging the information from both global and local attention maps. With different information bases, both carve-based SAM and insert-based SAM are proposed.
- Multiple point-wise sampling score computation methods are also designed and explored, achieving a better trade-off between sampling edge points and preserving global uniformity.
- With the obtained point-wise sampling scores, a simple tunable random sampling policy is proposed with Boltzmann sampling. It controls the sampling process from uniform sampling to Top-M sampling easily with the help of a temperature parameter.

Furthermore, when considering the actual sampling process after point-wise sampling scores are obtained, both APES and SAMPS exhibit a limitation in their treatment of the inherent variations in point distributions observed across diverse shapes. As a result, these methods fail to adequately account for the distinctive characteristics exhibited by different shapes and instead employ the same sampling strategy that is applied uniformly across all point clouds. Hence, we propose a **S**parse **A**ttention **M**ap and **B**in-based **Le**arning method (termed SAMBLE) to learn shape-specific sampling strategies for point cloud shapes. The contributions of SAMBLE are summarized as follows:

- A binning strategy is introduced by using the point-wise sampling scores to partition points for each point cloud, with boundary values updated adaptively during the training phase.
- Additional learnable tokens are introduced during the attention computation step to learn sampling weights for each bin, which determines the number of sampled points in each bin.
- Given the learned shape-specific binning strategy and bin sampling weights, the proposed method allows for more fine-grained control over the sampling process to tailor it to the specific characteristics of each shape, thereby enabling the acquisition of shape-specific sampling strategies.

1.4 Dissertation Outline

This dissertation is organized as follows:

Chapter 2: Related Work This study begins with an extensive survey of the existing literature within the scope of this thesis. The survey includes foundational concepts concerning neural networks and the attention mechanism, as well as recent advancements in point cloud deep learning. Additionally, the survey investigates the various methods proposed for point cloud sampling, encompassing both traditional and learning-based approaches.

Chapter 3: Concept This chapter presents the design concepts underlying the proposed methods. A comprehensive pipeline is introduced to illustrate the learning-based sampling process. The pipeline consists of multiple sub-modules, and different proposed methods address distinct aspects within each sub-module. Furthermore, comprehensive evaluation benchmarks, encompassing shape classification, part segmentation, and few point sampling, are provided to assess the performance and effectiveness of the proposed methods.

Chapter 4: Attention-Based Point Cloud Edge Sampling As a starting point, the first proposed sampling method in this dissertation, named APES, pioneers the integration of task-oriented learning and mathematical traceable direct point selection. It gives a solution that innovatively enables gradient backpropagation during the training while maintaining the traceability of sampled points. Furthermore, a systematic deduction is provided, demonstrating the step-by-step development of this method from the classical Canny edge detection algorithm to the proposed attention-based edge point sampling.

Chapter 5: Trade-off between Edge Sampling and Global Uniformity Preserving In this chapter, we analyze the limitations of APES and propose our more powerful method SAMPS with novel designs to overcome these limitations. Based on the analysis from both qualitative and quantitative aspects, APES concludes that increasing the sampling of edge points enhances the performance of downstream tasks, while on the other hand, solely selecting edge points may have negative consequences. In light of this, SAMPS is introduced as a solution to achieve a better trade-off between sampling edge points and preserving global uniformity.

Chapter 6: Learning Shape-Specific Sampling Strategies with Bin Partitioning Building upon the advancements of APES and SAMPS, in this chapter, we propose an even more advanced sampling method called SAMBLE. After obtaining point-wise sampling scores, both previous methods employ a uniform sampling policy for all shapes, disregarding the inherent variations in point distributions observed across different shapes. SAMBLE aims to address this limitation. It introduces a bin partitioning methodology to learn shape-specific sampling strategies to capture the distinctive characteristics exhibited by individual shapes.

Chapter 7: Concluding Remarks Finally, the outcomes of this work are summarized with directions for future research.

2 Related Work

Exploring learning-based point cloud sampling represents a relatively under-explored domain within point cloud processing. Building upon deep neural network architectures and trained in a task-oriented manner, this approach harnesses the power of attention mechanisms. The overall objective of this thesis is to introduce a pioneering point cloud sampling methodology aimed at learning more effective sampling strategies for downstream tasks. This chapter initiates by elucidating the fundamentals of deep learning models and the attention mechanism. Subsequently, a thorough investigation of existing related work in the realm of deep learning for point cloud analysis and point cloud sampling is conducted.

2.1 Neural Network and Attention Mechanism

Deep learning, a subset of machine learning inspired by the structure and function of the human brain, has emerged as a powerful paradigm in the field of artificial intelligence. At its core, deep learning revolves around the concept of neural networks, intricate systems composed of interconnected layers that process data in a manner akin to how the human brain processes information. What sets deep learning apart is its ability to automatically discover intricate patterns and representations within vast amounts of data, rendering it particularly adept at tasks such as image and point cloud data analysis, natural language processing, and more. By leveraging hierarchical layers of representation, deep learning models can learn to extract features at multiple levels of abstraction, enabling them to tackle complex problems with remarkable accuracy and efficiency. The versatility and scalability of

deep learning have propelled it to the forefront of cutting-edge technological advancements, revolutionizing fields ranging from computer vision and robotics to autonomous vehicles and beyond.

2.1.1 Basics of Deep Neural Networks

Neuron and Multi-Layer Perceptron

The fundamental building block of a neural network is the neuron, a unit designed to simulate the behavior of biological neurons, as shown in Figure 2.1.

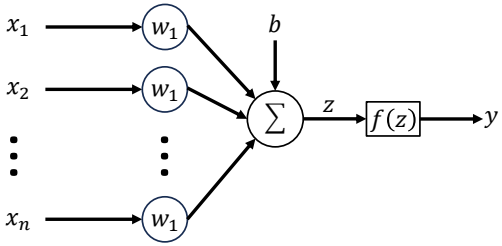


Figure 2.1: Schematic diagram of the structure of a perceptron.

Each neuron receives inputs, computes a weighted sum of these inputs, and then processes the sum through a nonlinear function to generate an output. This process can be Mathematically described as follows:

$$y = f(\mathbf{w} \cdot \mathbf{x} + b)\,, \tag{2.1}$$

where $\mathbf{x}$ represents the input vector, $\mathbf{w}$ denotes the weights associated with the inputs, b is the bias term, f is the activation function, and y is the output of the neuron. These notations are used exclusively in this section.

Common choices for the activation function of deep neural networks are nowadays Rectified Linear Units (ReLU) [Jar09, Nai10]

$$f(x) = \max(x,0) \tag{2.2}$$

and Leaky Rectified Linear Units (LReLU) [Maa13]

$$f(x) = \max(x,0) + \mu \cdot \min(x,0)\,, \tag{2.3}$$

where μ is a factor of a small positive number, such as 0.01. Alternatively, the sigmoid or hyperbolic tangent activation functions are other possible choices.

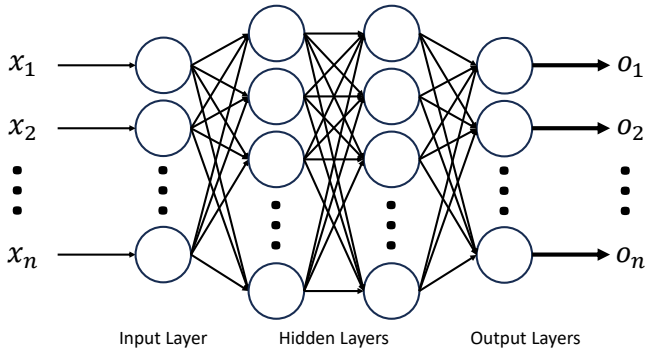


Figure 2.2: Schematic diagram of the structure of an MLP.

As depicted in Figure 2.2, the neurons are further expanded to a Multi-Layer Perceptron (MLP), which is a network composed of individual neurons, organized in layers. MLPs are capable of learning and modeling complex nonlinear relationships through their layered structure. The mathematical formulation of an MLP is shown as follows:

$$\mathrm{MLP}(\mathbf{x}) = f(\mathbf{W}^L f(\mathbf{W}^{L-1} \cdots f(\mathbf{W}^2 f(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2) \cdots) + \mathbf{b}^L)\,, \tag{2.4}$$

where $\mathbf{x}$ represents the input vector, $\mathbf{W}^l$ and $\mathbf{b}^l$ respectively denote the weight matrix and bias vector for the l-th layer from a total number of L layers in the network. These notations are used exclusively in this section. After being transformed by the activation function f, the output of each layer serves as the input to the next layer. This process continues until the final layer is reached. This layered architecture enables the MLP to capture deep patterns and relationships within the data.

Convolutional Neural Network

When applied to more complex tasks, MLPs face a significant challenge due to the vast number of weights required for training. For instance, an MLP with an input RGB image of 256×256 resolution connected to a first hidden layer of 1,024 neurons would need approximately 0.2 billion weights just for the first layer.

In contrast, Convolutional Neural Networks (CNNs) consist of one or more convolutional layers and are primarily employed for image processing, classification, segmentation, and other computer vision tasks. CNNs generally require fewer weights than MLPs due to their convolutional layers, which can be thought of as weight-shared fully connected layers. This architecture not only reduces the number of necessary weights but also enhances the ability to extract features from images.

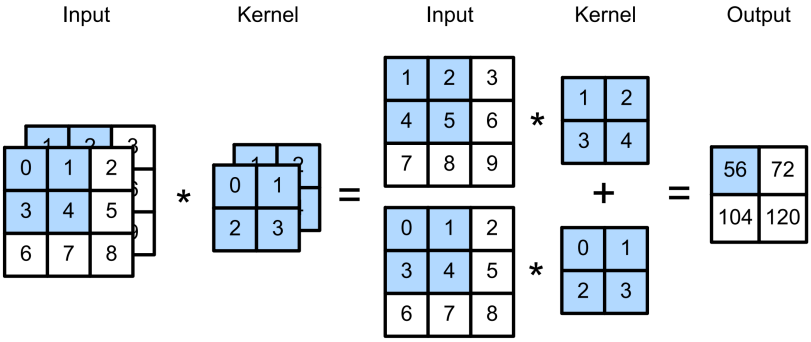


Figure 2.3: Convolution operation with 2 input channels. This is only for one kernel. Repeating this operation C' times for C' kernels and then stacking along the channel dimension results in the final output of a convolutional layer.

The convolution operation in a CNN is pivotal, especially when handling multi-channel input data. The convolution kernel must match the number of input channels to perform effective cross-correlation. Assuming the input has C channels, the convolution kernel will also have C channels, forming a three-dimensional tensor of dimensions $K \times K \times C$. During the convolution operation, a weighted sum calculation across each channel results in a tensor

of dimensions $H' \times W' \times C$. H' and W' are the height and width of the output tensor from a convolutional layer that can be mathematically described as:

$$H' = \lfloor \frac{H + 2P - K}{S} + 1 \rfloor \tag{2.5}$$

$$W' = \lfloor \frac{W + 2P - K}{S} + 1 \rfloor \tag{2.6}$$

where P represents the padding, S is the stride, and H and W are the height and width of the input image data, respectively. These notations are used exclusively in this section. When performing a convolution operation, summing all values along the third dimension (channel dimension) of the tensor results in a two-dimensional tensor of dimensions $H' \times W'$. This operation corresponds to the output for a single convolution kernel. To obtain the final output of the convolutional layer, this convolution operation is repeated C' times, corresponding to the number of output channels. The results are then stacked along the channel dimension, producing a three-dimensional tensor with dimensions $H' \times W' \times C'$.

Figure 2.3 illustrates a simple convolution operation with an input of two feature channels, showing the output of one convolution kernel and the associated input and kernel tensor elements.

2.1.2 Attention Mechanism and Transformer

The Origin of Attention Mechanism

The development of the attention mechanism began with early explorations before 2014, mainly in foundational research in image and speech recognition, where it focused on improving the efficiency of recognizing specific areas or features. In 2014, Bahdanau et al. [Bah14] formally introduced and applied the attention mechanism to enhance neural machine translation. This landmark development led to its rapid adoption across a broad spectrum of sequence modeling tasks, including text summarization, sentiment analysis, and language understanding, thereby sparking extensive research into its improvement and optimization.

Transformer

The vanilla Transformer model, introduced in the seminal 2017 paper "Attention is All You Need" [Vas17], revolutionized machine learning, particularly in the field of Natural Language Processing (NLP). As shown in Figure 2.4, this groundbreaking architecture departed from traditional recurrent and convolutional neural networks, utilizing self-attention mechanisms to enable parallel data processing and more effective sequence dependency handling.

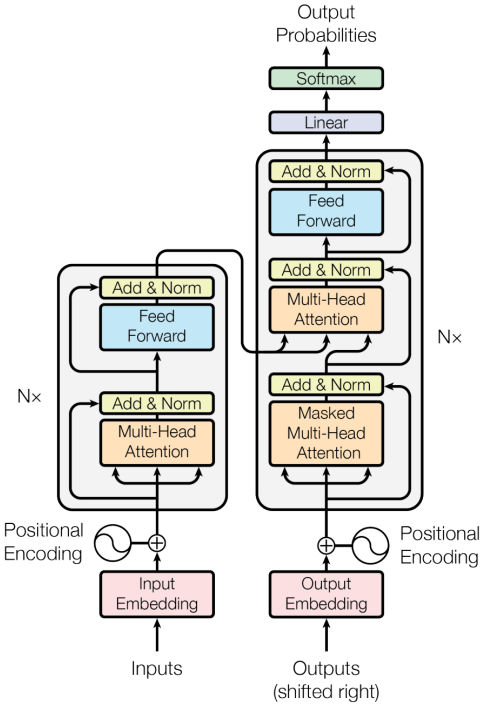


Figure 2.4: Schematic diagram of the structure of a Transformer [Vas17].

As the core of the Transformer, the self-attention mechanism enables the model to capture dependencies between different positions within a sequence by attending to all positions simultaneously. To be more specific, it weighs the

importance of different elements in a sequence by computing attention scores that highlight the relationships between them. The foundational concept of attention is depicted in Figure 2.5.

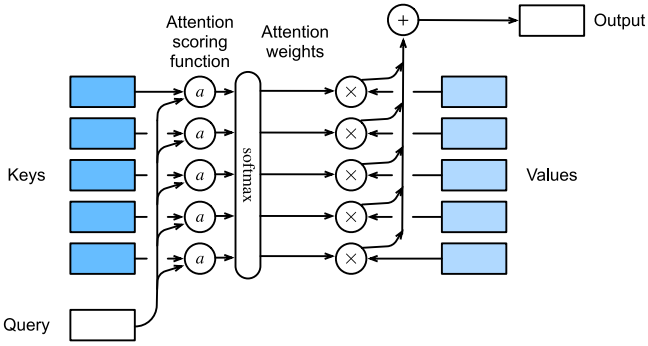


Figure 2.5: Computing the output of attention pooling as a weighted average of values, where weights are computed with the attention scoring function and the softmax operation.

There are different types of attention-scoring functions, such as additive attention, multiplicative attention, and scaled dot product attention [1]. While Bahdanau et al. [Bah14] uses additive attention as the scoring function, Transformer [Vas17] adopts scaled dot product, which has become the predominant scoring function in contemporary applications. Mathematically, the attention mechanism in Transformer can be succinctly represented through the following equation:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\,\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}\,, \tag{2.7}$$

where $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ represent the query, key, and value matrices respectively. Following the computation of the dot product between $\mathbf{Q}$ and $\mathbf{K}$, the result is scaled by the square root of the key dimension $\sqrt{d_k}$ to prevent the dot products from growing excessively large for high d_k values. This scaling adjustment

[1] https://d2l.ai/chapter_attention-mechanisms-and-transformers/attention-scoring-functions.html

ensures that the softmax output remains appropriately calibrated, averting scenarios where the softmax values become extremely small due to overly large dot products. With attention scores being derived through a softmax function, these scores are multiplied by the value matrix $\mathbf{V}$ to yield the final attention output. In this output, each element is a weighted sum of the values, reflecting the importance of different elements in the sequence.

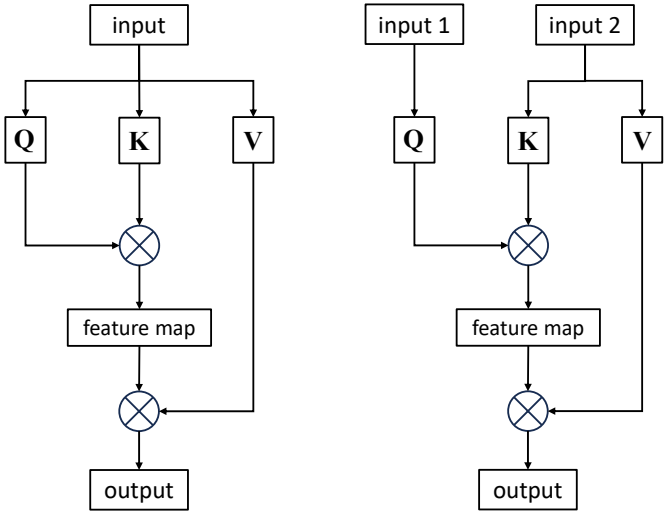
The impact of Transformer is both profound and rapid, markedly enhancing performance across various NLP tasks and inspiring the development of state-of-the-art models such as BERT [Dev19]. In subsequent years, the model has been further refined with variations like Transformer-XL [Dai19], XL-Net [Yan19b], and GPT series [Rad18, Rad19, Bro20, Ope23], each designed to address specific challenges and extend capabilities, especially in managing longer context sequences. Furthermore, as the application of attention mechanisms became more sophisticated, computational efficiency and model interpretability emerged as new focal points of research. This shift led to the development of lightweight attention variants [Sun20a, San19, Sun20b, Yan21, Zha21b] and the exploration of methods to visualize attention weights to elucidate model decisions [Cla19, Vig19, Ser19, Wie19, Kov19]. Overall, the attention mechanism has transitioned from a supportive tool to a fundamental component in deep learning frameworks, particularly in natural language processing and computer vision, significantly advancing the comprehension of complex data relationships.

2.1.3 Advances in Attention Mechanism

Since 2017, researchers have explored more complex and diverse models with attention mechanisms, which not only enhanced model performance but also deepened understanding of the intricate layers of relationships within data. We further introduce several key variants for the common attention mechanism applied in Transformer, including cross-attention, vector-attention, and the integration of additional tokens.

Self-Attention and Cross-Attention

As mentioned above, self-attention is a cornerstone of the Transformer architecture, which enables each element of a sequence to dynamically interact with and weigh the importance of every other element within the same sequence. In this mechanism, features from one sequence form all the queries, keys, and values. Such capability allows it to capture internal dependencies regardless of positional distances between elements. For example, in natural language processing, self-attention allows each word in a sentence to directly interact with every other word, thus facilitating an enhanced understanding of context and relational dynamics within the text.



(a) Self-Attention **(b)** Cross-Attention

Figure 2.6: The characteristic of self-attention is that **Q** and **K**, **V** are from the same source, while cross-attention is that **Q** and **K**, **V** are from different sources.

Similarly, as depicted in Figure 2.6b, cross-attention extends this concept to facilitate interactions between two distinct sequences or sets of elements. It is especially vital in tasks requiring the alignment or comparison of different types of data, such as machine translation or image-captioning, where elements of text must be related to parts of an image. In this mechanism, features from

one sequence form the queries, while elements from another sequence provide the keys and values. This setup enables each element in the query sequence to attend to another sequence, allowing the model to integrate information and context from both sources effectively.

As with self-attention, in cross-attention, the attention scores are also usually calculated using a scaled dot-product mechanism, which determines the level of focus each element in the query sequence should give to elements in the target sequence. This functionality is crucial for tasks where understanding and integrating diverse sets of information is paramount, thereby enabling models to bridge and synthesize different sources of data efficiently.

Scalar-Attention and Vector-Attention

In the Transformer, for each value vector, a scalar weight is generated for the subsequent multiplication and summation operations. That is, the weight for one value vector is applied equally to all the feature dimensions for this value vector. This is also the case in many attention mechanisms, it is termed as scalar-attention since the weight is a scalar per value vector. Alternatively, it is feasible to generate a same-dimensional vector weight for each value vector. As illustrated in Figure 2.7, the Hadamard product (element-wise multiplication) is employed between the learned attention scores and the value matrix to produce the final output.

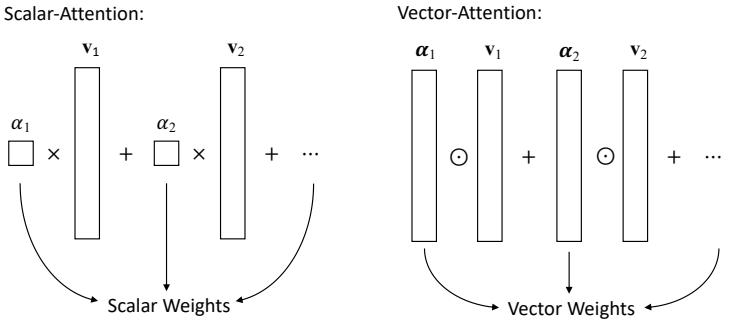


Figure 2.7: While scalar-attention learns a scalar weight for each value vector, vector-attention learns a vector weight for each value vector. In this case, the Hardmard product operations is employed instead of the common multiplication operation.

A common implementation of such vector-attention methods is introduced in PT [Zha21c]. Compared with the scalar-attention mechanism Equation (2.7), the vector-attention mechanism can be described as:

$$\text{VectorAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\mathbf{Q} - \mathbf{K}\right) \odot \mathbf{V}\,, \tag{2.8}$$

where $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ represent the query, key, and value matrices respectively. A softmax function is applied to obtain the vector weights for the value matrix and $\odot$ denotes the Hadamard product (element-wise multiplication). In actual applications, the vector-attention mechanism is mostly used in a cross-attention manner, i.e., $\mathbf{Q}$ and $\mathbf{K}$ are from different sources. Since the subtraction operation is used instead of the dot product operation, a normalization step of dividing the scores by $\sqrt{d_k}$ is unnecessary in this case.

This approach enables the attention mechanism to assess the value vector concerning each of its individual feature dimensions, enabling a more nuanced and multi-dimensional computation of attention. Such a mechanism is more complex and often involves intricate interactions, potentially using nonlinear functions to process the query, key, and value matrices. This complexity allows for a higher degree of flexibility and expressiveness, enabling the model to capture more intricate and diverse data dependencies.

Additional Tokens

In the context of the attention mechanism, a "token"typically refers to the basic unit or element of input data within a sequence. Tokens are individual entities or components that form the input sequence, such as words in natural language processing tasks or pixels in image processing tasks. Each token represents a specific piece of information or a feature within the sequence, and the attention mechanism operates on these tokens to capture relationships and dependencies between them. In recent attention models, especially in advanced architectures like Transformers, various additional tokens are often introduced to enhance the model's capabilities. These additional tokens serve specific purposes and help improve the model's performance.

For example, in BERT [Dev19], the [CLS] (Classification) token and the [SEP] (Separator) token serve crucial roles in the input representation. As illustrated in Figure 2.8, the [CLS] token is typically inserted at the beginning of the input sequence and is used to aggregate information from the entire sequence. In BERT, the output corresponding to the [CLS] token is utilized for various classification tasks, serving as a representation of the entire input sequence. On the other hand, the [SEP] token is employed to separate different segments or sentences within the input data. By using the [SEP] token, BERT can distinguish between distinct parts of the input, enabling the model to process multiple sentences or segments simultaneously. Together, the [CLS] and [SEP] tokens contribute to the effectiveness of BERT in capturing bidirectional contextual information and performing tasks such as text classification, question answering, and named entity recognition with high accuracy and efficiency.

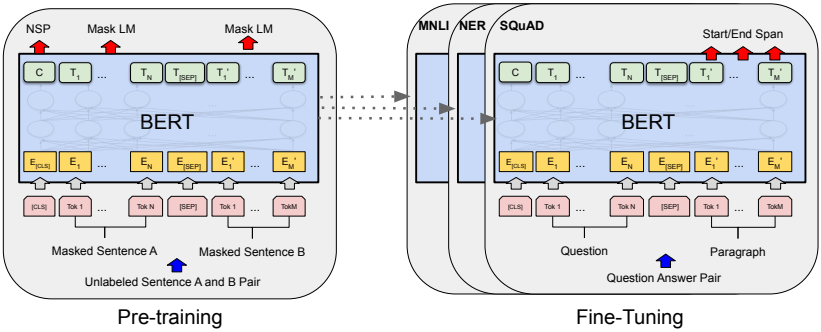


Figure 2.8: Overall pre-training and fine-tuning procedures for BERT [Dev19]. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Extending the Transformer from the natural language processing domain to the computer vision domain, the Vision Transformer (ViT) [Dos21] exemplifies the effective application of the Transformer architecture to image recognition tasks, showcasing its potential for classifying images on a large scale. As shown in Figure 2.9, an additional classification token [CLS] has also been introduced

in ViT as a learnable embedding vector. The [CLS] token is processed alongside the image patch embeddings throughout the Transformer's multiple encoding layers. Leveraging the self-attention mechanism, the [CLS] token accumulates global information from all image patches. At the output of the Transformer encoder, the state of the [CLS] token represents the global features of the entire image. This state is subsequently passed to one or more fully connected layers (the classification head) to generate the final classification prediction.

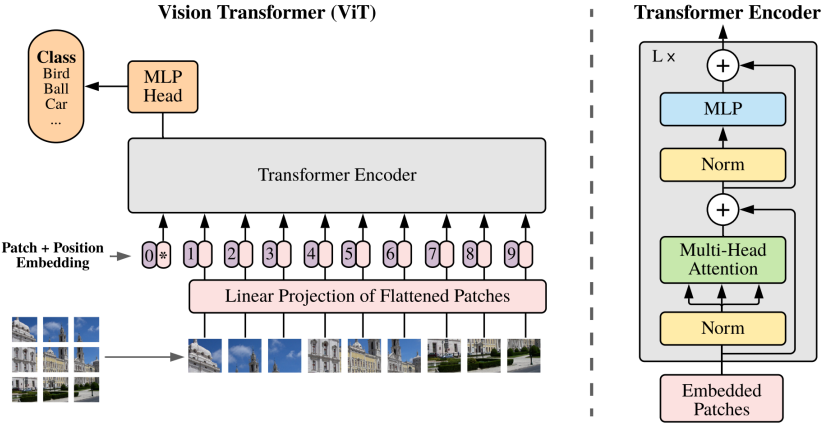


Figure 2.9: Schematic diagram of the structure of ViT [Dos21]. "*" is the additional token for the classification task.

Similarly, ViLT [Kim21], a vision-and-language transformer model, also uses additional learnable [CLS] tokens. These [CLS] tokens are initialized and processed along with the input image patches and text sequences through the transformer layers. It aggregates cross-modal context information and its representation, after being updated across layers, is used for making predictions relevant to the specific task, such as matching text with images, answering questions based on image content, or generating descriptive captions for images.

In the 3D computer vision domain, additional tokens have also been introduced for attention-based point cloud processing. For example, in PointCAT [Yan23], the `[CLS]` token is utilized as a learnable embedding that serves as a representative summary of the entire point cloud. As illustrated in Figure 2.10, at the beginning of the process, the `[CLS]` token is introduced alongside the point cloud data and is fed through the transformer layers. Throughout the transformer layers, the `[CLS]` token interacts with the point cloud data via cross-attention mechanisms, allowing it to aggregate information from all the points in the cloud. This interaction ensures that the `[CLS]` token captures the global context and key features of the point cloud. Finally, after passing through all the transformer layers, the `[CLS]` token is used for downstream tasks such as classification or segmentation, providing a comprehensive representation of the point cloud's overall structure and features.

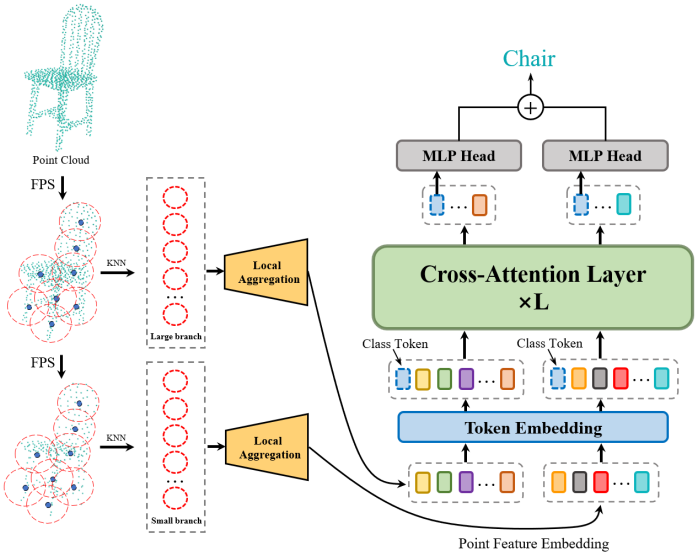


Figure 2.10: Illustration of the PointCAT architecture [Yan23]. An extra learnable class token to each sequence before a stack of cross-attention transformer layers.

2.2 Deep learning for Point Cloud Analysis

In recent years, deep learning methods have achieved remarkable success across various research fields, including computer vision, speech recognition, and natural language processing. The application of these methods to 3D data holds vast implications, especially in areas like autonomous vehicles, robotics, remote sensing, and medical care. 3D data can be represented in several formats, including depth images, point clouds, meshes, and volumetric grids. Of these, the point cloud representation is particularly popular as it preserves the original 3D geometry without any discretization and is widely used across different applications. However, learning from point cloud data presents several challenges as discussed in Section 1.2, particularly including limited dataset sizes, high-dimensional feature spaces, permutation invariance, and the unstructured nature of 3D point clouds. Addressing these issues requires innovative approaches and techniques that can effectively handle the complexity of 3D data.

By overcoming the above challenges, an increasing number of deep learning models have been developed to process and analyze 3D point clouds, leading to significant advancements in the field. In contrast to the early exploration of the voxelization-based methods [Mat15, Jia18, Le18] and the multi-view-based methods [Law17, Bou17, Aud16, Tat18], point-based methods directly operate on the raw point cloud data without converting it into other representations. Therefore, point-based methods are popular approaches nowadays for 3D point cloud understanding. Following the pioneering work of PointNet [Qi17a] and PointNet++ [Qi17b], current methods typically involve two main stages within their designed network models: local feature extraction and global feature aggregation. During the feature extraction stage, local features are computed for each point in the point cloud, capturing the geometric and semantic properties of the point and its immediate surroundings. Commonly utilized features include geometric descriptors like 3D coordinates, normal vectors, and local surface features, as well as learned features from deep neural networks. In the global feature aggregation stage, these extracted features are integrated through a carefully designed method to form a global representation

of the entire point cloud. This global representation is crucial for tasks such as classification, segmentation, and object detection.

While mean-pooling and max-pooling are mostly employed for global feature aggregation, various local feature extraction techniques have been proposed for learning local features. We can categorize these methods into the following groups: pointwise MLP methods, convolution-based methods, graph-based methods, and attention-based methods.

2.2.1 Pointwise MLP Methods

Pointwise MLP Methods employ multiple multi-layer perceptrons to process each point independently and then integrate the point features into a unified feature set using a symmetric function, which is invariant to permutations of the points. PointNet [Qi17a] first employs such a symmetric function to learn the global feature representation of the entire point cloud, enabling it to capture spatial relationships and patterns effectively. Building upon PointNet's success, PointNet++ [Qi17b] further refines point cloud processing by introducing hierarchical neural networks. PointNet++ hierarchically partitions point clouds into a series of local regions, allowing for the extraction of both local and global features. This hierarchical approach enhances the model's ability to capture fine-grained details and intricate structures within complex 3D data, leading to significant advancements in point cloud analysis tasks.

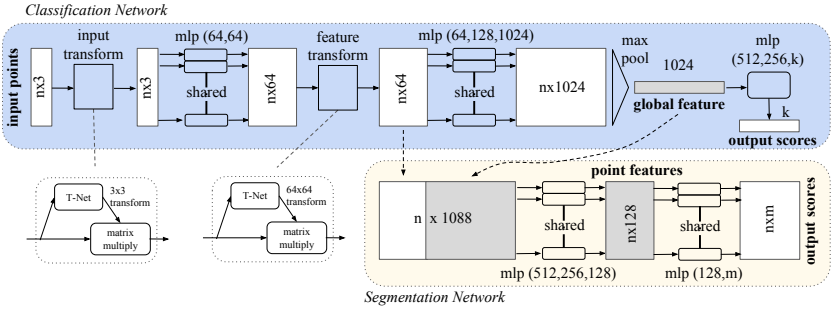


Figure 2.11: Network architecture of PointNet [Qi17a]. It takes points directly as input, applies input and feature transformations, and aggregates point features by max pooling.

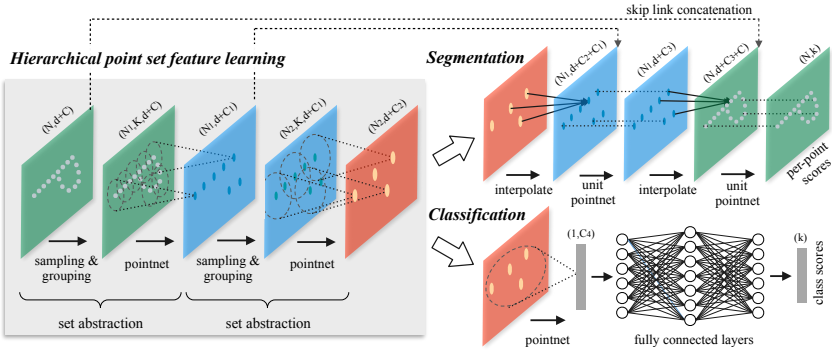


Figure 2.12: Network architecture of PointNet++ [Qi17b]. Local features capturing fine geometric structures are extracted from small neighborhoods. Such local features are further grouped into larger units and processed to produce higher level features.

More recently, MLP-based methods like PointMLP [Ma22], PointNeXt [Qia22], and PointMetaBase [Lin23] have rekindled people's interest. PointMLP [Ma22] rethinks the network design for local geometry in point cloud deep learning and proposes a simple residual MLP framework. By revisiting PointNet++ [Qi17b], PointNeXt [Qia22] is proposed with improved training and scaling strategies. PointNeXt shares the same set abstraction and feature propagation blocks as PointNet++, while adding an additional MLP layer at the beginning and scaling the architecture with the proposed inverted residual MLP blocks. Furthermore, PointMetaBase [Lin23] reformulates existing models for point cloud analysis into a general framework and provides in-depth analysis.

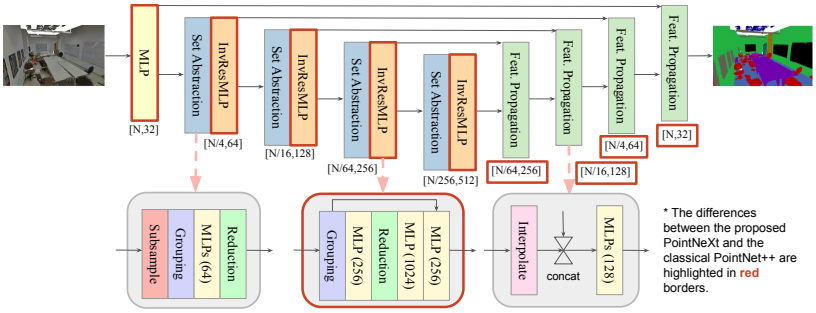


Figure 2.13: The network architecture of PointNeXt [Qia22].

2.2.2 Convolution-based Methods

Compared with kernels defined on 2D grid structures (e.g., images), convolutional kernels for 3D point clouds are hard to design due to the irregularity of point clouds. Various methods have been proposed by adapting the traditional convolution operation to accommodate the irregularly sampled nature of point clouds [Li18b, Mao19, Kom19, Lin20a, Zhu23, Ahn22, Wu19, Tho19, Wu23c]. This adaptation involves defining a kernel, which is a small set of neighboring points, and applying it to each point in the point cloud to generate a feature vector. Repeating this process across all points produces a feature map, which can then be utilized for tasks such as classification or segmentation. Existing 3D convolution methods fall into two categories of continuous convolution and discrete convolution based on the type of convolutional kernels used, as shown in Figure 2.14.

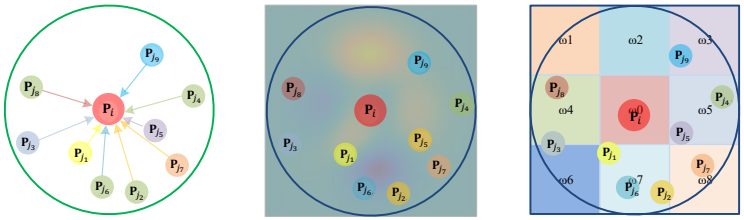


Figure 2.14: An illustration of a continuous and discrete convolution for local neighbors of a point. Figure reproduced from [Guo20].

3D convolution can be interpreted as a weighted sum over a given subset. In 3D Continuous Convolution Methods, the convolutional kernels are defined in continuous space, where the weights for neighboring points are determined by their relative location to the central point. For example, ConvPoint [Bou20] selects kernel elements randomly in a unit sphere and uses an MLP-based continuous function to establish the relation between the locations of the kernel elements and the central point. In DensePoint [Liu19a], convolution is defined as a single-layer perceptron with a nonlinear activator. Features are learned by concatenating features from all previous layers to sufficiently

exploit the contextual information. Both rigid and deformable kernel point convolution operators are proposed in KPConv [Tho19] for 3D point clouds using a set of learnable kernel points. The rigid KPconv operator is illustrated in Figure 2.15. The deformable KPConv operator further integrates local shifts with the results from rigid KPConv operators.

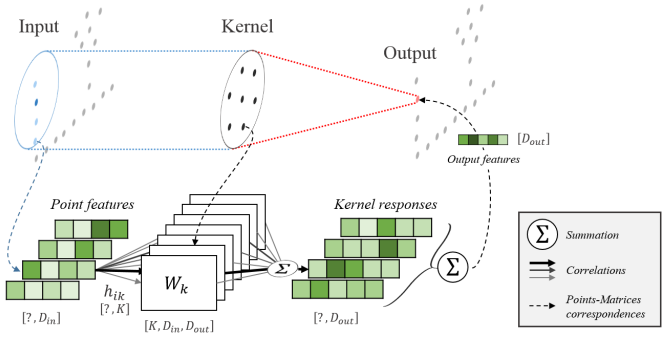


Figure 2.15: An illustration of the KPConv operation. In KPConv, each point feature is multiplied by all the kernel weight matrices, with a correlation coefficient depending on its relative position to kernel points [Tho19].

Some methods also use existing algorithms to perform convolution. In PointConv [Wu19], convolution is modeled as a Monte Carlo estimation of continuous 3D convolution via importance sampling, in which convolutional kernels consist of a learned weighting function and a density function. MCCNN [Her18] also adopts a Monte Carlo estimation approach, utilizing sample density functions implemented with MLPs. By employing Poisson disk sampling to build a point cloud hierarchy, MCCNN enables convolution between varied sampling densities. SpiderCNN [Xu18] introduces SpiderConv, defining convolution as a step function multiplied by a Taylor expansion on the k nearest neighbors. This method captures coarse geometry through local geodesic distances and local geometric variations through interpolation within a cube. Additionally, several methods have been proposed to address the rotation equivariant problem faced by 3D convolution networks [Est18, Coh18, Pou19, Wie22].

Conversely, 3D discrete convolution methods define convolutional kernels on regular grids, where the weights for neighboring points are assigned based on their distances to the center point. Hua et al. [Hua18] transformed non-uniform 3D point clouds to uniform grids, applying convolutional kernels on each grid. Their 3D kernel assigns uniform weights to all points within a grid, computing mean features of neighboring points within the same grid from the previous layer. Lei et al. [Lei19] introduced a spherical convolutional kernel by dividing a 3D spherical neighborhood into volumetric bins, each associated with a learnable weighting matrix. GeoConv [Lan19] models geometric relationships using six bases, weighting edge features along each basis direction with learnable matrices. As illustrated in Figure 2.16, PointCNN [Li18b] reorders input points into a canonical sequence via a $\mathcal{X}$-Conv MLP transformation before applying standard convolution. PAConv [Xu21] presents a more generic convolution procedure for 3D point cloud analysis by dynamically building convolution kernels using self-adaptively learned weight matrices from point positions via the ScoreNet module. This data-driven approach allows PAConv to handle irregular and unordered point cloud data more effectively. Additionally, studies have been conducted to reduce the computational and memory cost of 3D CNNs [Kum19, Rao19].

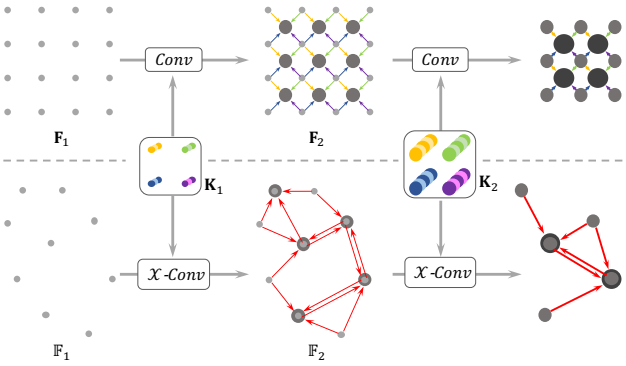


Figure 2.16: Hierarchical convolution on regular grids (upper) and point clouds (lower). PointCNN applies $\mathcal{X}$-Conv recursively to aggregate information from neighborhoods into fewer representative points for point cloud [Li18b].

2.2.3 Graph-based Methods

Graph-based Methods model the point cloud as a graph structure, wherein each point represents a node, and edges are established between neighboring points [Wan19, Sim17, Wu20, Che21a, Zha21a, Xu20, Lin20b, Liu19b]. This graph structure is then leveraged to extract features in spatial or spectral domains [Sim17] and facilitate subsequent analytical tasks. The basic idea of a typical graph-based method is shown in Figure 2.17.

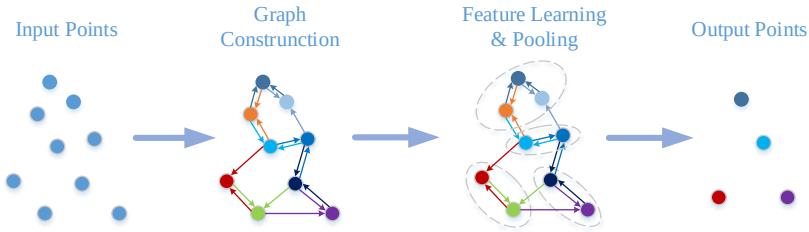


Figure 2.17: A simple illustration of graph-based methods [Guo20].

A prominent example of such a method is the Graph Convolutional Network (GCN) [Kip17], which applies convolutions directly on the graph to extract features. GCNs can be enhanced to learn hierarchical features by stacking multiple graph convolution layers. Simonovsky [Sim17] proposes an ECC (edge conditional convolutional) network that can be applied to any graph structure in combination with the application of edge labels. PointWeb [Zha19], based on PointNet++ [Qi17b], uses adaptive feature adjustment to improve point features in the local neighborhood context, generating a graph in the feature space that is dynamically modified after each layer. DGCNN [Wan19] also generates a graph in the feature space, and an EdgeConv module illustrated in Figure 2.18 is used to capture the local geometric features of point clouds and maintain arrangement invariance. Its excellent performance with the EdgeConv module demonstrates the importance of local geometric features for 3D recognition tasks. Subsequently, by removing the transformation network in DGCNN, LDGCNN [Zha21a] is proposed to simplify the network model, which is optimized by connecting hierarchical features of different dynamic graphs.

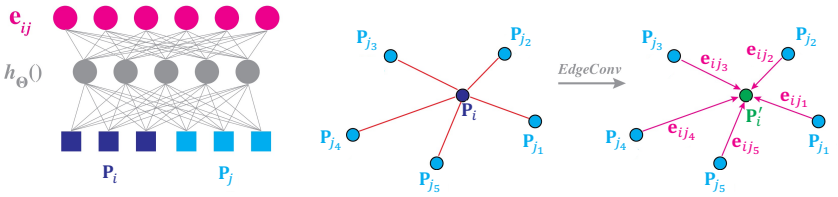


Figure 2.18: Left: Computing an edge feature from a point pair. Right: The EdgeConv operation. The output of EdgeConv is calculated by aggregating the edge features associated with all neighbors. Figure reproduced from [Wan19] .

Graph-based methods applied in the spectral domain implement convolutions as spectral filters, computed by multiplying graph signals with eigenvectors of the graph Laplacian matrix [Bru13, Def16]. RGCNN [Te18] connects each point in the point cloud to all others, updating the graph Laplacian matrix iteratively with a smoothness prior to strengthen feature coherence among neighbors. AGCN [Li18a] learns a distance metric to adapt to diverse graph topologies, normalizing adjacency with Gaussian kernels and learned distances. HGNN [Fen19] introduces hyperedge convolution by applying spectral convolutions on a hypergraph. For local structural insights, LocalSpecGCN by Wang et al. [Wan18] operates on a local graph, avoiding offline Laplacian matrix computation and graph coarsening. PointGCN [Zha18] constructs a graph based on k nearest neighbors in a point cloud, weighting edges with Gaussian kernels and using Chebyshev polynomials for spectral filters.

Additionally, PointManifold [Yan20b] introduces a method of integrating graph neural networks and manifold learning by incorporating point cloud characteristics in a lower-dimensional space and merging them with the original 3D features. Deformable kernel learning is proposed in CIC [Lin20b] for 3D graph convolution networks. It dynamically adjusts a cluster of kernels to match local point cloud structures through two stages: initial kernel sampling and iterative updates based on a loss function measuring deviation from the ground truth label. CurveNet [Muz20] enriches point cloud geometry learning by leveraging innovative curve grouping operators and curve aggregation operators, which are used to create continuous sequences of point segments to facilitate effective feature learning.

2.2.4 Attention-based Methods

A pivotal advancement in recent years within natural language processing and 2D vision is the Transformer [Vas17, Dos21], renowned for its exceptional ability to capture long-range dependencies. This success has significantly influenced point-based models by integrating attention mechanisms. Through attention, Transformers can assess the importance of individual points relative to others, thereby improving feature extraction and discrimination capabilities. The evolution of Transformer-based architectures has notably elevated model performance across various point cloud learning tasks [Guo21, Eng21, Hu20, Pan21, Bha21, Zha22a, Lu22b, Lu22a, Hua23].

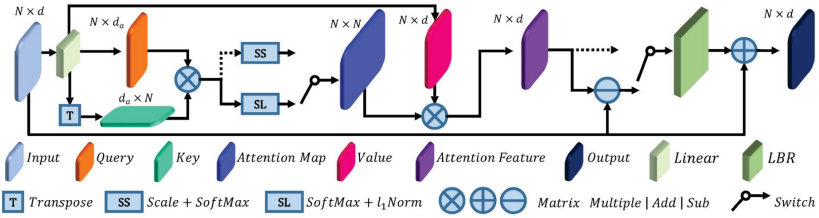


Figure 2.19: Network architecture of PCT [Guo21]. A typical self-attention is employed.

PCT [Guo21] pioneers this direction by replacing the encoder layers in the PointNet framework with self-attention layers, as illustrated in Figure 2.19. It has also been adapted for various real-world point cloud learning tasks [Wu23a, Wu24a]. Based on U-Net [Ron15], PT[1] [Zha21c] is proposed to better aggregate local features by leveraging vector-attention modules. Its subsequent series work of PTv2 [Wu22] and PTv3 [Wu24d] further improve model performance by introducing grouped vector-attention and point cloud serialization. SA-Det3D [Bha21] proposes a deformable self-attention module to enable scaling explicit global contextual modeling to larger point clouds, leading to more discriminative and informative feature descriptors. To empower multi-scale learning, 3DCTN [Lu22b] introduces a multi-scale KNN grouping strategy for better context fusion, while a dual self-attention module is proposed in 3DPCT [Lu22a] to improve feature extraction. Furthermore, Stratified Transformer

[Lai22] partitions the 3D space into cubes and samples additional distant points as the key input to capture long-range contexts, as illustrated in Figure 2.20. A more comprehensive ablation study is performed in [Wu24c] to explore the impact of different module choices for various attention mechanisms applied to 3D point clouds.

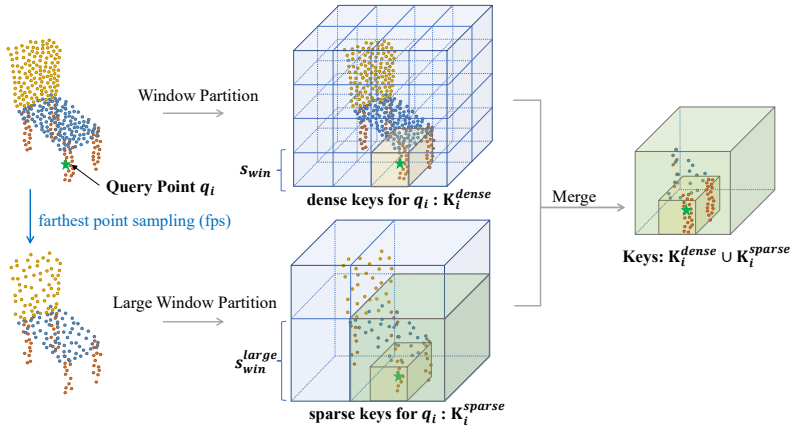


Figure 2.20: Illustration of the stratified strategy for keys sampling [Lai22]. The green star denotes the given query point and multi-scale key points are selected.

Apart from point feature-based attention, multiple models have been proposed to utilize patch or even global features for attention operations. For example, PatchFormer [Zha22a] introduces a lightweight patch-based attention block, which can be seen as a variant of the original self-attention to approximate the global map at a lower computational cost. More interestingly, as illustrated in Figure 2.21, PT^2 [Eng21] introduces SortNet for computing local-global attention to capture spatial point relations and shape information. The proposed SortNet learns certain patches for the input point clouds that capture important shape details. MPAN [Li20] employs a multi-part attention network for point cloud shape retrieval. In addition, approaches that apply Transformers for point cloud self-supervised learning have also been proposed and explored [Yu22, Pan22, Zha22b, Wu24b, Liu22].

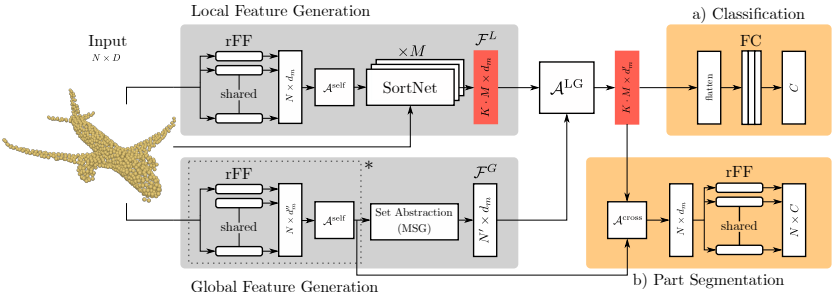


Figure 2.21: Overview of the Point Transformer architecture [Eng21]. SortNet produces an ordered set of local patch features that are attended against the global feature of the input point cloud.

2.3 Point Cloud Sampling

2.3.1 Traditional Sampling Methods

Point cloud sampling is a key process in 3D data handling for simplifying high-resolution dense point clouds. Over the past decades, non-learning-based traditional sampling methods [Eld97, Moe03, Gro18] have predominantly been widely employed due to their simplicity, effectiveness, and robustness.

Uniform sampling involves selecting points from the point cloud at regular intervals, ensuring an even distribution of points across the entire dataset. This method is straightforward to implement and guarantees a uniform density of sampled points, which is useful for maintaining the overall structure of the point cloud. However, it might not capture important features adequately if the point cloud has varying densities. For large point cloud scenes, voxel-based grid sampling [Sun01, Dow14] is more often employed. As illustrated in Figure 2.22, it divides the point cloud into a 3D grid of voxels and replaces all points within each voxel with a single representative point, typically the centroid. This method effectively reduces the number of points while preserving the spatial distribution of the original point cloud. It is efficient and easy to implement, making it a popular choice for preprocessing steps in many point

cloud processing pipelines. However, the resulting sampled point cloud can suffer from a loss of fine details and sharp features.

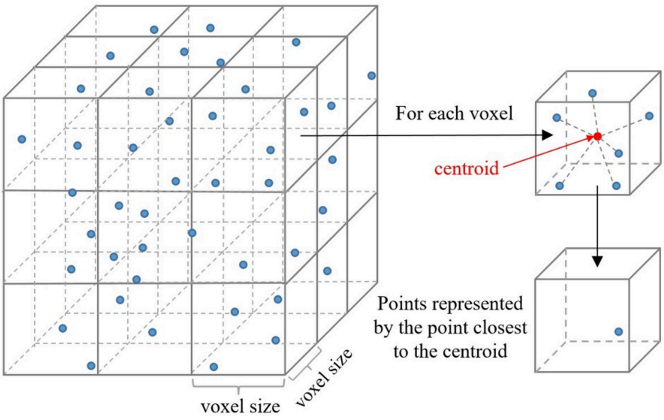


Figure 2.22: Process of voxel-based grid sampling. Image sources from [Wan22].

Random Sampling (RS) has also been frequently adopted [Zho18, Qi20, Gro18]. It selects a subset of data from a larger dataset through random selection mechanisms. The key principle of RS is that each member of the dataset has an equal probability of being chosen, ensuring that the sample is generally unbiased and representative. This technique is also straightforward to implement and is applicable across diverse datasets. Random Sampling is particularly beneficial for large point clouds, as it reduces computational costs while providing a snapshot for analysis. However, its randomness can sometimes lead to less optimal representation of smaller subgroups within the dataset.

For deep learning on point clouds of common sizes, Farthest Point Sampling (FPS) [Eld97] is the most widely used one [Qi17b, Li18b, Wu19, Qia22, Zha21c]. As illustrated in Figure 2.23, the process begins by selecting a random starting point and then iteratively chooses subsequent points that are the farthest away from all previously selected points. This strategy ensures a uniform distribution of samples across the entire dataset, making FPS particularly effective for reducing redundancy while retaining a representative subset of

the original point cloud. Additionally, the algorithm is straightforward to implement and can be adapted for various data sizes and dimensions.

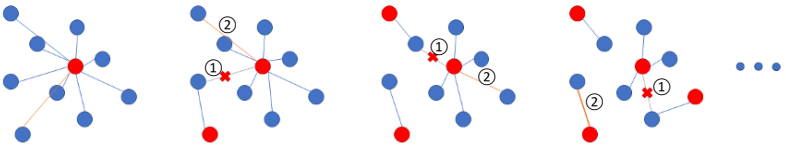


Figure 2.23: Process of Farthest Point Sampling, points that are the farthest away from all previously selected points are iteratively sampled.

Similar to uniform sampling, Poisson disk sampling [McC92, Leh03] aims to create a sample distribution where points are spaced as evenly as possible. By ensuring a minimum distance between any two points, this method is particularly useful for generating visually pleasing and evenly distributed point clouds, making it ideal for rendering and visualization purposes. Additionally, many geometry-based sampling methods have been proposed. For example, curvature-based sampling [Pau02] selects points based on the local curvature of the point cloud. Points in regions with high curvature are sampled more densely than those in flatter regions, ensuring critical geometric features are well represented in the sampled point cloud. Another method of Inverse Density Importance Sampling (IDIS) [Gro18] defines the inverse density importance of a point by simply adding up all distances between the center point and its neighbors, and samples points whose sum values are smaller.

2.3.2 Learning-Based Sampling Methods

Recently, learning-based sampling methods have shown better performances on point cloud sampling when trained in a task-oriented manner. The pioneering work of S-Net [Dov19] generates new point coordinates directly from the global representation. As illustrated in Figure 2.24, S-Net employs a progressive network to create a new point cloud with fewer points and uses a sampling regularization loss to ensure close matches between the generated points and those from the original point cloud.

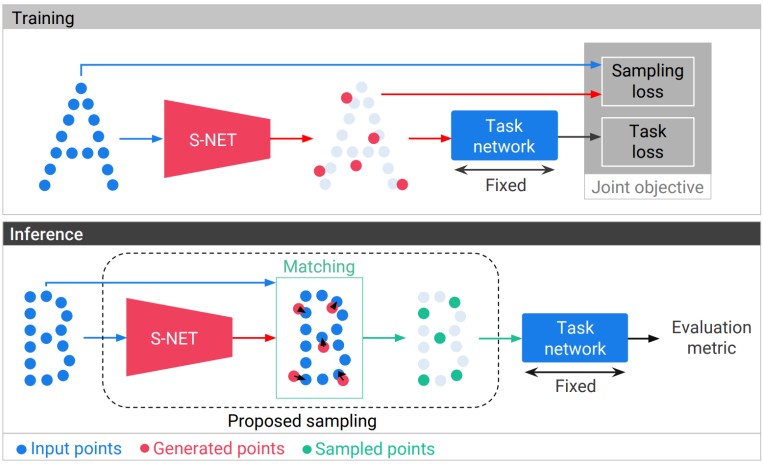


Figure 2.24: An illustration of S-NET [Dov19].

Its subsequent work of SampleNet [Lan20] further introduces a soft projection operation for better point approximation in the post-processing step. Its training process is illustrated in Figure 2.25. Additionally, DA-Net [Lin21] extends S-Net with a density-adaptive sampling strategy, which decreases the influence of noisy points. Replacing the MLP layers in S-Net with several self-attention layers, PST-NET [Wan21] reports better performances on trained tasks. Its subsequent work of LighTN [Wan23] proposes a lightweight Transformer framework for resource-limited cases.

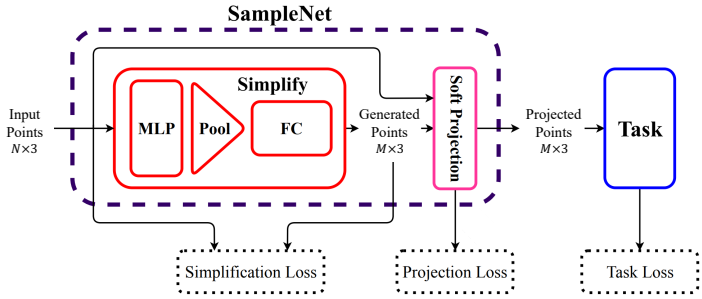


Figure 2.25: Training pipeline of SampleNet. Figure reproduced from [Lan20].

There are also some methods that take a different approach to achieve learning-based point cloud sampling. For example, by investigating the output in the max-pooling layer, CPL [Nez20] proposes an adaptive hierarchical downsampling method for point cloud classification. Instead of generating a new point cloud from the latent features, MOPS-Net [Qia20] employs a learned sampling transformation matrix to directly generate a new sampled point cloud by multiplying it with the original point cloud, as illustrated in Figure 2.26.

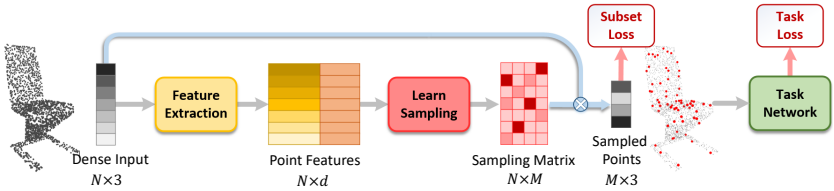


Figure 2.26: The structure of MOPS-Net. It is a matrix optimization-driven learning method for task-oriented 3D point cloud downsampling. Figure reproduced from [Qia20].

On the other hand, there is an increasing body of work designing neural network-based local feature aggregation operators for point clouds. Although some of them (e.g., PointCNN [Li18b], PointASNL [Yan20a], GSS [Yan19a]) decrease the point number while learning latent features, they can hardly be considered as sampling methods in the true sense as no real spatial points exist during the processing.

Overall, current learning-based point cloud sampling methods predominantly adopt a generative approach. They generate a new point cloud with reduced points and establish connections between points in the generated point cloud and their counterparts in the original point cloud, often through tailored loss functions. While these methods accomplish downsampling, the resultant point cloud is not a direct subset of the original. Instead, it forms a distinct entity linked to the original input through enforced relationships, necessitating subsequent processing. As a result, this leads to the non-traceability of the generated new points. Furthermore, they fail to consider specific geometric information from the input point clouds, leading to unidentifiable patterns in the sampling outcomes.

3 Concept

3.1 Overall Sampling Framework

This section presents the design concepts underlying the proposed methods. The overall concept of the proposed sampling framework is depicted in Figure 3.1. To perform the sampling operation on an input point cloud, three steps are executed sequentially: building the attention map, computing point-wise sampling scores, and establishing the sampling policy. This section provides a brief introduction to these steps, while detailed explanations are presented in the later chapters, addressing the reasons behind their proposal, their sources of inspiration, and intricate design details.

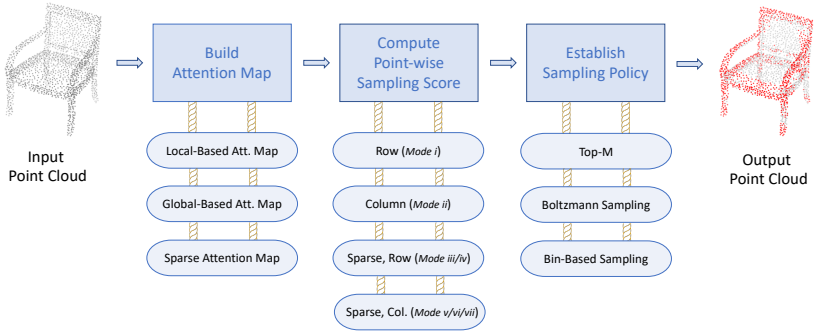


Figure 3.1: Overall concept.

Furthermore, it is important to highlight that the three methods proposed in this dissertation, APES, SAMPS, and SAMBLE make distinct contributions to these three steps. Each method builds upon the previous one, overcoming

certain limitations and achieving better performance. These advancements are summarized in the figures in their respective summary sections, showcasing the specific strengths and improvements introduced by each method.

3.1.1 Build Attention Map

In a transformer model, an attention map refers to the matrix that represents the importance or relevance of each token in a sequence to every other token in the same sequence. It is a key component of the self-attention mechanism used in transformers. It allows the transformer to weigh the importance of different tokens in a sequence when processing the input. This mechanism is based on the concept of attending to different parts of the input sequence to build contextual representations.

To compute the attention map, the transformer model applies three linear transformations to the input sequence: the query matrix **Q**, the key matrix **K**, and the value matrix **V**. These transformations project the input sequence into three different subspaces. The attention map is computed by taking the dot product between the query vectors and the key vectors, followed by a softmax operation to obtain the attention weights. These attention weights determine how much each word or token "attends" to other tokens in the sequence. The attention weights are then used to compute a weighted sum of the value vectors, producing the final representation for each token.

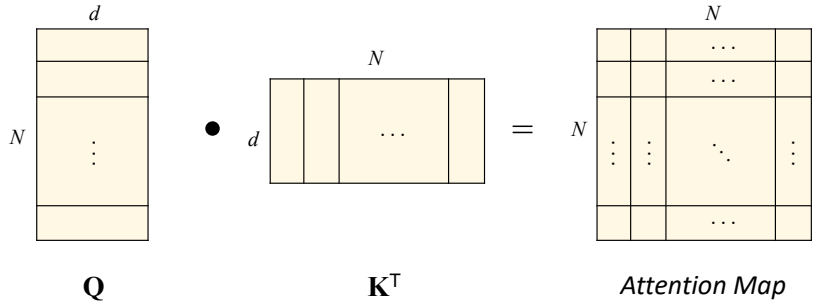


Figure 3.2: Attention map computed from global-based self attention. Both query and key input are the latent representations of all points. N stands for the number of points, and d stands for the dimension of encoded latent representation.

In point cloud data analysis, beyond point-to-point global attention, another useful technique is point-to-neighbor local attention. This approach enables modeling of local dependencies within the point cloud by considering the relationships between a particular point and its neighboring points. In this case, the latent representation of a specific point serves as the query, while the latent representations of its neighbor points act as the keys. By calculating the dot product between the query vector and the key vectors of its neighbors, a local attention map can be generated. This attention map reflects the relevance or significance of each neighbor point to the query point. This process is repeated for each point in the point cloud, resulting in an individual local attention map for every point. By leveraging these attention maps, the model can focus on the local context surrounding each point, capturing fine-grained relationships and enhancing the overall understanding and representation of the point cloud data. This point-to-neighbor local attention mechanism facilitates the extraction of local features and enables the model to effectively process and analyze the intricate structures present in the point cloud.

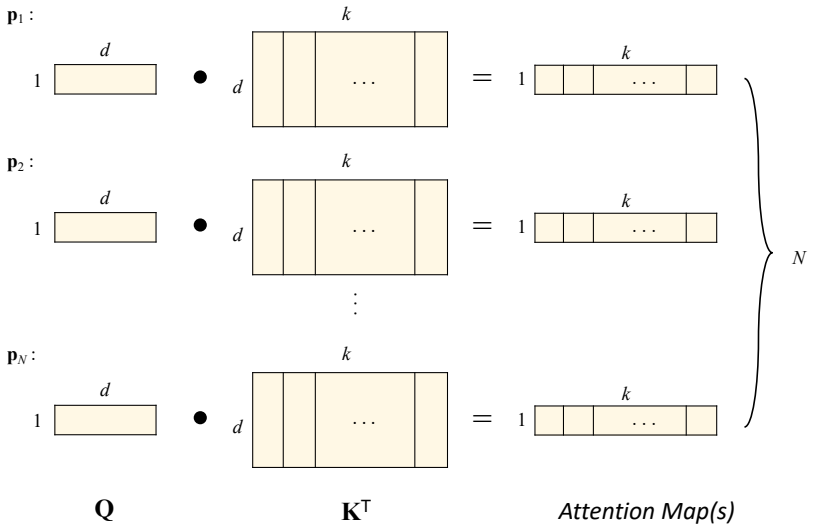


Figure 3.3: Attention map computed from local-based cross attention. The query input is the latent representation of a certain point, while its neighbors' latent representations serve as the key input. k stands for the number of selected neighbors.

By merging the information from both local and global attention, it is possible to generate a Sparse Attention Map (SAM). This involves selecting a certain number of cells for each row in the attention map, where each row represents a specific point. These selected cells indicate the neighbors of the corresponding point, with the values in the non-selected cells set to zero. The sparse attention map can be constructed either from a global basis or a local basis.

On the global basis, the attention matrix is pre-computed as the global attention map. From this matrix, the selected cells corresponding to the neighbors of each point are "carved out", while the remaining cells are set to zero. This approach considers the overall relationships between all points in the point cloud. On the other hand, on the local basis, the local attention maps are pre-computed. These local attention maps capture the point-to-neighbor relationships. The values from the local attention maps are then "inserted into" the corresponding cells of the attention matrix, while the other cells are set to zero. This approach focuses on the specific local context surrounding each point.

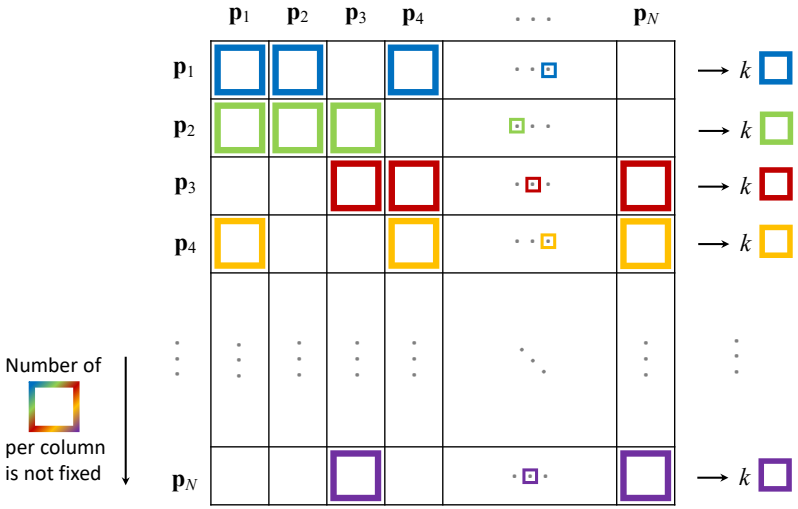


Figure 3.4: Sparse attention map, merging the information from both local and global. The values of non-selected cells are set to zero. k stands for the number of selected neighbors in the local region for each point.

3.1.2 Compute Point-wise Sampling Score

Based on the built sparse attention map, point-wise sampling scores can be computed using various methods. For instance, when using local-based attention maps, one approach is to compute the row-wise standard deviation as the score for each point. This metric quantifies the variability of attention assigned to a point's neighbors, indicating the diversity of information it receives from its local context. On the other hand, when using a global-based attention map, the column-wise sum can be computed as the score for each point. This metric represents the overall attention received by a point from the entire point cloud, capturing its global significance. The specific reasons behind using row-wise standard deviation and column-wise sum as score metrics can be found in Section 4.3.

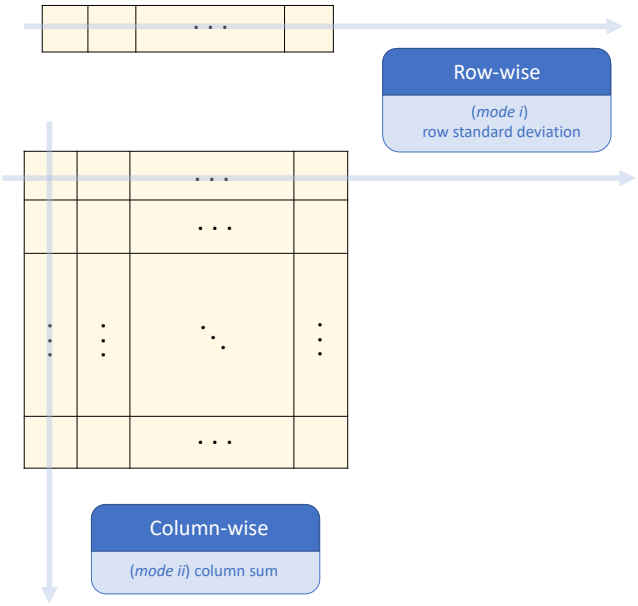


Figure 3.5: With local-based attention map, the row-wise standard deviation can be defined as the score computation metric. With global-based attention map, the column-wise sum can be defined as the score computation metric.

With a sparse attention map, similar metrics can be defined based on the sparsity condition. These metrics may take into account the specific characteristics and properties of the sparse attention map, enabling effective scoring of individual points based on their relevance and connectivity within the point cloud. Additionally, considering that points are chosen as neighbors with varying frequencies, additional score metrics can be devised to account for such variations and provide a comprehensive representation of each point's importance and influence within the point cloud. Detailed explanations of the proposed metrics are given in Section 5.3.

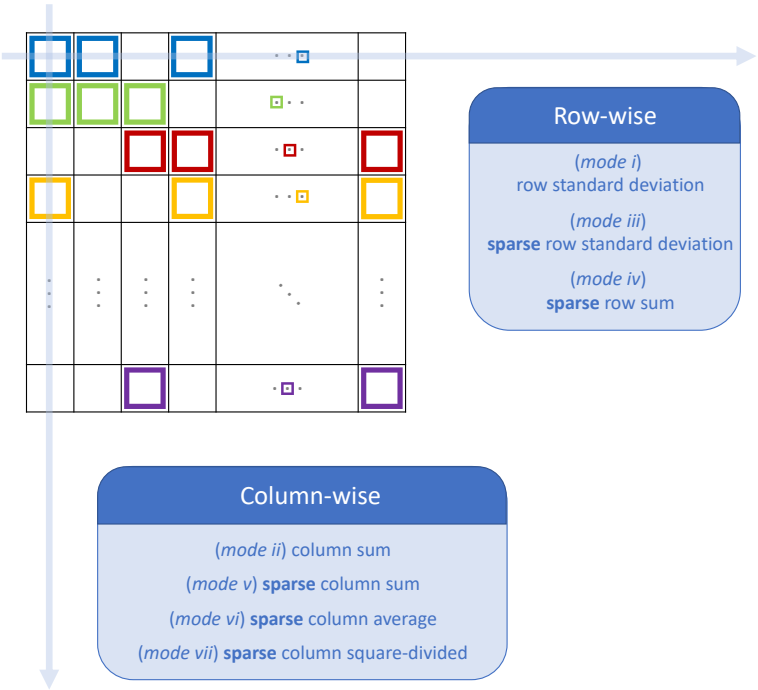


Figure 3.6: With sparse attention, more score computation metrics can be defined.

We call the method of computing point-wise sampling scores from full/sparse attention maps as *Indexing Mode*. Note that not all indexing modes are applicable to all attention maps. The modes that can be used for different attention maps are summarized in Table 3.1.

Table 3.1: The applicable indexing modes for different kinds of attention maps.

Attention Map		Indexing Mode						
		i	*ii*	*iii*	*iv*	*v*	*vi*	*vii*
Local-Based		✓						
Global-Based		✓	✓					
Sparse Attention Map	Insert-Based			✓		✓	✓	✓
	Carve-Based	✓	✓	✓	✓	✓	✓	✓

3.1.3 Sampling Policy

After computing the point-wise sampling scores, the next step in the sampling framework is to establish a sampling policy for the sampling operation. The commonly used sampling policy is the Top-M policy, where M points are sampled from N points by selecting the points with the top M sampling scores among all N points. This policy, known for its simplicity and effectiveness, aims to prioritize points with higher sampling scores. However, it is important to note that this approach may not always yield the optimal sampling result for downstream tasks. This is due to the fact that the data distribution of the sampled output can differ significantly from the data distribution of the input, potentially omitting important latent features and causing information loss.

To address this limitation and introduce more mathematical statistics into the sampling process, it is possible to transform the point-wise sampling scores into point sampling probabilities. By assigning probabilities to each point based on their sampling scores, the sampling process can be modified to involve random selection based on these probabilities. This transition can be achieved using the well-known Boltzmann distribution, which provides a probabilistic framework for sampling. In this dissertation, we refer to this

policy as Boltzmann sampling. By incorporating Boltzmann sampling, we aim to introduce a more probabilistic and exploration-oriented element into the sampling process.

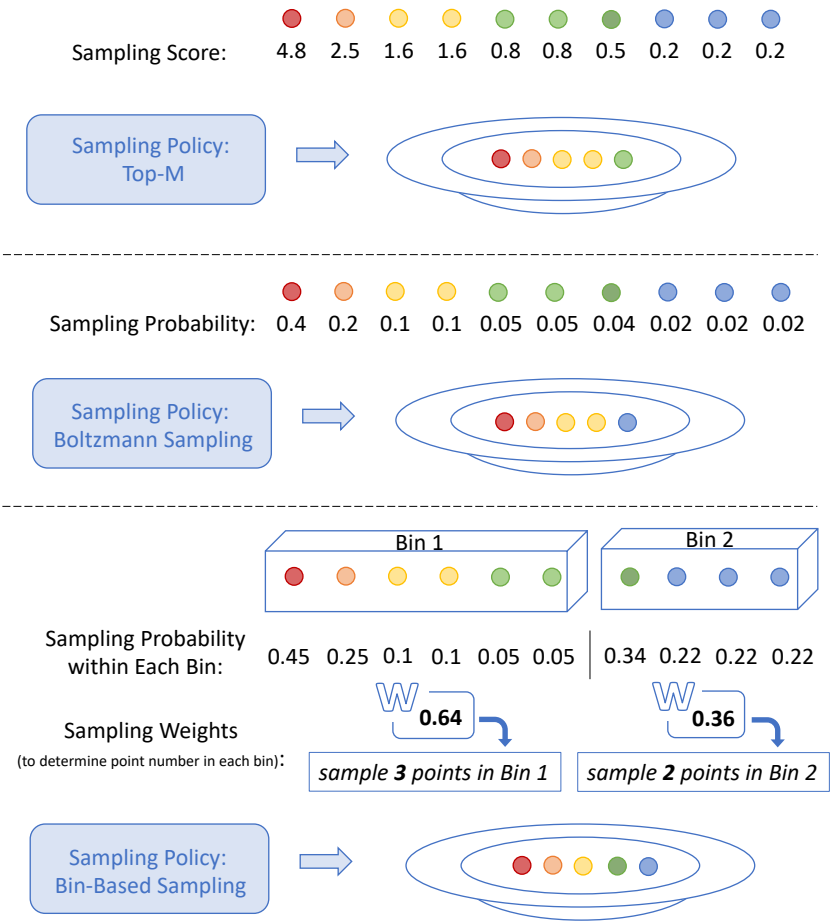


Figure 3.7: An illustration of three sampling policies. Note for bin-based sampling, either top-M sampling or Boltzmann sampling may be used within each bin.

Despite the advantages of Top-M sampling and Boltzmann sampling, both approaches overlook the inherent variations in point distributions observed across diverse shapes. By employing the same sampling strategy uniformly across all point clouds, these methods fail to adequately account for the distinctive characteristics exhibited by different shapes. Consequently, important shape-specific details and variations may be overlooked during the sampling process.

To address this limitation and enable shape-specific sampling strategies, we propose a more advanced sampling policy called bin-based sampling. This policy introduces a binning strategy that partitions points for each point cloud based on their point-wise sampling scores. The boundary values of the bins are updated adaptively during the training phase. Moreover, additional learnable tokens are introduced during the attention computation step to learn sampling weights for each bin. Given the binning strategy and the bin sampling weights learned for each shape, this approach enables the acquisition of shape-specific sampling strategies, taking into account the unique properties and distribution patterns of individual shapes.

An illustration of the three sampling policies is provided in Figure 3.7. The Top-M sampling policy samples the points with larger sampling scores directly. The Boltzmann sampling policy samples points randomly according to their converted sampling probabilities. The bin-based sampling policy further builds upon that. It first partitions the point set into several bins, and then samples points within each bins. In each bin, either top-M sampling or Boltzmann sampling can employed. In our case, we use the Boltzmann sampling.

By incorporating the bin-based sampling policy into the framework, we aim to overcome the limitations of previous sampling methods and improve the overall performance of point cloud sampling. This policy allows for more fine-grained control over the sampling process, tailoring it to the specific characteristics of each shape. Through the adaptive partitioning of points and the application of learnable tokens, the proposed method enhances the ability to capture shape-specific details and variations during sampling, leading to more accurate and representative samples for downstream tasks.

3.2 *k*NN and Pair-wise Distance Computation

In point cloud deep learning, the selection of neighbor points is crucial when utilizing local information. The most commonly employed method for this purpose is *k*-Nearest Neighbor (*k*NN), which allows us to identify the k closest points to a given point, forming a local neighborhood that captures its surrounding context. To implement *k*NN, the pairwise distances between points are computed using various distance metrics such as Euclidean distance or cosine similarity. The k closest neighbors are then selected based on these computed distances. In the context of computing with tensors, when calculating pairwise distances between vectors or tensors, it is often beneficial to use the squared Euclidean distance as the metric. However, when it comes to the actual implementation of such a metric in code, certain challenges may arise that require careful consideration and resolution.

The most intuitive computational method, Method $\mathscr{A}$, is outlined in Listing 3.1. It computes such squared Euclidean distance directly. However, Method $\mathscr{A}$ involves the term `diff`, a torch tensor of size (B, N, M, C), requiring an excessively large amount of Video Random-Access Memory (VRAM) space. This results in significant memory access times and can sometimes render the computation infeasible.

Listing 3.1: Method $\mathscr{A}$: Direct Calculation of Pair-wise Distances

```
# a.shape = (B,N,C)
# b.shape = (B,M,C)
diff = torch.unsqueeze(a, dim=1) - torch.unsqueeze(b, dim
    =2)
# diff.shape == (B, N, M, C)
pairwise_distance = torch.sum(diff ** 2, dim=-1)
# pairwise_distance.shape == (B, N, M)
```

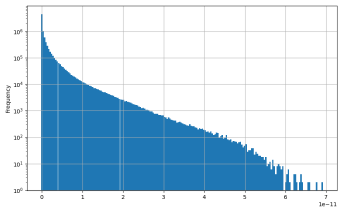
Therefore, Method $\mathscr{B}$, presented in Listing 3.2, is proposed as an alternative. It computes the squared Euclidean distance through factorization. Method

$\mathscr{B}$ is mathematically equivalent to Method $\mathscr{A}$ but is more memory-efficient since the largest tensor involved in Method $\mathscr{B}$ is only of size $(\boldsymbol{B}, \boldsymbol{N}, \boldsymbol{M})$. This reduction in tensor size can significantly save VRAM space and reduce memory access time.

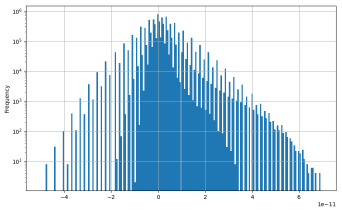
Listing 3.2: Method $\mathscr{B}$: Pair-wise Distances Calculation through Factorization

```
# a.shape = (B,N,C)
# b.shape = (B,M,C)
inner = 2 * torch.matmul(a, b.transpose(2, 1))
# inner.shape == (B, N, M)
aa = torch.sum(a ** 2, dim=2, keepdim=True)
# aa.shape == (B, N, 1)
bb = torch.sum(b ** 2, dim=2, keepdim=True)
# bb.shape == (B, M, 1)
pairwise_distance = aa - inner + bb.transpose(2, 1)
# pairwise_distance.shape == (B, N, M)
```

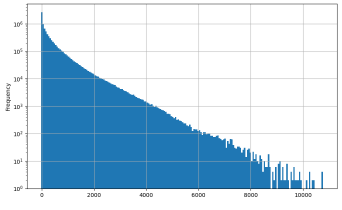
Nevertheless, in practical calculations, the two methods are not strictly equivalent due to computational and storage limitations. For computations, we used `torch.float32`, which is not infinitely precise and introduces rounding errors. During the computation in the shallow layers, all features are of a large magnitude, which poses no issues for Method $\mathscr{B}$. However, when it comes to the deeper layers, a problem arises due to rounding errors. After operations in the preceding layers, the features have been refined, but they have been attenuated in magnitude. As a result, they have become more "similar" on a larger scale, thereby being more severely affected by rounding errors. Figure 3.8 displays example distributions of all elements in `pairwise_distance` from both Method $\mathscr{A}$ and Method $\mathscr{B}$ in a deep layer. The results computed by Method $\mathscr{B}$, which requires more approximations, contain larger rounding errors compared to those from Method $\mathscr{A}$, posing a distinct distribution to the actual pair-wise distance distribution. Moreover, it is evident that the errors in Method $\mathscr{B}$ range between `1e-11` and `1e-13`, significant enough to impact the accuracy of neighbor searches.



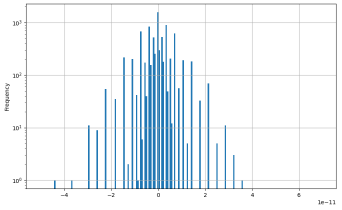
(a) Method $\mathscr{A}$



(b) Method $\mathscr{B}$



(c) Method $\mathscr{C}$



(d) Tensor Diagonal Elements from Method $\mathscr{B}$

Figure 3.8: (a, b, c) The distribution of elements in `pairwise_distance` from Method $\mathscr{A}$, $\mathscr{B}$, and $\mathscr{C}$ respectively. The diagonal elements of `pairwise_distance` from $\mathscr{A}$ and $\mathscr{C}$ are all zeros, while (d) The diagonal elements of `pairwise_distance` from $\mathscr{B}$ deviates from zero due to rounding errors.

Figure 3.8b illustrates an example distribution of elements in `pairwise_distance` from Method $\mathscr{B}$ in a deeper layer. The distribution of the diagonal elements are additionally provided in Figure 3.8d. However, it is important to note that in theory, all diagonal elements should represent a zero distance, as they measure the distance between a point and itself. However, as illustrated in Figure 3.8d, the observed data deviates from zero, and there are instances of negative values arising from rounding errors. These negative values erroneously suggest a negative distance, implying that certain points are closer to each other than to themselves during the processing.

Given the small feature sizes and the necessity of relative distance sizes for neighbor sorting, we introduced a normalization operation before calculating distances, scaling the features to a larger range. The proposed Method $\mathscr{C}$ is outlined in Listing 3.3, and the distribution of pair-wise distances is shown

in Figure 3.8c. The diagonal elements of `pairwise_distance` from it are all zeros. Please note that in this scenario, the absolute magnitudes of the pair-wise distances may change, but their relative magnitudes remain unchanged. Therefore, the kNN points obtained through sorting will not be altered. We use Method $\mathscr{C}$ for our experiments.

Listing 3.3: Method $\mathscr{C}$: Pair-wise Distances Calculation with Factorization and Normalization

```
# a.shape = (B,N,C)
# b.shape = (B,M,C)

a_mean = torch.mean(a, dim=1, keepdim=True)
a = a - a_mean
b = b - a_mean
a_std = torch.mean(torch.std(a, dim=1, keepdim=True), dim
    =2, keepdim=True)
a = a / a_std
b = b / a_std

inner = 2 * torch.matmul(a, b.transpose(2, 1))
# inner.shape == (B, N, M)
aa = torch.sum(a ** 2, dim=2, keepdim=True)
# aa.shape == (B, N, 1)
bb = torch.sum(b ** 2, dim=2, keepdim=True)
# bb.shape == (B, M, 1)
pairwise_distance = aa - inner + bb.transpose(2, 1)
# pairwise_distance.shape == (B, N, M)
```

Overall, the proposed Method $\mathscr{C}$ effectively addresses the potential challenges stemming from memory limitations and rounding errors. This results in an effective approximate computation of the pair-wise distances, enabling the fast and accurate retrieval of k-Nearest Neighbor points through sorting.

3.3 Evaluation Benchmarks

Evaluating point cloud sampling methods involves assessing their effectiveness in capturing the essential characteristics and preserving the information of the original point cloud while reducing its complexity. It is important to employ appropriate evaluation benchmarks to measure the performance of sampling methods. The proposed methods in this dissertation are grounded in a task-oriented learning approach, allowing us to leverage the evaluation metrics defined for the corresponding downstream tasks to assess the performance of the sampling methods. Specifically, we evaluate our proposed sampling methods using three tasks: shape classification, part segmentation, and few-point sampling.

The quality of the sampled point clouds can be evaluated by examining their impact on the classification and segmentation performance. Improved performance in these tasks indicates that the sampled point clouds effectively capture the essential characteristics and distinctive features of the shapes. Higher accuracy and precision demonstrate that the sampling methods produce more representative point clouds, enabling better classification and segmentation results. The few-point sampling task shares similarities with the key point detection task, where the objective is to select a small number of points that carry significant information about the shape. Evaluating the effectiveness of the sampling methods in this task involves assessing their impact on the classification performance using the same test network model. A positive correlation between the quality of the sampling results and the classification performance indicates that the sampled points successfully capture the crucial characteristics of the shape.

In addition to quantitative assessments of the sampling results, qualitative evaluations play a crucial role in demonstrating the effectiveness of the proposed methods. Since our methods are not generative-based and offer traceability, we can easily obtain the indices of the selected points during the sampling process. This traceability enables the visualization of the sampled results, providing a means to analyze the underlying patterns or structures captured by

the sampling methods. If the learned sampling strategy is effective, the qualitative results should reveal discernible and recognizable patterns or structures, further confirming the effectiveness of the sampling methods in preserving important information.

By combining qualitative and quantitative evaluations, we can comprehensively assess the performance of the proposed sampling methods. The qualitative results offer visual insights into the sampled point clouds, while the quantitative metrics from the downstream tasks provide objective measurements of the sampling performance. This comprehensive evaluation framework allows us to validate the effectiveness and suitability of the proposed methods for various point cloud processing tasks.

3.3.1 Shape Classification

Dataset

The ModelNet40 classification benchmark [Wu15] is a widely recognized and extensively used benchmark in the field of 3D shape analysis and understanding. It serves as a standardized evaluation framework for assessing the performance of various algorithms and models on the task of shape classification. ModelNet40 consists of 12,311 3D CAD models from 40 different object categories, including common objects such as chairs, tables, airplanes, and cars. Each shape in the dataset is represented as a point cloud or a mesh, providing a diverse and comprehensive set of 3D shapes for evaluation. The goal of the benchmark is to classify these shapes accurately into their respective object categories. Researchers and practitioners employ various machine learning and deep learning techniques to develop models that can learn discriminative features and effectively classify the shapes. For a fair comparison, we use the official train-test split, in which 9843 models are used for training and 2468 models for testing. From each model mesh surface, points are uniformly sampled and normalized to the unit sphere. Only 3D coordinates are used as point cloud input.

Figure 3.9: Examples of 3D models from the chair category included in the ModelNet dataset. The modelNet40 dataset has 40 categories in total [Wu15].

Data Augmentation

Point cloud data augmentation is a crucial technique in the field of 3D computer vision and point cloud analysis. It involves applying a set of transformations and modifications to the original point cloud data to artificially increase the diversity and variability of the dataset. By introducing these augmentations, the model can learn to be more robust and generalize better to unseen data. In our experiments, we employ common point cloud data augmentation techniques including rotation, translation, scaling, and jittering. During the test, no data augmentation was applied.

The input point cloud data are first normalized into a standardized range, typically [-1, 1], to ensure that all dimensions have equal importance and that the data is centered around zero. Subsequently, one of the following four augmentation operations is applied to each input point cloud with a random parameter setting:

- **Rotation.** The input point cloud undergoes rotation with an angle ranging in [-15°, 15°].

- **Translation.** The input point cloud undergoes translation with a value ranging in [-0.2, 0.2] along the three axes respectively.
- **Scaling.** The input point cloud undergoes scaling with a factor ranging in [0.66, 1.5].
- **Jittering.** The input point cloud undergoes jittering, which is applied based on a normal distribution with a mean of 0 and a standard deviation of 0.01. Additionally, a clipping threshold of 0.05 is set to limit the magnitude of the jittering.

Loss

In the ModelNet40 classification task, the loss function plays a critical role in training the models to accurately classify 3D shapes into their respective object categories. The most commonly used loss function for this task is the categorical Cross-entropy (CE) loss, which can be defined as:

$$\mathcal{L}_{\text{CE}}(\mathbf{y}^{\text{gt}}, \mathbf{y}^{\text{pred}}) = -\sum_{cls=1}^{N_{\text{classes}}} y_{cls}^{\text{gt}} \log\left(y_{cls}^{\text{pred}}\right), \tag{3.1}$$

where $\mathbf{y}^{\text{gt}} = (y_1^{\text{gt}}, y_2^{\text{gt}}, \ldots, y_{N_{\text{classes}}}^{\text{gt}})$ represents the ground truth label distribution and its each element y_{cls}^{gt} is a binary indicator (0 or 1) that indicates if class label cls is the correct classification for the current input. Meanwhile, $\mathbf{y}^{\text{pred}} = (y_1^{\text{pred}}, y_2^{\text{pred}}, \ldots, y_{N_{\text{classes}}}^{\text{pred}})$ represents the predicted class probabilities and its each element y_{cls}^{pred} is the predicted probability of the current input being classified as class cls.

This loss function compares the predicted class probabilities with the ground truth labels and penalizes the model based on the dissimilarity between the predicted and actual distributions. During training, the model aims to minimize this loss by adjusting its parameters to improve the accuracy of the predictions. The categorical cross-entropy loss encourages the model to assign high probabilities to the correct class and low probabilities to the other classes. The choice of the categorical cross-entropy loss in the ModelNet40

classification task aligns with the goal of maximizing classification accuracy and has been widely adopted in the research community for training models on this benchmark.

Metric

In the ModelNet40 point cloud classification task, the metric used to evaluate the performance of models is the classification Overall Accuracy (OA). It measures the percentage of correctly classified objects out of the total number of objects in the test set. The classification accuracy can be calculated using the following equation:

$$\text{OA} = \frac{TP}{N_{total}}, \tag{3.2}$$

where TP is the number of true positives and N_{total} is the total number of observations. The metric provides a straightforward and intuitive measure of how well the models are able to classify the 3D point cloud shapes into their respective object categories. A higher classification accuracy indicates a better performance of the model in correctly identifying the objects.

3.3.2 Part Segmentation

Dataset

While both the ModelNet [Wu15] and ShapeNet [Cha15] datasets are valuable resources for research in 3D shape analysis, they differ in terms of their primary focus, the types of objects included, and the level of semantic annotations provided. ModelNet is more focused on object recognition and shape classification, while ShapeNet offers a broader scope with rich semantic information and hierarchical part structures for various tasks related to 3D shape understanding and modeling. In 3D computer vision, the ShapeNet part segmentation benchmark is a widely recognized evaluation benchmark for assessing the performance of algorithms and models on the task of part segmentation.

Figure 3.10: Examples of aligned models in the chair, laptop, bench, and airplane subsets in the ShapeNet dataset [Cha15].

The ShapeNet Part segmentation benchmark [Yi16] comprises a diverse collection of 16,880 3D CAD models across 16 object categories, with 14,006 3D models for training and 2,874 for testing, providing a rich and comprehensive dataset for part segmentation tasks. Each shape in the dataset is meticulously annotated with fine-grained part segmentations into 2 to 6 parts, resulting in a total of 50 distinct part labels. These part segmentations enable the evaluation and comparison of algorithms and models in accurately predicting the semantic labels for each point or voxel within the 3D shape. We use the sampled point sets produced in [Qi17a] for a fair comparison with prior work.

Figure 3.11: 16 shape categories are included in the ShapeNet Part segmentation benchmark, with each shape annotated with fine-grained part segmentations [Yi16].

Data Augmentation and Loss

For the ShapeNet part segmentation task, we utilize the same augmentation method employed in the ModelNet40 classification task. The loss function employed in our model remains the cross-entropy loss; however, it operates at the point-wise level rather than the shape-wise level.

Metric

In the ShapeNet part segmentation benchmark, based on the common Intersection-over-Union (IoU) metric, two metrics are commonly used to evaluate the performance of models: instance mean Intersection-over-Union (mIoU) and category mIoU. These metrics provide a comprehensive assessment of the segmentation accuracy at both the instance level and the category level.

IoU is a metric employed to evaluate the accuracy of segmentation tasks. It calculates the ratio of the number of common points between the predicted and the true segmentation to the total number of points in their union.

$$\text{IoU} = \frac{TP}{TP + FP + FN}, \tag{3.3}$$

where TP (True Positives) is the number of points correctly identified as part of the segment, FP (False Positives) is the number of points incorrectly labeled as part of the segment, and FN (False Negatives) is the number of points that are part of the segment but were not identified. This metric can be applied to each individual point cloud shape in the dataset, providing an IoU score for that specific shape. On a broader scale, the metrics of instance mIoU and category mIoU are used when considering the entire dataset.

Instance mIoU assesses the average intersection over union across all instances. It specifically focuses on the model's precision in segmenting each distinct object instance, highlighting its ability to accurately capture the specific parts of each instance. On the other hand, category mIoU evaluates the average intersection over union across all categories. This metric provides an assessment of the model's capability to accurately segment each category, considering the overall performance across different categories. It demonstrates the model's ability to generalize and differentiate between various object categories. These measurements effectively quantify the overlap between the predicted segmentation and the ground truth, providing a clear indication of the model's performance in segmentation tasks.

3.3.3 Few-Point Sampling

In the few-point sampling task for point clouds, the sampling performance is assessed using a classification network, and the metric commonly used is the classification accuracy. While the focus is on sampling, the classification accuracy metric evaluates the ability of the generated point sets to accurately classify the objects represented by the point clouds. It measures the percentage of correctly classified point clouds out of the total number of point clouds in the test set. By utilizing a classification network, the metric provides an indication of how well the sampled points capture the discriminative features necessary for accurate object classification. The goal is to achieve a high classification accuracy, indicating that the sampled points effectively represent the object's essential characteristics and enable reliable classification.

Evaluation Framework

For a fair comparison, we use the same evaluation framework from recent learning-based methods including S-Net [Dov19], SampleNet [Lan20], and LighTN [Wan23]. The framework is illustrated in Figure 3.12. In this dissertation, we follow prior works by using ModelNet40 classification as the evaluation task, and the task network is a simple PointNet. Sampling methods are evaluated with multiple sampling sizes to assess their performance across various scenarios.

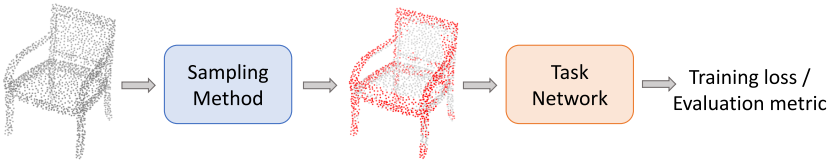


Figure 3.12: Framework for sampling methods evaluation.

Few-Point Sampling and Key Point Detection

Note in this case, the few-point sampling task resembles the key point detection task for point clouds. The evaluation metric expands beyond classification accuracy to include qualitative results and the observation of recognizable

patterns. While classification accuracy remains relevant for assessing the discriminative power of the sampled points, the emphasis shifts a bit toward the ability of the sampling method to capture distinctive and informative key points. A good sampling method should yield point sets where the few sampled points reveal recognizable patterns or salient features that are indicative of the underlying object's shape or structure. Qualitative evaluation allows for visual inspection and assessment of the sampled points, taking into account factors such as point distribution, coverage, and the preservation of important geometric characteristics.

By considering both quantitative metrics, such as classification accuracy, and qualitative observations, researchers can holistically evaluate the performance of point sampling methods, ensuring the generation of representative and informative point sets as "detected" key points for point clouds.

4 APES: Attention-Based Point Cloud Edge Sampling

4.1 The Underlying Pattern: Shape Edges

In the computer vision domain, shape edges are of significant importance as they serve as vital cues for object detection, recognition, and segmentation. Edges represent the boundaries between objects, enabling algorithms to differentiate one object from another or separate objects from the background. By detecting and analyzing the edges in an image or video, computer vision systems can extract valuable information about the underlying shapes and structures present, as illustrated in Figure 4.1. This information can be used to classify objects, estimate their poses, measure dimensions, and even reconstruct 3D models. Additionally, shape edges are often used as input features for various computer vision tasks to improve performance. By leveraging the information encoded in shape edges, computer vision algorithms can better understand the visual scene, leading to improvements in object recognition, scene understanding, and augmented reality applications.

Figure 4.1: CNNs are capable of learning various features from images, including texture, color, and shape silhouette as well as edges.

Convolutional neural networks are powerful models that excel at learning and extracting various features from images, including texture, color, and shape silhouette as well as edges [Kri17, He16, Gei19]. While texture and color provide valuable information, the importance of shape silhouette and edges becomes evident when visualizing the feature maps generated by CNNs. Feature maps highlight the regions of an image that activate specific filters within the network. From the example [1] given in Figure 4.2, it is observed that these feature maps emphasize the edges and boundaries of objects, indicating that shape silhouettes and edges are crucial for CNNs to capture and represent meaningful features. By leveraging these features, especially the shape silhouettes and edges, CNNs can perform tasks such as image classification, object detection, and image segmentation with remarkable accuracy.

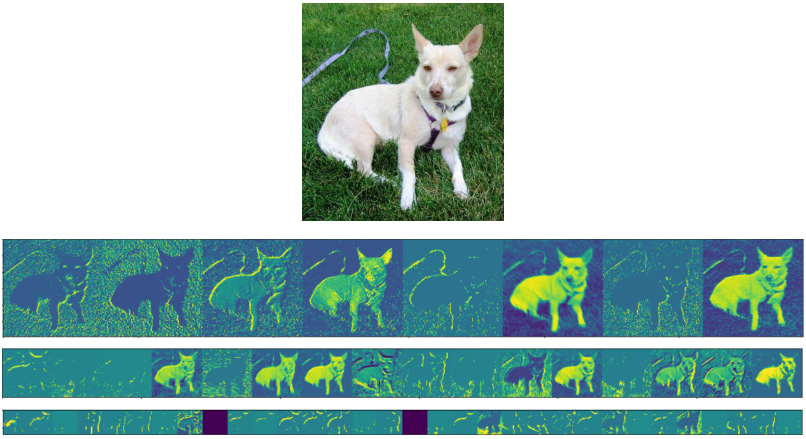


Figure 4.2: Learned feature maps visualized directly from CNN layers.

When dealing with 3D point clouds, neural network models face different challenges compared to traditional 2D image processing. Point clouds typically lack texture information due to their sparse nature, and color information

[1] https://www.analyticsvidhya.com/blog/2020/11/tutorial-how-to-visualize-feature-maps-directly-from-cnn-layers/

may also be unavailable or inconsistent. As a result, shape silhouettes and edges become even more crucial for extracting meaningful features from point clouds. The edges of 3D shapes provide valuable cues for object recognition, segmentation, and reconstruction. In a recent research, a neural network model named PT [Eng21] was introduced with the aim of improving performance on downstream tasks by learning specific patches for point clouds. The visualization results, as depicted in Figure 4.3, provide scientific evidence that the learned key patches predominantly correspond to shape edges. This observation reinforces the significance of shape edges as critical features for point cloud deep learning.

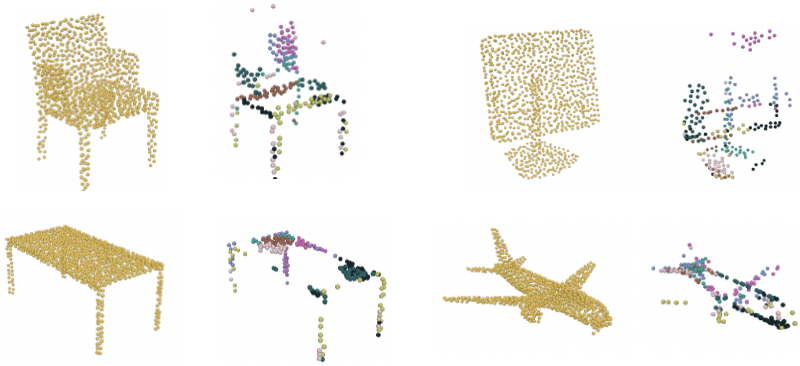


Figure 4.3: Key patches of different point cloud shapes learned from PT [Eng21].

In the context of point cloud sampling for computer vision tasks, selecting edge points as samples can be a promising approach. By prioritizing edge points during sampling, the network can focus on capturing crucial shape information and improve the overall performance of the network model. From a geometric perspective, sampling shape edges may bear some resemblance to traditional curvature-based sampling. However, it is crucial to acknowledge that network models lack prior knowledge of geometry curvature and may have the potential to learn more effective sampling strategies for point clouds. This serves as a key motivation for the introduction of learning-based sampling approaches, as they allow neural networks to acquire implicit features and leverage their

representation power. By incorporating learning-based sampling, network models can discover and exploit intricate relationships within point clouds, leading to improved performance in various computer vision tasks.

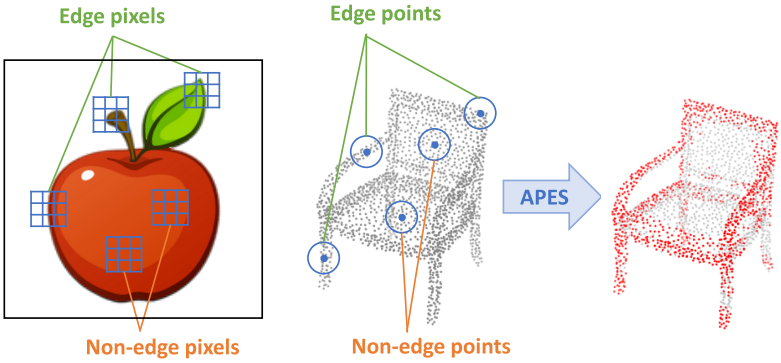


Figure 4.4: Similar to the Canny edge detection algorithm that detects edge pixels in images, our proposed APES algorithm samples edge points which indicate the outline of the input point clouds. The blue grids/spheres represent the local patches for given center pixels/points.

In the remaining part of this chapter, drawing inspiration from the well-established 2D Canny edge detection algorithm, we present a step-by-step derivation process to propose a novel sampling method specifically designed for point clouds, with a primary focus on capturing shape edges.

4.2 Revisiting Canny Edge Detection on Images

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It extracts useful structural information from different vision objects and dramatically reduces the amount of data to be processed, thus has been widely applied in various computer

vision systems. The process of Canny edge detection algorithm can be broken down to five different steps [1] :

1 Apply Gaussian filter to smooth the image in order to remove the noise.

2 Find the intensity gradients of the image.

3 Apply gradient magnitude thresholding or lower bound cut-off suppression to get rid of spurious response to edge detection.

4 Apply double threshold to determine potential edges.

5 Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Figure 4.5 gives an example that shows the progression of an image through each of the five steps.

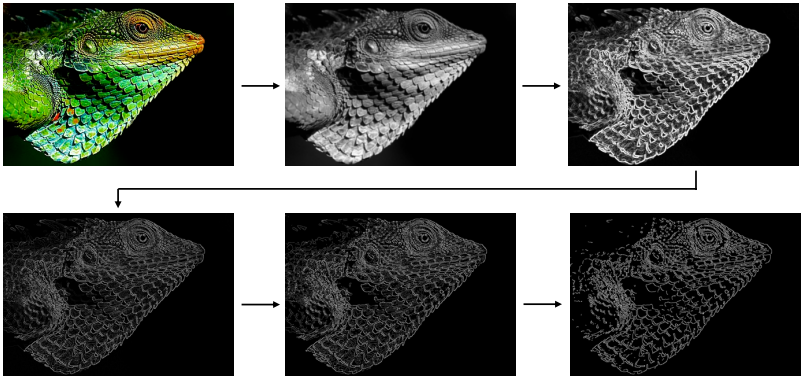


Figure 4.5: Walkthrough of the Canny edge detection algorithm.

To compute pixel intensity gradients, an edge detection operator (e.g., Roberts, Prewitt, or Sobel $\mathbf{S}_x, \mathbf{S}_y$) is applied on the image $\mathbf{I}$ to obtain a value for the first derivative in the horizontal direction $\mathbf{G}_x$ and the vertical direction $\mathbf{G}_y$ with

[1] https://en.wikipedia.org/wiki/Canny_edge_detector

convolution, which is mathematically formulated as:

$$\mathbf{G}_x = \mathbf{S}_x * \mathbf{I},\ \mathbf{G}_y = \mathbf{S}_y * \mathbf{I}\,. \tag{4.1}$$

From this, the edge gradient and direction can be determined:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}\,, \tag{4.2}$$

$$\mathbf{\Theta} = \arctan(\mathbf{G}_y/\mathbf{G}_x)\,. \tag{4.3}$$

The key to the effectiveness of the Canny edge detector is how edge pixels are defined, i.e., the pixels with large intensity gradients. The intensity gradient of each pixel i is computed in comparison to its neighbors in a patch set $\mathcal{S}_i$, which is typically a 3×3 or 5×5 patch. Pixels with larger intensity gradients are defined as edge pixels. We make the following observation: *If there are large differences between the pixels from a patch set $\mathcal{S}_i$, then the standard deviation σ_i of the intensities in the patch is also high.* Hence, an alternative method for edge detection is to select pixels whose patch sets have larger σ_i.

We further generalize beyond pixel intensities to any (latent) per-pixel features $\mathbf{p}_i$ with a "measure of feature correlation" $h(\mathbf{p}_i, \mathbf{p}_{ij})$ defined between the center pixel i and its neighbor pixel j. A larger correlation value indicates more similar pixel features. In each patch $\mathcal{S}_i$, we call the vector $\mathbf{m}_i = \mathrm{softmax}\big(h(\mathbf{p}_i, \mathbf{p}_{ij})_{j \in \mathcal{S}_i}\big)$ the normalized correlation map between the center pixel and its neighbors. Then the standard deviation σ_i is computed over the elements of $\mathbf{m}_i$, and *pixels with larger σ_i are selected as edge pixels.* An illustration is given in the top row of Figure 4.6. When the neighbor number k is fixed (e.g., $k = 9$ for the top row, the center pixel is self-contained as a neighbor), for each patch, the mean value of its normalized correlation map is always $1/k$. However, for edge pixels, the standard deviations of their normalized correlation maps are larger. The measure $h(\cdot, \cdot)$ can be defined directly on the pixel color values, or the pixel latent features if neural networks are used.

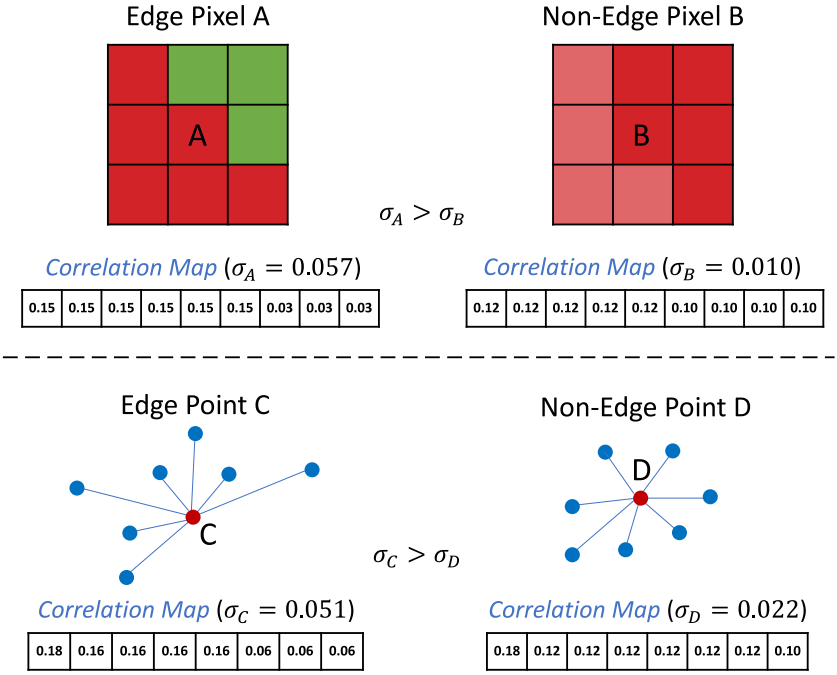


Figure 4.6: Illustration of using standard deviation to select edge pixels/points. A normalized correlation map is computed between the center pixel/point and its neighbors. The center pixel/point is self-contained as a neighbor. A larger standard deviation in the normalized correlation map means a higher possibility that it is an edge pixel/point.

For images, the proposed alternative edge detection algorithm, and in particular using the standard deviation for the normalized correlation map, is computationally much more expensive compared to the Canny edge detector. However, it provides the starting point to transfer the idea to point cloud edge sampling. Unlike images where pixels are well-aligned and patch operators can be easily defined and applied, point clouds are usually irregular, unordered, and potentially sparse. Voxel-based 3D convolution kernels are not applicable. Moreover, image pixels come with a color value (e.g., RGB or grayscale). For many point clouds, however, the point coordinates are the only available feature.

4.3 Point Cloud Edge Sampling

4.3.1 Local-Based APES

To adopt the previously introduced alternative edge detection algorithm to a point cloud set with $|S| = N$ points, we use k-nearest neighbor to define a local patch $S_i \subseteq S$ for each point i to compute normalized correlation maps. As illustrated in the bottom row of Figure 4.6, when the neighbor number k is fixed (e.g., $k = 8$ for the bottom row, the center point is self-contained as a neighbor), for each patch, the mean value of its normalized correlation map is again always $1/k$. However, for edge points, the standard deviations of their normalized correlation maps are larger.

On the other hand, the attention mechanism is an ideal option to serve as the "measure of correlation" between point features within each patch, i.e., *the attention map serves as the normalized correlation map* directly. The local-based correlation measure $h^l(\cdot, \cdot)$ is defined as

$$h^l(\mathbf{p}_i, \mathbf{p}_{ij}) = Q(\mathbf{p}_i)^\mathsf{T} K(\mathbf{p}_{ij} - \mathbf{p}_i) \tag{4.4}$$

where Q and K stand for the linear layers applied on the query input and the key input, respectively. Here we use the (latent) features of the center point $\mathbf{p}_i$ as the query input, and the feature difference between the neighbor point and the center point $\mathbf{p}_{ij} - \mathbf{p}_i$ as the key input. As in the original Transformer model [Vas17], the square root of the feature dimension count $\sqrt{d}$ serves as a scaling factor. The final normalized correlation map $\mathbf{m}_i^l$ is given as

$$\mathbf{m}_i^l = \mathrm{softmax}\left(h^l(\mathbf{p}_i, \mathbf{p}_{ij})_{j \in S_i} / \sqrt{d}\right) . \tag{4.5}$$

Again, a standard deviation σ_i is computed for each normalized correlation map. *The edge points are sampled by selecting the points with higher* σ_i. A key requirement for point cloud deep learning is that the model should be invariant to the point order of the raw point cloud. Our proposed method successfully fulfilled this property.

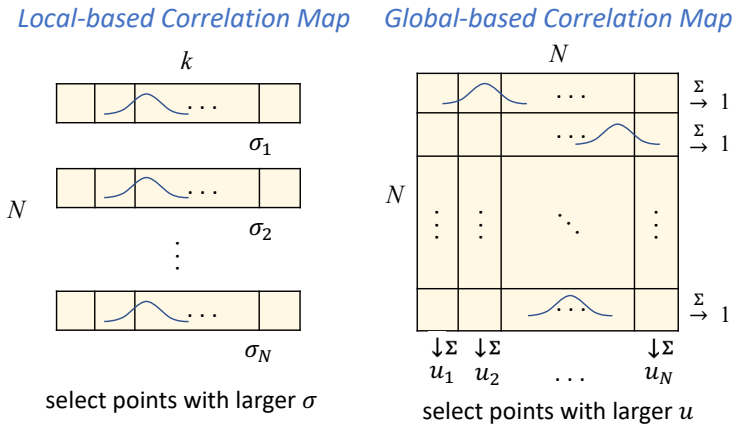


Figure 4.7: The key idea of proposed methods. N denotes the total number of points, while k denotes the number of neighbors used for local-based sampling method.

4.3.2 Global-Based APES

We term the above-applied attention as neighbor-to-point (N2P) attention, which is specifically designed to capture local information using patches. For sampling problems, global information is also crucial. Considering the special case where all points are included in the local patch (i.e., $k = N$), a new global correlation map $\mathbf{M}^g$ of size $N \times N$ is obtained with the linear layers Q and K being shared for all points. Now the N2P attention simplifies into the common self-attention. We term it point-to-point (P2P) attention in this paper. In this case, the correlation measure $h^g(\cdot, \cdot)$ and the normalized correlation map are defined as:

$$h^g(\mathbf{p}_i, \mathbf{p}_j) = Q(\mathbf{p}_i)^\top K(\mathbf{p}_j) \tag{4.6}$$

$$\mathbf{m}_i^g = \mathrm{softmax}\left(h^g(\mathbf{p}_i, \mathbf{p}_j)_{j \in S} / \sqrt{d}\right) \tag{4.7}$$

Note that all $\mathbf{m}_i^g$ now have the same point order, but the attention result for each point pair is not affected by the order.

The global correlation map $\mathbf{M}^g$ regroups the point-wise normalized correlation maps into a $N \times N$ matrix:

$$\mathbf{M}^g = \begin{pmatrix} — & \mathbf{m}_1^{g\,\mathsf{T}} & — \\ — & \mathbf{m}_2^{g\,\mathsf{T}} & — \\ & \vdots & \\ — & \mathbf{m}_N^{g\,\mathsf{T}} & — \end{pmatrix} \tag{4.8}$$

In the context of the global correlation map $\mathbf{M}^g$, instead of computing row-wise standard deviations for selecting points, we propose an alternative approach. Denote m_{ij} as the value of ith row and jth column in $\mathbf{M}^g$. For point i, if it is an edge point, $\mathbf{m}_i^g$ has a larger standard deviation. In this case, considering its neighboring area, if a point j is close (based on 3D spatial space or latent space) to point i, m_{ij} is larger and point j is also likely to be an edge point. Given this property, now consider $\mathbf{M}^g$ column-wise. For a point j, in order to qualify it as an edge point, it needs to rank a higher value of m_{ij} in $\mathbf{M}^g$ more often compared to other points. Hence *instead of computing row-wise standard deviations, we compute column-wise sums.* Denote $u_j = \sum_i m_{ij}$, we then *sample the points with higher* u_j. Overall, we can consider it as follows: if a point contributes more in the self-attention correlation map, it is more likely to be an "important" point. An illustrative figure of the two proposed methods is given as Figure 4.7.

4.4 Experimental Results

In this section, we conduct a thorough evaluation of the proposed APES method on two prominent datasets: ModelNet40 for 3D object classification and ShapeNet Part for 3D object segmentation. Furthermore, we extend our analysis to include a comparison with other prevalent point cloud sampling methods in the context of the few point sampling task. The results encompass both local-based APES and global-based APES approaches, providing a comprehensive understanding of their respective performance. Apart from

quantitative results, our evaluation also presents a range of qualitative results, offering visual insights into the quality of the sampled point clouds. To gain further insights, we conduct several ablation studies, systematically exploring the impact of various components and parameters on APES' performance, thus contributing to a comprehensive analysis of this innovative sampling technique.

4.4.1 Classification

Network Architecture Design

The network architecture of a point cloud deep learning model comprises various layers designed with specific objectives. These include the embedding layer, feature learning layer, feature pooling layer, output layer, and potentially downsampling and upsampling layers. The embedding layer serves to transform the input 3D coordinates into higher-dimensional features. Feature learning layers, often used iteratively, are possibly faciliated with downsampling or upsampling layers to enable the learning of deeper implicit features. Feature pooling layers are commonly employed in tasks that necessitate shape-wise outputs, such as classification. Finally, the output layer typically takes the form of a MLP responsible for generating the desired final result. The classification network architecture we use for our experiments is given in Figure 4.8. The optional residual links are used for better feature transmission.

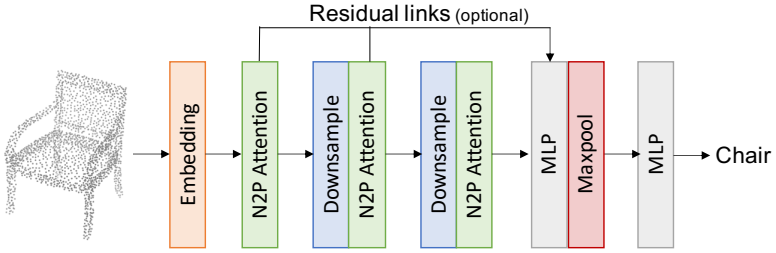


Figure 4.8: Network architecture for the ModelNet40 classification task.

The embedding layer converts the input 3D coordinates into a higher-dimensional feature with multiple EdgeConv blocks, with a pre-defined dimension number. Balancing the trade-offs, in most of our experimental settings, it consists of 2 EdgeConv blocks and outputs an embedding of 128 dimensions for each point. For feature learning layers, there are multiple choices. It is possible to use the layers designed for a similar purpose in other papers, e.g., Linear MLP, EdgeConv, KPConv, or Point Transformer block. Alternatively, the aforementioned N2P attention or P2P attention can also be used as feature learning layers. Their designs are given in Figure 4.9. We use 32 neighbor points for N2P attention blocks and the multi-head strategy is applied with 4 heads. A feed-forward network is added after each attention block in the feature learning layers as the original Transformer model did.

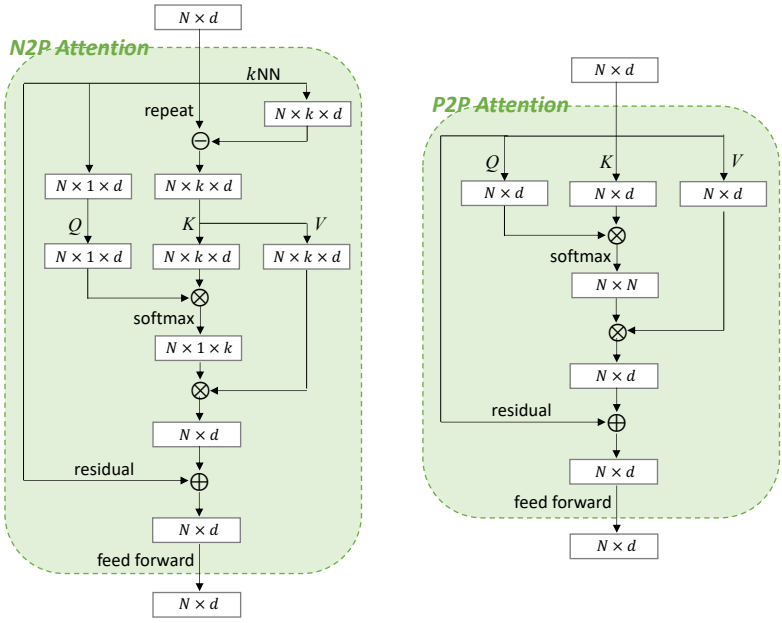


Figure 4.9: The network structures of N2P attention feature learning layer (left) and P2P attention feature learning layer (right). We use N2P layer as the default feature learning layer for most experiments.

The design of downsampling layers is given in Figure 4.10. We use $k = 32$ neighbor points as default in local-based APES downsample layers. For an input point cloud of N points from the previous layer, each downsampling layer samples it to $N/2$ points. Note that our method can sample the point cloud to any desired number of points.

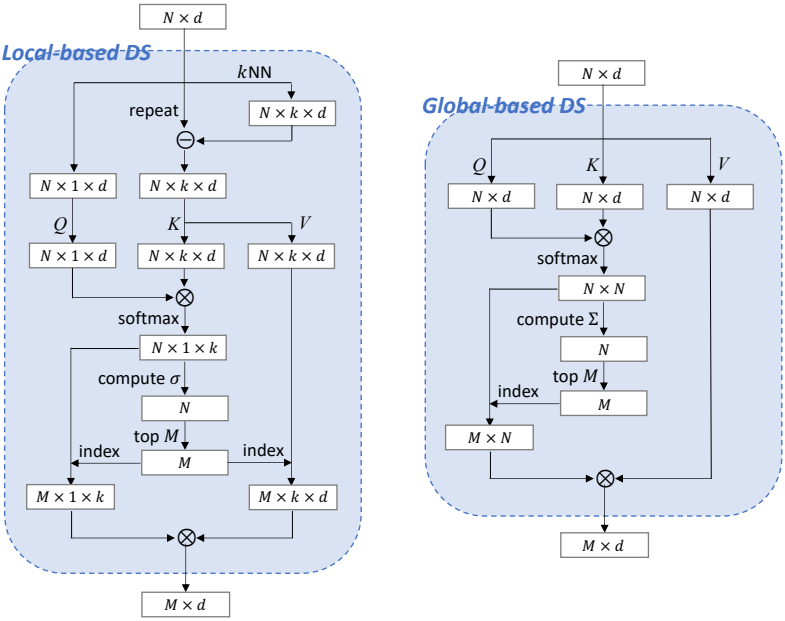


Figure 4.10: The network structures of two alternative downsampling layers: local-based downsampling layer (left) and global-based downsampling layer (right). Both kinds of downsampling layers downsample a point cloud from N points to M points, while upsample layer upsamples it from M points to N points.

Traceable Sampling and Differentiability

The indexing operation in neural networks naturally disables the gradient backpropagation, which is why most previous learning-based point cloud sampling methods rely on generation-based approaches, resulting in untraceable

sampled points [Dov19, Lan20, Lin21, Qia20, Wan21, Wan23]. Therefore, the design of a learning-based sampling method that achieves both differentiability for enabling end-to-end training with neural networks and preserves the traceability of sampled points poses a significant challenge. Compared to those state-of-the-art methods, the main advantage of our method is that it enables the gradient backpropagation during the training of sampling.

Our method can be divided into three steps: Computing the correlation map, aggregating the correlation map row-wise (local APES) or column-wise (global APES) to a per-point importance evaluation, and lastly selecting the most important points as samples. Computing the correlation map is fully differentiable. The last two steps, similar to FPS, do not contain learnable parameters. With the obtained indices, point indexing is basically a Top-M pooling operation, which is differentiable, just as the widely used max-pooling operations. As illustrated in Figure 4.11, the gradient follows the green line, while the orange block contains no learnable parameters and thus does not need to be differentiable.

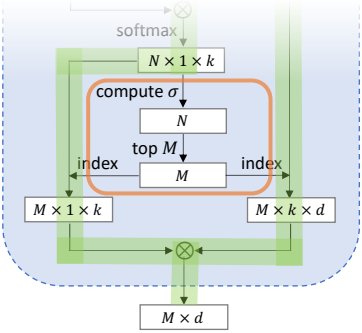


Figure 4.11: Gradient backpropagation is enabled in our sampling layer.

Training Details

To train the model, we use AdamW optimizer with an initial learning rate 1×10^{-4} and decay it to 1×10^{-8} with a cosine annealing schedule. The

weight decay hyperparameter for network weights is set as 1. Dropout with a probability of 0.5 is used in the last two fully connected layers. A common cross-entropy loss is used as the task loss. We train the network with a batch size of 8 for 200 epochs.

Quantitative and Qualitative Results

The quantitative comparison with state-of-the-art (SOTA) methods is briefly summarized in Table 4.1, showcasing the impressive performance of our proposed method. This evaluation demonstrates that APES stands among the top-performing techniques in the field. Complementing the quantitative analysis, we present qualitative results in Figure 4.12, providing visual evidence of the efficacy and accuracy of APES in capturing and representing intricate details of 3D objects.

Table 4.1: Classification results on ModelNet40. In comparison with other SOTA methods that also only use raw point clouds as input.

Method	Overall Accuracy
PointNet [Qi17a]	89.2%
PointNet++ [Qi17b]	91.9%
SpiderCNN [Xu18]	92.4%
DGCNN [Wan19]	92.9%
PointCNN [Li18b]	92.2%
PointConv [Wu19]	92.5%
PVCNN [Liu19c]	92.4%
KPConv [Tho19]	92.9%
PointASNL [Yan20a]	93.2%
PT[1] [Eng21]	92.8%
PT[2] [Zha21c]	93.7%
PCT [Guo21]	93.2%
PRA-Net [Che21b]	93.7%
PAConv [Xu21]	93.6%
CurveNet [Muz20]	**93.8%**
DeltaConv [Wie22]	**93.8%**
APES (local-based)	93.5%
APES (global-based)	**93.8%**

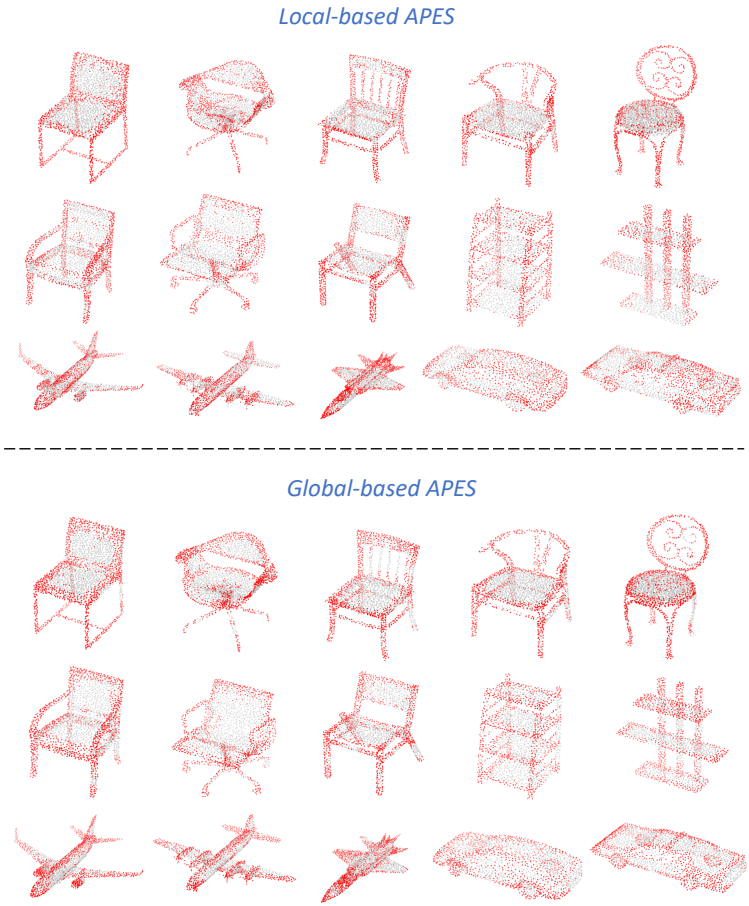


Figure 4.12: Visualized sampling results of local-based APES and global-based APES on different shapes. All shapes are from the test set.

From it, we can observe that both local-based APES and global-based APES achieve good edge sampling results. On the other hand, local-based APES focuses more strictly on edge points, while global-based APES ignores some edge points and leverages a bit more on other non-edge points that are close to the edges. For example, in the case of chair shape illustrated in Figure 4.13,

the global-based APES approach exhibits a particular behavior. It tends to discard some points corresponding to chair legs, which are typically thin enough to be categorized as edges and would have been selected by the local-based APES approach (the blue circle). Conversely, global-based APES selects additional points along the edges of the chair seat, effectively making the edges appear "thicker"(the green circles). While the local-based APES approach could achieve a similar outcome, the total number of sampled points is limited, thus influencing the extent of the edge thickness of the chair seat.

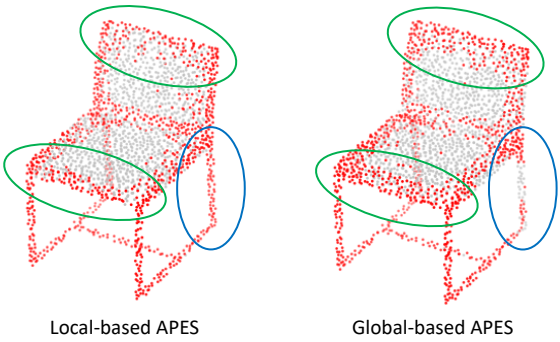


Figure 4.13: A detailed comparison between local-based APES and global-based APES, using a typical chair shape as an example.

Interestingly, while local-based APES achieves more ideal point cloud edge sampling results qualitatively, global-based APES achieves slightly better quantitative results on the classification metric. This is probably because some non-edge points can also provide important information. Overall, sampling more edge points improves the performance of downstream tasks. However, this can be overdone, and selecting only edge points can be detrimental. APES uses end-to-end training that includes the downstream task to make a good trade-off in the sample selection. Local-based APES imposes stronger mathematical statistics constraints during the task loss minimization, while global-based APES pursues better performance by allowing sampling the points that are less belong to the edge yet more important globally.

4.4.2 Segmentation

Network Architecture Design

The segmentation network architecture is given in Figure 4.14.

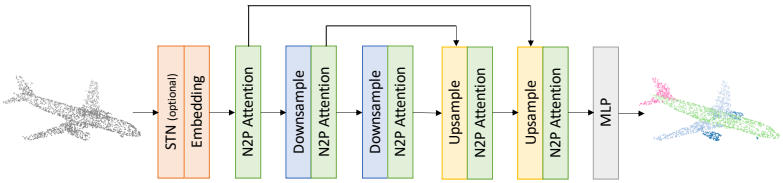


Figure 4.14: Network architecture for the Shapenet Part segmentation task.

Most network layers are identical to the layers in the classification model, except for the spatial transform network (STN) and the upsample layer. The optional STN layer learns a spatial transformation matrix to transform the input cloud for better initial alignment [Qi17a, Wan19]. The upsample layer is a cross-attention-based layer to map the input point cloud to an upsampled size. Its key and value input is the feature from the last layer, while the query input is the corresponding residual feature within the downsampling process.

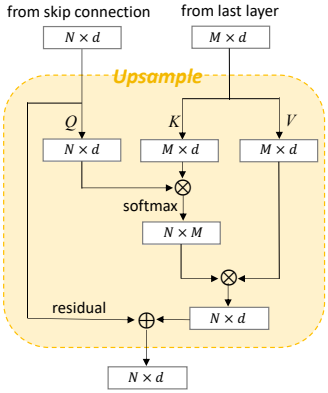


Figure 4.15: The network structure of the upsampling layer. While a downsampling layer downsamples a point cloud from N points to M points, an upsampling layer upsamples it from M points back to N points.

Training Details

To train the model, we use AdamW optimizer with an initial learning rate 1×10^{-4} and decay it to 1×10^{-8} with a cosine annealing schedule. The weight decay hyperparameter for network weights is set as 1×10^{-4}. The dropout with a probability of 0.5 is used in the last two fully connected layers. A common cross entropy loss is used as the task loss. We train the network with a batch size of 16 for 200 epochs.

Quantitative and Qualitative Results

The segmentation quantitative results are given in Table 4.2. Our method achieves decent performance but is not on par with the best ones.

Table 4.2: Segmentation results on ShapeNet Part.

Method	Cat. mIoU	Ins. mIoU
PointNet [Qi17a]	80.4%	83.7%
PointNet++ [Qi17b]	81.9%	85.1%
SpiderCNN [Xu18]	82.4%	85.3%
DGCNN [Wan19]	82.3%	85.2%
SPLATNet [Su18]	83.7%	85.4%
PointCNN [Li18b]	84.6%	86.1%
PointConv [Wu19]	82.8%	85.7%
KPConv [Tho19]	85.0%	86.2%
PT[1] [Eng21]	-	85.9%
PT[2] [Zha21c]	83.7%	**86.6%**
PCT [Guo21]	-	86.4%
PRA-Net [Che21b]	83.7%	86.3%
PAConv [Xu21]	84.6%	86.1%
CurveNet [Muz20]	-	**86.6%**
StratifiedTransformer [Lai22]	**85.1%**	**86.6%**
APES (local-based)	83.1%	85.6%
APES (global-based)	83.7%	85.8%

However, as we compute the same metrics on the intermediate downsampled point clouds in Table 4.3, we surprisingly find that their performances are extremely good, even far better than the SOTA methods. This indicates the downsampled edge points contribute more to the performance, while the features of the discarded non-edge points are not well reconstructed by the upsample layer. The key weakness of our current segmentation network is the upsample layer.

Table 4.3: Segmentation results of the full point clouds and intermediate downsampled point clouds of different sizes.

Method \ Points	Cat. mIoU (%)			Ins. mIoU (%)		
	2048	1024	512	2048	1024	512
APES (local)	83.11	85.56	**86.17**	85.58	87.27	87.41
APES (global)	83.67	84.86	85.44	85.81	87.78	**88.06**

Most other neural network papers use FPS for downsampling and FPS preserves a similar data distribution compared to the original point cloud. When upsampling, simple interpolation operations [Qi17b, Zha21c, Lai22] are used to create new points. However, our method focuses on edge points and the sampled result has a quite different data distribution than the original point cloud. For non-edge points, especially those far from edges, neighbor-based interpolation methods are no longer applicable. We have designed a cross attention-based layer for upsampling, but it is still hard to get the features of the former discarded points back, even with residual links. Note that in this case, the upsampling problem actually turns into a point cloud completion or reconstruction task, which is another advanced topic for point cloud analysis and out of the scope of this work.

The qualitative segmentation results are given in Figure 4.16, with intermediate visualization results also provided simultaneously. Our proposed APES method successfully samples edge points, highlighting its effectiveness in capturing crucial geometric features during the sampling process.

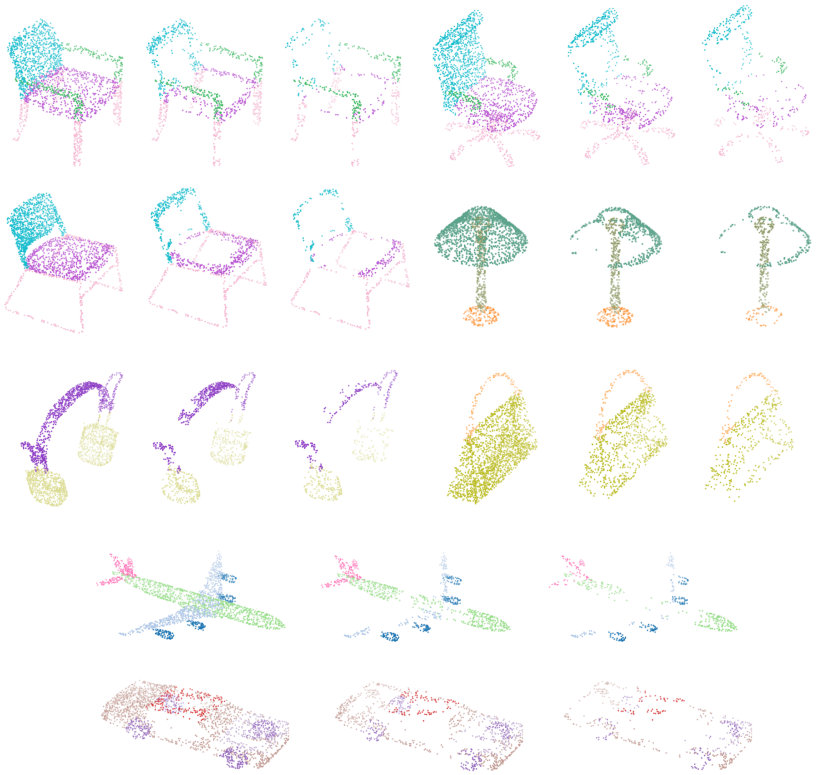


Figure 4.16: Visualized segmentation results as shape point clouds are downsampled. All shapes are from the test set.

4.4.3 Few Point Sampling

Experiment Setting

We additionally compare our sampling method to previous work including RS, FPS, and the more recent learning-based S-Net, SampleNet, LighTN, etc. The same evaluation framework from [Dov19, Lan20, Wan23] is used, as

illustrated in Figure 3.12. The task here is the ModelNet40 Classification, and the task network is PointNet. Sampling methods are evaluated with multiple sampling sizes.

As discussed in the results part of Section 4.4.2, edge point sampling changes the data distribution compared to the original point cloud, especially when a large downsampling ratio (defined as N/M) is used. Hence for a fair comparison, in order to achieve a downsampled point cloud size of M, we first sample the input point cloud to a size of $2M$ with FPS, then sample it to the desired size M with our method APES.

Quantitative and Qualitative Results

Quantitative results are presented in Table 4.4, revealing the strong classification performance of both the local-based and global-based APES methods across various sampling ratios. Notably, our method exhibits significant performance gains, particularly when only a limited number of points are sampled.

Table 4.4: Comparison with other sampling methods. Evaluated on the ModelNet40 classification benchmark with multiple sampling sizes.

M	Voxel	RS	FPS [Eld97]	S-NET [Dov19]
512	73.82	87.52	88.34	87.80
256	73.50	77.09	83.64	82.38
128	68.15	56.44	70.34	77.53
64	58.31	31.69	46.42	70.45
32	20.02	16.35	26.58	60.70
M	PST-NET [Wan21]	SampleNet [Lan20]	MOPS-Net [Qia20]	DA-Net [Lin21]
512	87.94	88.16	86.67	89.01
256	83.15	84.27	86.63	86.24
128	80.11	80.75	86.06	85.67
64	76.06	79.86	85.25	85.55
32	63.92	77.31	84.28	85.11
M	LighTN [Wan23]	APES (local)	APES (global)	
512	89.91	90.79	**90.81**	
256	88.21	90.38	**90.40**	
128	86.26	89.73	**89.77**	
64	86.51	88.68	**89.57**	
32	86.18	86.49	**88.56**	

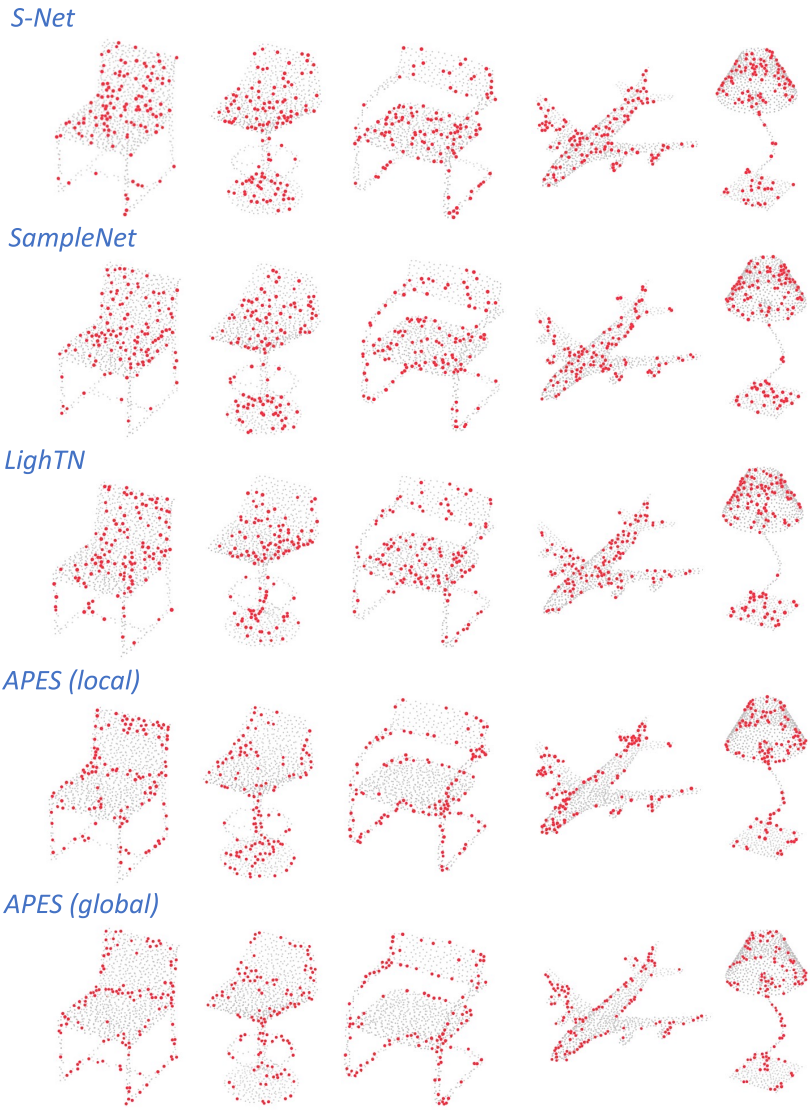


Figure 4.17: Qualitative comparison for the sampling of 128 points from input point clouds. The results from APES exhibit a clear pattern of sampling shape edge points.

In addition to the quantitative evaluation, we provide further qualitative insights through visualizations in Figure 4.17. While other learning-based methods may achieve satisfactory numerical results, it is challenging to discern their underlying sampling patterns from the visualizations. Their results often resemble random sampling, lacking a discernible structure. In contrast, our proposed method exhibits a distinct and coherent sampling pattern, effectively capturing the outlines of the point clouds. This comprehensive sampling approach sets our method apart, emphasizing its ability to extract meaningful geometric information from the point cloud data.

4.4.4 Ablation Study

In this subsection, multiple ablation studies are conducted regarding the design choices of neural network architectures. All following experiments are performed on the classification benchmark of ModelNet40.

Feature Learning Layer

The feature learning layer we used in the above experiments is the N2P attention layer. However, as discussed in Section 4.4.1, it is possible to replace it with other feature layers. We additionally report the results of using EdgeConv or P2P attention as the feature learning layer in Table 4.5. From it, we can observe that N2P attention achieves the best performance. Meanwhile, the results of using EdgeConv are improved when using our proposed sampling methods.

Table 4.5: Ablation study of using different feature learning layers in the classification network.

Method	Feature Learning Layer	OA (%)
DGCNN	EdgeConv	92.90
APES (local-based)	EdgeConv	93.02
	P2P Attention	93.30
	N2P Attention	**93.47**
APES (global-based)	EdgeConv	93.18
	P2P Attention	93.46
	N2P Attention	**93.81**

Embedding Dimension

In most network-based methods, it is often reported that better performances are achieved when a larger embedding dimension is used. In our experiments, we use an embedding dimension of 128 as the default. We additionally report the results of using embedding dimensions of 64 and 192 in Table 4.6. Performance generally improves as the embedding dimension increases. However, when $d = 192$, the improvement becomes marginal compared to $d = 128$. Therefore, considering the trade-off between accuracy and computational efficiency, we adopt $d = 128$ for most experiments.

Table 4.6: Ablation study of using a different number of embedding dimensions for the classification task.

Method	Embedding Dimension	OA (%)
APES (local-based)	64	93.10
	128	93.47
	192	**93.54**
APES (global-based)	64	93.34
	128	93.81
	192	**93.83**

Choice of k in Local-Based APES

When local-based APES is used, the parameter of neighbor number k is a very important parameter since it decides the perception area size of local patches. We additionally report the results of using different k in Table 4.7. Performance generally improves as k increases. However, larger k values incur substantial computational overhead. Hence, following the choice of many prior works that also used the kNN method, we adopt $k = 32$ for most of our experiments.

Table 4.7: Ablation study of using a different number of neighbors for local-based edge point sampling.

k	8	16	32	64	128	256	512
OA (%)	93.14	93.26	93.47	93.52	93.54	93.59	**93.63**

Successive Sampling vs. Direct Sampling

An advantage of our proposed method is that we can sample any desired number of points with it. We further provide qualitative comparison results of successively sampling the raw point cloud to a quarter ($N \rightarrow N/2 \rightarrow N/4$) and directly sampling it to a quarter ($N \rightarrow N/4$) in Figure 4.18. We observe that the sampled results are mostly similar. With the exception of a few extreme edge points which are better captured by successive sampling.

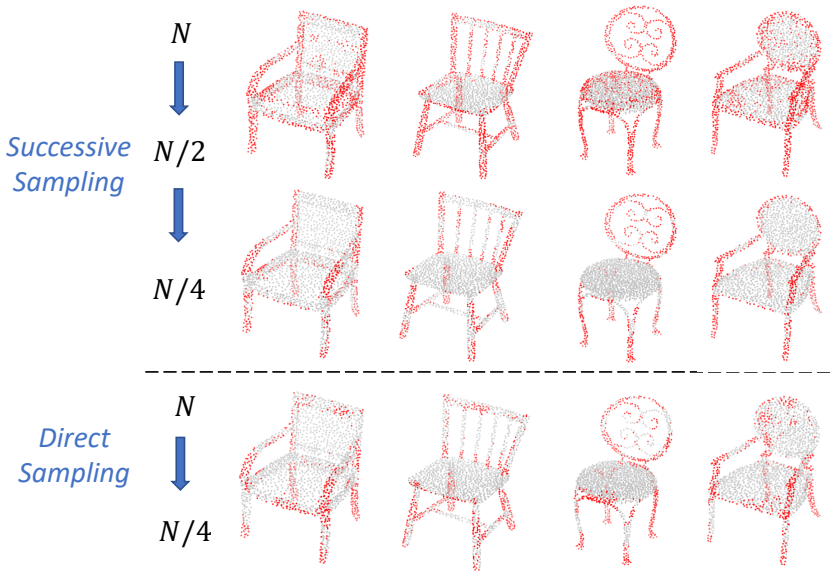


Figure 4.18: Sampling results of successively sampling to a fourth of the original size and directly sampling by a factor of four.

Additional Edge Point Supervision

Since it is possible to compute "ground-truth" edge points from the shapes using local curvatures, we further study the cases where an edge supervision loss term is introduced. Experiments of not using the edge supervision, using it for pre-training (and fixing it during the downstream task training), and using

it for joint training are conducted. Numerical results are given in Table 4.8. The results are consistent with our conclusion in Section 4.4.1. For local-based APES which already focuses on edge point sampling, edge supervision has no significant impact. However, for global-based APES, edge supervision decreases performance slightly.

Table 4.8: Ablation study of considering the edge supervision. Results of using it for pre-training or joint training are both presented.

Edge Supervision	None	Pre-trained and Fixed	Joint Training
APES (local-based)	93.47%	93.45%	93.46%
APES (global-based)	**93.81%**	93.47%	93.51%

4.5 Summary

This chapter introduces a novel point cloud edge sampling method called APES, derived from the well-recognized Canny edge detection algorithm originally developed for images. APES leverages attention-based mechanisms to compute correlation maps and effectively sample edges in point clouds, offering a promising approach for edge detection in 3D data. Moreover, APES pioneers the integration of task-oriented learning and mathematical traceable direct point selection. It gives a solution that innovatively enables gradient backpropagation during the training while maintaining the traceability of sampled points. Two variations of local-based APES and global-based APES are proposed based on different attention modes. Qualitative and quantitative results show that our method achieves excellent performance on common point cloud benchmark tasks.

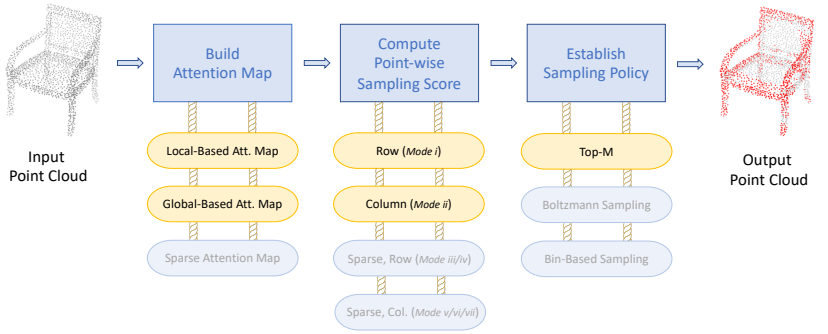


Figure 4.19: In this chapter, APES is proposed by leveraging the attention mechanism. It successfully samples edge points of the input point cloud.

However, based on the findings from their results, while it is evident that increasing the number of sampled edge points generally enhances the performance of downstream tasks, it is also important to note that excessive emphasis on edge points alone may have a negative impact. Therefore, there is a need for a more sophisticated sampling method that strikes a better balance between sampling edge points and preserving shape uniformity. In the upcoming chapter, our objective is to develop such an advanced sampling approach that would enable improved trade-offs, ensuring both the extraction of important edge information and the maintenance of overall shape uniformity within the point cloud data.

5 SAMPS: Balancing between Edge Sampling and Global Uniformity Preserving

5.1 Going Beyond APES

By integrating neural network-based learning and mathematical statistics-based direct point selection ingeniously, Attention-based Point cloud Edge Sampling (APES) [Wu23b] introduces an innovative approach to point cloud sampling, with a particular emphasis on the edges of point clouds. Two different methods were proposed in APES based on different attention modes: local-based APES and global-based APES. In both methods, point-wise sampling scores are computed under certain rules from the local or global attention map(s). Then top-M points with larger sampling scores are sampled. From the qualitative and quantitative results of APES, an interesting phenomenon is observed: while local-based APES focuses more rigorously on edge points, its quantitative performance on downstream tasks is mostly not on par with global-based APES, which ignores some edge points and leverages a bit more on other non-edge points. Hence, an intriguing conclusion was drawn by APES:

> Overall, sampling more edge points improves the performance of downstream tasks. However, this can be overdone, and selecting only edge points can be detrimental.

Overall, APES aims to capture detailed edge information but falls short in maintaining good global uniformity of the input shapes. This deficiency leads to challenges in performing interpolation operations during the upsampling process and results in subpar sampling quality for the few-point sampling

task when pre-processing is not performed, as demonstrated in Figure 5.1. To address these limitations, in this chapter, we propose a novel method called SAMPS. SAMPS aims to strike a better balance between sampling edge points and preserving global uniformity by optimizing the sampling strategy. Our proposed approach overcomes the limitations of APES and showcases improved performance on downstream tasks, demonstrating its effectiveness in achieving a more favorable trade-off between capturing edge details and preserving overall shape uniformity.

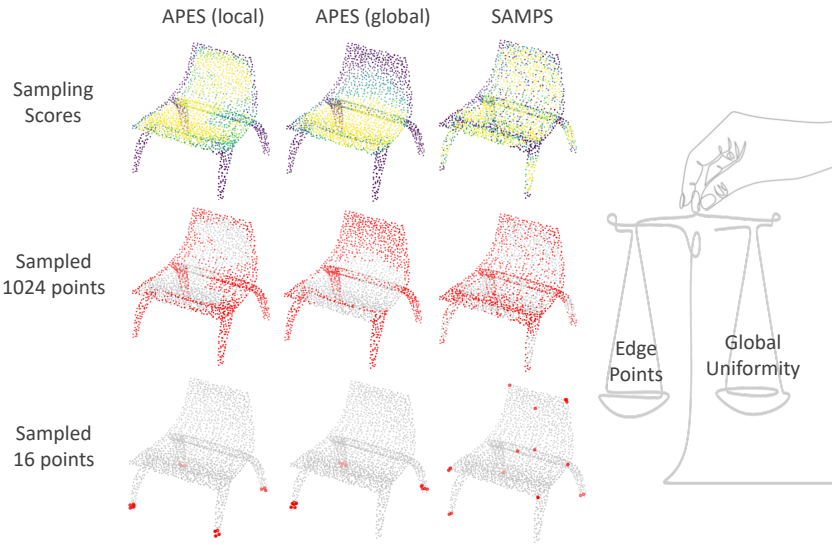


Figure 5.1: Sampling results with SAMPS in comparison with its predecessor. Our method achieves a better trade-off between sampling edge points and preserving global uniformity.

Point Categories Exploration

APES introduces two distinct methods for computing sampling scores for points, where higher scores indicate the presence of edge points. However, its design exhibits certain limitations, particularly in terms of emphasizing sharp

areas with strong focus and displaying a strong bias towards these regions. Consequently, when applied to few-point sampling, APES yields suboptimal results, as demonstrated in Figure 5.1. To overcome these challenges and improve the performance, it is crucial to develop a more effective approach for computing point-wise sampling scores. This enhanced method should aim to better distinguish between different point categories and accurately identify edge points. By addressing these shortcomings, we can devise a more robust and reliable framework for point-wise sampling.

During our exploration, we have observed that if $\mathbf{p}_i$ is among the k-nearest neighbors of $\mathbf{p}_j$, this does not necessarily imply that $\mathbf{p}_j$ is among the k-nearest neighbors of $\mathbf{p}_i$. To clarify this concept, we provide an intuitive example shown in Figure 5.2.

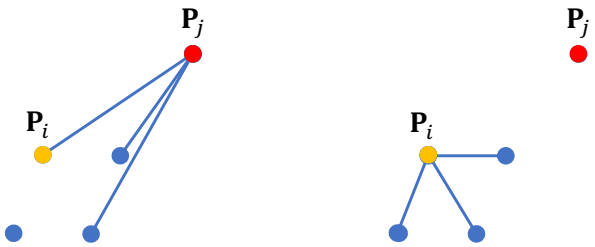


Figure 5.2: A simplified case of $k = 3$. While $\mathbf{p}_i$ is the k-nearest neighbor of $\mathbf{p}_j$, $\mathbf{p}_j$ is not the k-nearest neighbor of $\mathbf{p}_i$.

Therefore, it is evident that *the frequency of each point being selected as a neighbor* exhibits variation across a single point cloud. Taking a broader perspective, we have also observed that this particular parameter plays a pivotal role in accurately identifying edge points. Its significance lies in providing critical insights into the connectivity and proximity of each point to others within the data. By analyzing the frequency of point selection as a neighbor, we gain valuable information about the point's prominence and its potential involvement in forming edges.

An example is provided in Figure 5.3 to illustrate the concept. Let's consider the input point cloud as a simple grid structure. In this example, we select 5 neighbors for each point, resulting in all three possible cases shown on the left side of the figure. It is important to note that the center point is considered as a neighbor to itself. Notably, in the triangular and rectangular cases, there exist twin point pairs that share the possibility of being selected as neighbors. Such a point pair may be regarded as "quantum entangled", when one point is selected, the other one will not.

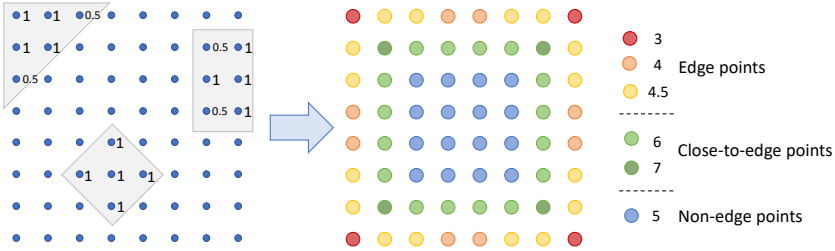


Figure 5.3: When selecting an equal number of neighbors for each point in the input point cloud, points at different positions are chosen as neighbors with varying frequencies.

Despite an equal number of neighbors being selected for each point in the input point cloud, points in different positions are chosen as neighbors with varying frequencies. The frequency of each point being selected as a neighbor is depicted on the right side of Figure 5.3. We can observe that in addition to the edge point and non-edge point categories, there is also another noteworthy point category of close-to-edge points. Moreover, within each category, the points can be further grouped into more sub-categories. The superior performance of global-based APES over local-based APES in downstream tasks can be attributed to its enhanced equilibrium across various point categories. Therefore, in this chapter, we propose a method to further explore a better sampling strategy given this phenomenon. To be more specific, *we take the frequency of each point being selected as a neighbor into consideration, achieving a better trade-off between sampling edge points and preserving global uniformity of the input point cloud for downstream tasks.*

Integrating Local and Global Information

Following the inception of the initial PointNet [Qi17a] method, which solely relied on global information, subsequent research within this domain has increasingly focused on integrating local information into learning frameworks and exploring methodologies for effectively combining both local and global information. Taking the most famous two of PointNet++ [Qi17b] and DGCNN [Wan19] as examples, they both learn local features either through local patches or in a point-centered manner, and progress the features iteratively for deriving a max-pooled global feature in a later layer. Subsequently, numerous methods have emerged aiming to directly integrate both local and global information within the framework [Hou22, Tho19, Li18b, Lu22b, Che21a]. However, these approaches primarily combine two features processed independently within separate modules in one way or another, e.g., simple concatenation [Wan19, Che21a, Lu22b], or slightly more advanced ways like cross-attention [Eng21].

In this chapter, we propose Sparse Attention Map (SAM) as a novel approach to integrating both local and global information directly within the attention map for point cloud sampling. The proposed SAM allows us to partition the points within a point cloud with the point-wise sampling scores being derived from it. Furthermore, It takes the aforementioned frequency of each point being selected as a neighbor into consideration to enhance the discrimination of various point categories during its construction process, thus contributing to better sampling outcomes. More details are given in Section 5.2. To the best of our knowledge, we are pioneering the fusion of local and global information at the attention map level for enhancing point cloud learning.

Once the SAM has been constructed, the next crucial step in the process is to design a method for computing point-wise sampling scores. Local-based APES computes row-wise standard deviations on local attention maps as point-wise sampling scores, while global-based APES computes column-wise sums on the global attention map as point-wise sampling scores. Although the rules for computing point-wise sampling scores are different in local and global-based APES, the key point here is that the scores are computed based on row-wise information or column-wise information. For example, in global-based APES, it is also possible to compute row-wise standard deviations as point-wise

sampling scores. Based on APES, and taking aforementioned the frequency of each point being selected as a neighbor into consideration, we explore multiple ways of computing the point-wise sampling scores either in rows or in columns in Section 5.3.

Alternative Sampling Policy

APES utilizes a straightforward top-M sampling policy based on the computed point-wise sampling scores. Building upon this, SAMPS introduces an alternative probability-based random sampling approach to enhance the trade-off balancing. Inspired by Boltzmann sampling [Bol68], SAMPS first maps the point-wise sampling scores into sampling probabilities to prepare for the sampling process. By introducing a temperature parameter to the mapping function, the sampling process can be fine-tuned from uniform sampling to top-M sampling. This temperature parameter plays a crucial role in achieving a desirable trade-off balance.

Boltzmann sampling is a probabilistic method used extensively in statistical mechanics and computational algorithms to generate samples from a given distribution, particularly the Boltzmann distribution [1]. This distribution describes the probabilities of a system's states as a function of their energy and temperature, capturing the likelihood of each state in thermal equilibrium. The probability p_i of a system being in state i with energy ε_i is given by the Boltzmann factor:

$$p_i = \frac{\exp\left(-\frac{\varepsilon_i}{k_B T}\right)}{\sum_{j=1}^{N} \exp\left(-\frac{\varepsilon_j}{k_B T}\right)}, \tag{5.1}$$

where k_B is the Boltzmann constant and T is the absolute temperature. The Denominator Equation (5.1) in sums over all possible states j. Boltzmann sampling leverages this distribution to sample states according to their thermal probability, making it a powerful tool in fields such as molecular dynamics,

[1] https://en.wikipedia.org/wiki/Boltzmann_distribution

where it helps simulate the behavior of particles at equilibrium. By appropriately adjusting T and calculating the energies ϵ_i, researchers can explore the system's configuration space and estimate properties like free energy and entropy.

Boltzmann sampling finds applications in various domains. In machine learning, it is employed in generative models like Restricted Boltzmann Machines (RBMs) and Deep Boltzmann Machines (DBMs) to generate synthetic data samples. In optimization, it aids in finding states with the lowest energy or cost. Additionally, Boltzmann sampling is used in simulating physical systems, exploring complex networks, and solving combinatorial problems.

In our case, we employ the Boltzmann sampling method to transform the point-wise sampling scores into sampling probabilities as a preparatory step for the sampling process. In this context, the sampling scores of points can be interpreted as the energies of the states. To assign higher sampling probabilities to points with larger sampling scores, we remove the minus sign. In this case, it is crucial to address the issue of exponential explosion that may arise. To avoid this problem, the incorporation of an additional normalization step becomes essential. For more detailed information, please refer to Section 5.4.

5.2 Sparse Attention Map

Both local and global attention maps are widely used in point cloud analysis. A global attention map is derived from the application of classical self-attention to point features of all points, while a local attention map concentrates on a point-centered area wherein cross-attention is specifically applied to the central point and its neighbors.

Denote $\mathcal{S}_i$ as the set of k-nearest neighbors of point $\mathbf{p}_i$, the local attention map for point $\mathbf{p}_i$ is defined as

$$\mathbf{m}_i^l = \mathrm{softmax}\left(Q(\mathbf{p}_i)K(\mathbf{p}_{ij} - \mathbf{p}_i)_{j\in\mathcal{S}_i}^{\top}/\sqrt{d}\right), \tag{5.2}$$

where Q and K stand for the linear layers applied on the query and key input, and the square root of the feature dimension count $\sqrt{d}$ serves as a scaling factor [Vas17].

For the global attention map which is equivalent to taking all points as the neighbors for each point, it is defined as

$$\mathbf{M}^g = \mathrm{softmax}\left(Q(\mathbf{p}_i)K(\mathbf{p}_j)^\top_{i,j\in S}/\sqrt{d}\right), \tag{5.3}$$

where S denotes the set of all input points.

In this chapter, we introduce a novel approach called the Sparse Attention Map (SAM), which effectively integrates local and global information at the attention map level. SAM offers two distinct variants based on its information basis. The first variant, known as Carve-based SAM, initiates with global information and subsequently incorporates local information into it. Conversely, the second variant, Insert-based SAM, begins with local information and contextualizes it within a global scenario. These two variants provide comprehensive strategies for leveraging both local and global cues in the attention map, enabling a more robust representation of the underlying data.

5.2.1 Carve-Based Sparse Attention Map

Instead of using local or global attention maps, SAM is proposed to compute point-wise sampling scores by integrating the knowledge from both local and global information. The idea is illustrated in Figure 5.4. After obtaining the global attention map with Equation (5.3), local kNN is used to find k neighbors for each point. In this case, k cells are being selected in each row. However, please notice that if a point $\mathbf{p}_j$ is a neighbor to point $\mathbf{p}_i$, it does not mean point $\mathbf{p}_i$ is always also a neighbor to point $\mathbf{p}_j$. This means while for each row k cells are selected, for each column, the number of selected cells is not fixed. The selected cells are then "carved out"to form the new sparse attention map, with the values of other non-selected cells being set to 0. We term it Carve-based Sparse Attention Map.

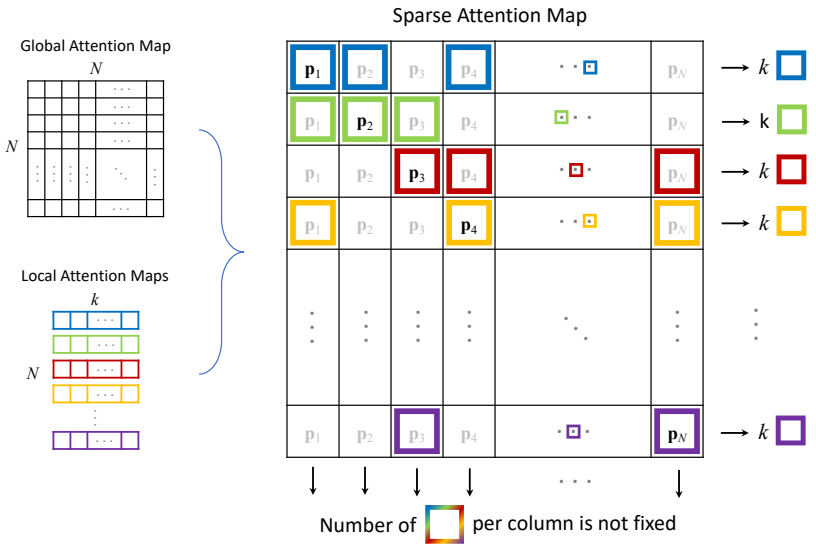


Figure 5.4: Sparse attention map. In each row, k cells are selected based on the kNN neighbor indexes for the corresponding point. The values of selected cells remain unchanged and other non-selected cells are all set to 0. While the number of cells selected within each row is k, the number of cells selected within each column is variable.

5.2.2 Insert-Based Sparse Attention Map

Carved-based sparse attention map starts from the global information, and then merges the local information. It can also be done in a reverse way: starting with the local information first, then considering it in a global situation. To be more specific, local-based attention maps are first computed with Equation (5.2), then the values in the local attention map of each point are inserted in the corresponding cells of each row in an empty (initialized as all 0s) global $N \times N$ attention map based on the kNN indexes. We term it Insert-based Sparse Attention Map. Again, for each row, k cells are inserted; and for each column, the number of inserted cells is not fixed.

Relation between Carve and Insert-based SAM

More vividly, consider the global attention map as a grid stone slab of size $N \times N$. For carve-based SAM, values of all cells are pre-computed and hidden in the slab grid cells, and only the selected cells are carved out; for insert-based SAM, only values of certain cells are pre-computed in the mosaic tile strings (the local-based attention maps), and they are then inserted into the slab according to the corresponding kNN indexes, like inserting mosaic tiles into an empty grid slate. The final outputs from carve-based SAM and Insert-based SAM are quite similar since they have the same places of non-zero cells. For both methods, the number of selected cells in each row is always k, while the number of selected cells in each column is variable. Their main difference is that the row-wise sum in insert-based SAM is always 1, while in carve-based SAM is not.

5.3 Indexing Mode

When sampling points, the points are indexed based on the computed point-wise sampling scores. We call the method of computing point-wise sampling scores from the full/sparse attention map as Indexing Mode. With the original full attention map, following APES, there are two possible indexing modes: (i) row standard deviation; and (ii) column sum. For a global attention map $\mathbf{M}^g$ of size $N \times N$, denote m_{ij} as the value of ith row and jth column in $\mathbf{M}^g$. To avoid possible confusion, we use notation $\mathbf{p}_o$ to denote a point only in this section. These two indexing modes can be formulated as follows.

Indexing mode i (row standard deviation). The standard deviation of the values in each row is computed as the point-wise sampling scores

$$a_{\mathbf{p}_o} = f_{\text{std}}(\{m_{oj} | j = 1,2,\dots,N\}), \tag{5.4}$$

where f_{std} is the function that computes the standard deviation of a set of values.

Indexing mode ii (column sum). The sum of the values in each column is computed as the point-wise sampling scores

$$a_{\mathbf{p}_o} = \sum_{i=1}^{N} m_{io} \,. \tag{5.5}$$

With the proposed carve-based sparse attention map, there are many other possible indexing modes. As highlighted in Section 5.1, achieving an improved sampling trade-off between sampling edge points and preserving global uniformity requires careful consideration of the frequency of each point being selected as a neighbor. In the context of SAM, this frequency is directly related to the number of selected cells in each column, which plays a crucial role in achieving the desired balance. With this variable, we further consider the following indexing modes: (iii) sparse row standard deviation; (iv) sparse row sum; (v) sparse column sum; (vi) sparse column average; and (vii) sparse column square-divided. Details and respective formulas are given as follows. Again, for a sparse attention map $\mathbf{M}^s$ of size $N \times N$, denote m^s_{ij} as the value of ith row and jth column in $\mathbf{M}^s$. For point $\mathbf{p}_o$, we denote the set of indexes of the selected k cells (indexes of kNN neighbors) in oth row as S_o, and denote the number of selected cells in oth column as n_o.

Indexing mode iii (sparse row standard deviation). The standard deviation of the selected cells in each row is computed as the point-wise sampling scores

$$a_{\mathbf{p}_o} = f_{\text{std}}(\{m^s_{oj} | j \in S_o\}) \,. \tag{5.6}$$

Indexing mode iv (sparse row sum). The sum of each row (note values of non-selected cells are all 0, same below) is computed as the point-wise sampling scores

$$a_{\mathbf{p}_o} = \sum_{j=1}^{N} m^s_{oj} \,. \tag{5.7}$$

Indexing mode v (sparse column sum). The sum of each column is computed as the point-wise sampling scores

$$a_{\mathbf{p}_o} = \sum_{i=1}^{N} m_{io}^s \,. \tag{5.8}$$

Indexing mode vi (sparse column average). The sum of each column is divided by n_o, i.e. the average, is used as the point-wise sampling scores

$$a_{\mathbf{p}_o} = \sum_{i=1}^{N} m_{io}^s / n_o \,. \tag{5.9}$$

Indexing mode vii (sparse column square-divided). To further enhance the impact of n_o, the sum of each column is divided by squared n_o to be used as the point-wise sampling scores

$$a_{\mathbf{p}_o} = \sum_{i=1}^{N} m_{io}^s / n_o^2 \,. \tag{5.10}$$

Based on carve-based SAM, all the above seven indexing modes are compatible. On the other hand, when insert-based SAM is used, since the sparse row-wise sum is always 1, only indexing modes iii, v, vi, vii are compatible.

To investigate how each indexing mode works, we train a separate model for each indexing mode with all other settings consistent. The sampling score distributions are visualized as heatmaps in Figure 5.5 to provide additional insights. From it, we can observe that both row standard deviation-based modes (mode i and mode iii) focus rigorously on edge points, which is consistent with the conclusion of APES [Wu23b]. However, since they always select all the points of the thin/detailed parts, certain parts are sometimes ignored. On the other hand, mode ii and mode iv share a similar output. They do not focus that rigorously on edge points, yet leverage a bit more on other point categories with certain preferences. This is also consistent with APES.

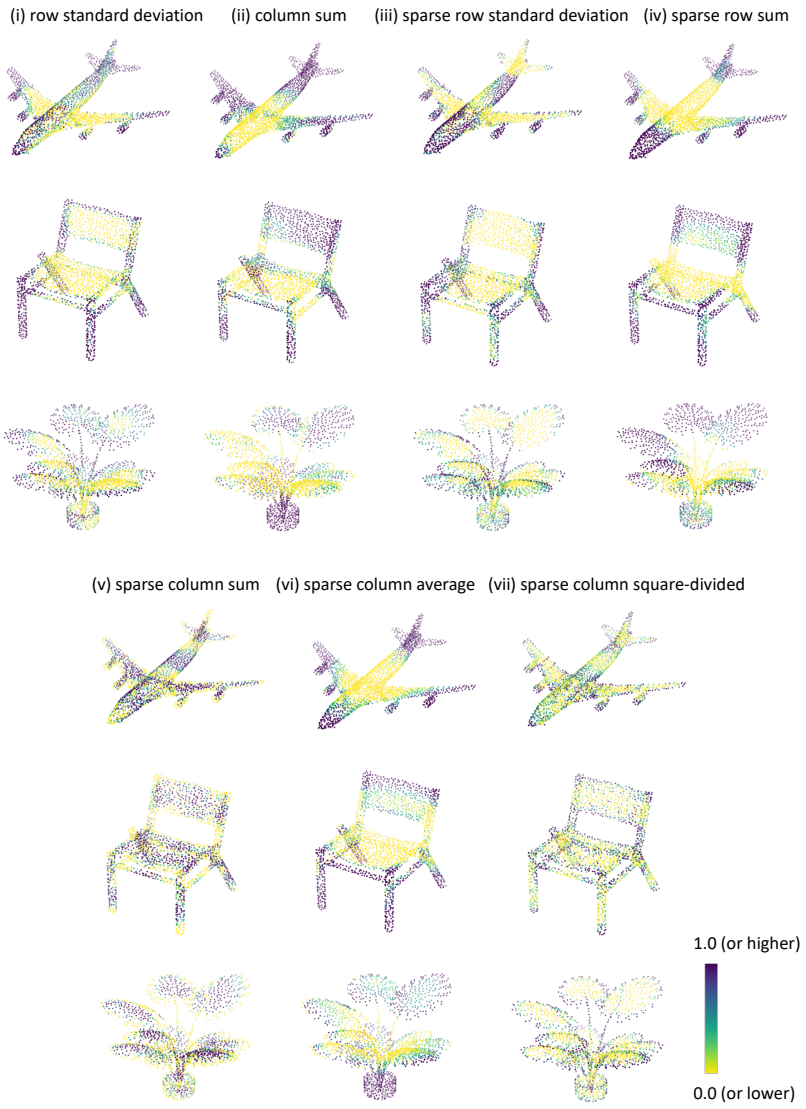


Figure 5.5: Heatmaps under different indexing modes with carve-based sparse attention map.

More interestingly, comparing the results of modes v, vi, and vii that use column-wise information from SAM, they pose different sampling preferences and strategies among different point categories, but the shape information is all more uniformly captured over the whole shape, with distinctive characteristics unique to each mode. Mode vi focuses more on the overall shape, while mode v focuses more on the non-edge points and mode vii focuses more on the edge points. This is due to edge points usually having a smaller number of n_o. In our case, we are interested in edge points yet we do not want it overdone. For example, chair leg points are points of detail, we want to sample some of them, yet we do not want to sample all of them; moreover, we want to sample some non-edge points as well to keep better global uniformity. Hence, we select mode vii as our main indexing mode and use it for most of the experiments in the following subsections.

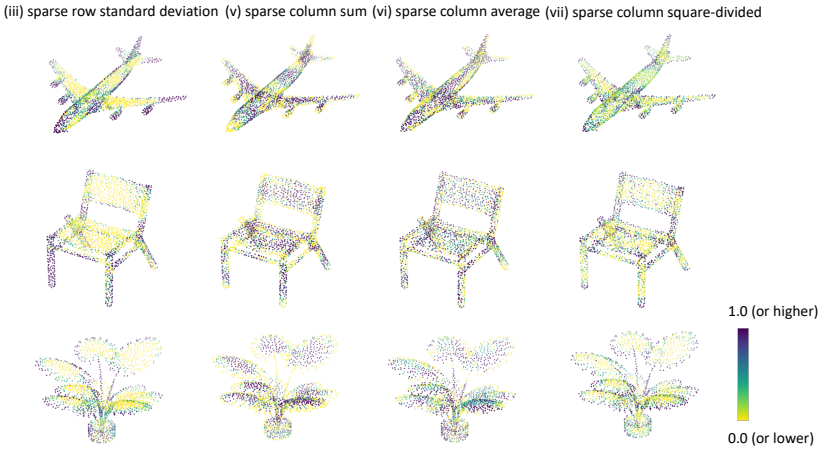


Figure 5.6: Heatmaps under different indexing modes with insert-based sparse attention map.

While Figure 5.5 illustrates the sampling score heatmaps for carve-based SAM under different indexing modes, Figure 5.6 gives the results for insert-based SAM. Note that only four indexing modes are applicable for insert-based SAM. The visualized results are quite similar to those of carve-based SAM except

for indexing mode vi, with which insert-based SAM shows smaller differences between point sampling scores. On the other hand, indexing mode vii still achieves a better trade-off between sampling edge points and preserving global uniformity. However, the performance of insert-based SAM on downstream tasks is mostly not on par with carve-based SAM (see Section 5.5.4), hence we use carve-based SAM as default for most experiments.

5.4 Tunable Random Sampling

After point-wise sampling scores are computed, points are sampled based on certain rules. In APES, points with larger scores are sampled, i.e. top-M. In our case, as we aim to enhance the neural network's flexibility by enabling the sampling of certain non-edge points, we suggest employing random sampling with priors. The idea is quite straightforward: process the point-wise sampling scores into point-wise sampling probabilities, and M non-repeated points are sampled randomly based on their sampling probabilities.

Score Normalization

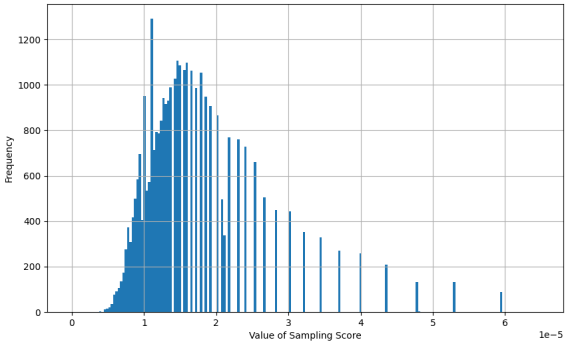


Figure 5.7: Histogram of sampling scores without normalization. Scores are of small values. As the sampling score becomes relatively large, its distribution exhibits increased discontinuity.

The sampling process is based on the computed sampling scores. When indexing mode vii is employed, an example of the distribution of sampling scores from it is provided in Figure 5.7. This figure highlights two main challenges for the subsequent process. Firstly, the distribution exhibits strong discontinuity, especially when the sampling score is large, indicating that the entire variation in value is dominanted by changes in n_o in Equation (5.10). Secondly, it is noteworthy that the sampling scores are confined to a narrow range of (0, 0.00006), which is comparatively small. After the softmax operation, these values tend to converge to almost identical numbers.

Therefore the sampling process should begin with the normalization of the point-wise sampling scores $a_{\mathbf{p}_i}$ across each shape. We employ the common Z-score normalization:

$$a'_{\mathbf{p}_i} = \frac{a_{\mathbf{p}_i} - f_{\text{mean}}(\{a_{\mathbf{p}_j} | j = 1,2,\dots,N\})}{f_{\text{std}}(\{a_{\mathbf{p}_j} | j = 1,2,\dots,N\})}, \tag{5.11}$$

where f_{mean} and f_{std} are the functions that compute the mean and the standard deviation of a set of values respectively.

Boltzmann Sampling

To process the normalized sampling scores a'_{p_i} into sampling probabilities ρ_{p_i}, the most simple way is Softmax, i.e.:

$$\rho_{\mathbf{p}_i} = \frac{e^{a'_{\mathbf{p}_i}}}{\sum_{j=1}^{N} e^{a'_{\mathbf{p}_j}}}. \tag{5.12}$$

Moreover, inspired by the Boltzmann Distribution [Bol68], we add a temperature parameter τ to Equation (5.12) to control the sampling distribution:

$$\rho_{\mathbf{p}_i} = \frac{e^{a'_{\mathbf{p}_i}/\tau}}{\sum_{j=1}^{N} e^{a'_{\mathbf{p}_j}/\tau}}. \tag{5.13}$$

As τ approaches 0, the sampling outcome converges towards top-M sampling. Conversely, as τ approaches infinity ($+\infty$), the result converges towards uniform sampling. Specifically, when $\tau = 1$, the sampling method aligns with Softmax-based sampling. This parameter allows seamlessly adjusting the sampling approach from uniform sampling to Softmax-based, and ultimately to top-M sampling. Sampling with τ can be seen as a compromise between the top-M sampling, Softmax-based sampling, and uniform sampling.

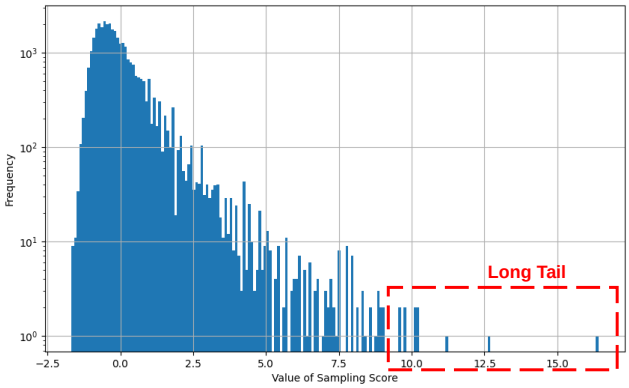


Figure 5.8: There are a small number of points in the long tail with high sampling scores. They can dominate the calculation of sampling probabilities

A typical distribution of sampling scores after normalization is depicted in Figure 5.8, showcasing a distribution with a pronounced long tail. The sampling probabilities are calculated according to eq. (5.13), and the sum of all probabilities equals 1. The long tail leads to a situation where the probabilities calculated are overwhelmingly influenced by a few points with high sampling scores. Consequently, the probabilities associated with the majority of points are nearly zero. To address this issue, a solution should be able to: (1) reduce the values of points in the long tail in the distribution, thereby diminishing their impact on the sampling probability of the remaining points, and (2) ensure minimal impact on the sampling scores of majority of points. The tanh function emerges as a fitting solution to these challenges, allowing us to

calculate the sampling probabilities by modifying Equation (5.13) as follows:

$$a''_{\mathbf{p}_i} = \tanh(a'_{\mathbf{p}_i}), \tag{5.14}$$

$$\rho_{\mathbf{p}_i} = \frac{e^{a''_{\mathbf{p}_i}/\tau}}{\sum_{j=1}^{N} e^{a''_{\mathbf{p}_j}/\tau}}. \tag{5.15}$$

5.5 Experimental Results

5.5.1 Classification

Network Architecture Design

For the classification task, we use the same network architecture of APES as in Figure 4.8 for a fair comparison. The downsampling layers have been replaced by the newly designed SAM-based ones. A new indexing mode and a new sampling policy have been adopted.

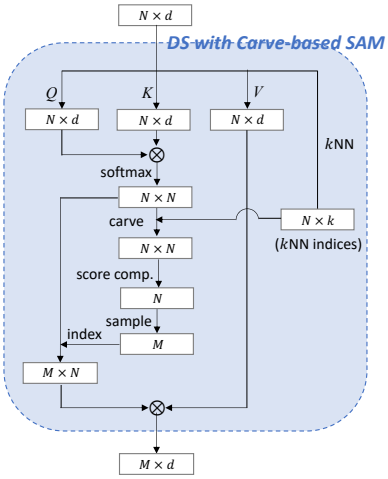


Figure 5.9: The network structure of SAMPS downsampling layers with carve-based SAM. The SAM is constructed by using the global attention map as the information basis.

The network structures of two alternative downsampling layers are illustrated in Figure 5.9 and Figure 5.10, respectively. In both approaches, a SAM is constructed after the computing attention map(s). The carve-based SAM utilizes the global attention map as the information basis, while the insert-based SAM employs local attention maps. In both cases, kNN indices of every point are computed from the input feature to guide the carve or the insert operation. Once a SAM is constructed, point-wise sampling scores are computed using one of the proposed indexing modes. Subsequently, these sampling scores are converted into sampling probabilities, which are then utilized for Boltzmann sampling. This conversion step prepares the probabilities for the subsequent sampling process, enabling the generation of samples based on the given distribution.

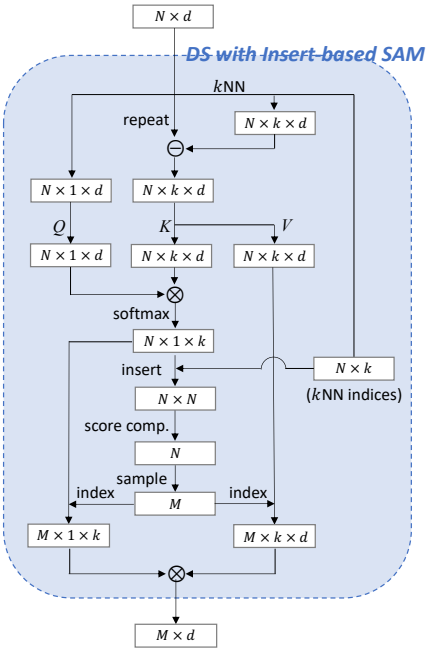


Figure 5.10: The network structure of SAMPS downsampling layers with insert-based SAM. The SAM is constructed by using the local attention maps as the information basis.

Training Detail

All the training settings are kept consistent with APES. AdamW is used as the optimizer. The learning rate starts from 1×10^{-4} and decays to 1×10^{-8} with a cosine annealing schedule. The weight decay hyperparameter for network weights is set as 1. Dropout with a probability of 0.5 is used in the last two fully connected layers. We use indexing mode vii for computing point-wise sampling scores and a temperature parameter of $\tau = 0.05$ is used during Boltzmann sampling. The network is trained with a batch size of 8 for 200 epochs.

Quantitative and Qualitative Results

The quantitative results of our evaluation are presented in Table 5.1, providing a comprehensive analysis of the performance of our method. In comparison to its predecessor, our proposed method surpasses previous methods and achieves state-of-the-art performance.

Table 5.1: Classification results on ModelNet40. In comparison with other SOTA methods that also only use raw point clouds as input.

Method	Overall Accuracy
PointNet [Qi17a]	89.2%
PointNet++ [Qi17a]	91.9%
SpiderCNN [Xu18]	92.4%
DGCNN [Wan19]	92.9%
PointCNN [Li18b]	92.2%
PointConv [Wu19]	92.5%
PVCNN [Liu19c]	92.4%
KPConv [Tho19]	92.9%
PointASNL [Yan20a]	93.2%
PT[1] [Eng21]	92.8%
PT[2] [Zha21c]	93.7%
PCT [Guo21]	93.2%
PRA-Net [Che21b]	93.7%
PAConv [Xu21]	93.6%
CurveNet [Muz20]	93.8%
DeltaConv [Wie22]	93.8%
PointNeXt [Qia22]	93.2%
APES (local) [Wu23b]	93.5%
APES (global) [Wu23b]	93.8%
SAMPS	**94.1%**

In Figure 5.11 and Figure 5.12, we present the qualitative results in a visually compelling manner, offering a deeper understanding of the capabilities of our proposed method compared to its predecessor. The visual comparison includes both sampling score heatmaps and sampled results, providing a comprehensive assessment of the performance and effectiveness of our approach.

One notable observation is that, unlike APES, our SAMPS algorithm demonstrates remarkable improvements in capturing global uniformity. By avoiding excessive focus on edge points, SAMPS achieves a more balanced and holistic representation of the underlying data. This is particularly evident in areas with intricate details, such as the delicate legs of a chair. Our method successfully achieves a better trade-off between sampling edge points and preserving the overall uniformity of the sampled results.

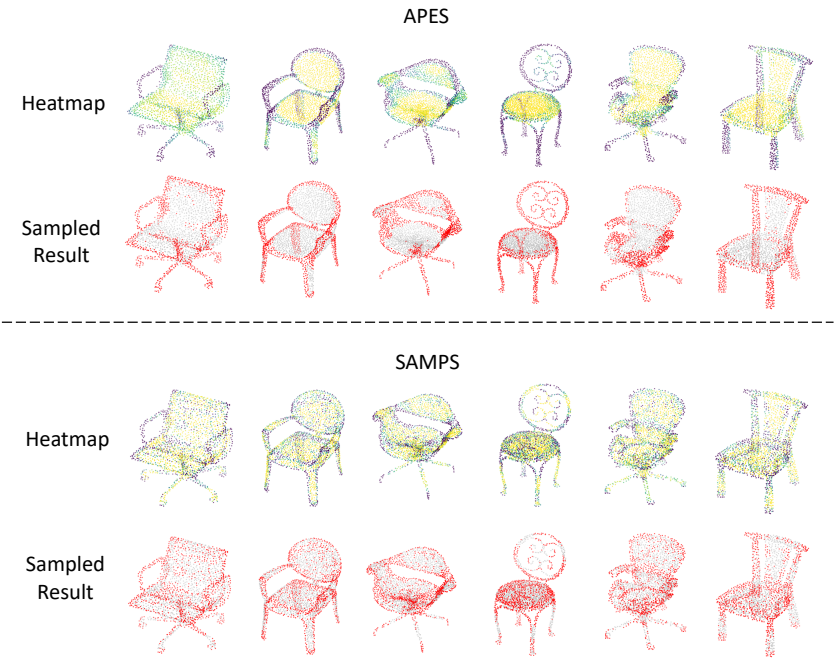


Figure 5.11: Qualitative results of our proposed SAMPS, in comparison with APES. Apart from the sampled results, sampling score heatmaps are also given. The chair category. All shapes are from the test set.

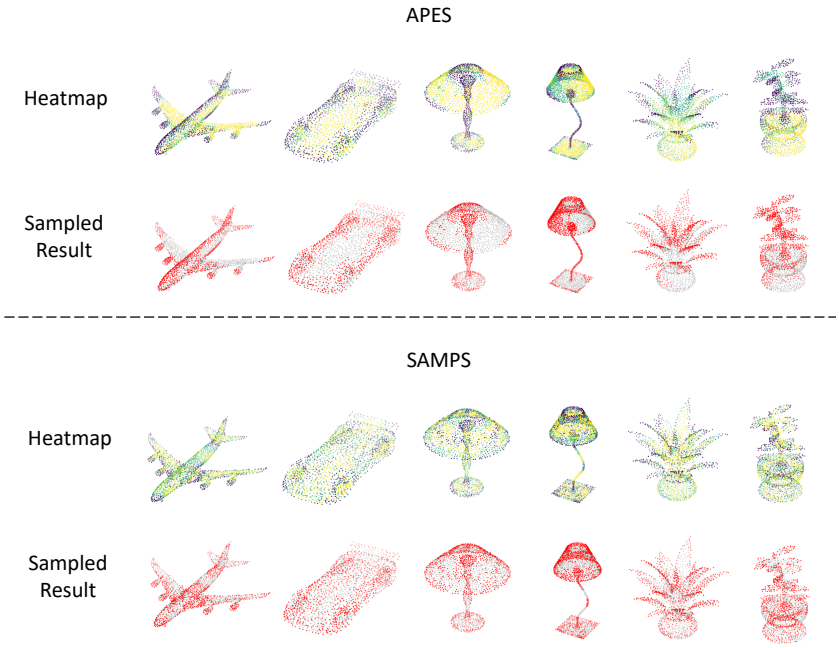


Figure 5.12: Qualitative results of our proposed SAMPS, in comparison with APES. Apart from the sampled results, sampling score heatmaps are also given. The airplane, car, lamp and plants categories. All shapes are from the test set.

These qualitative findings in Figure 5.11 and Figure 5.12 align with the quantitative results presented in Table 5.1, further solidifying the superiority of our proposed method. The combination of improved performance metrics and visually appealing results highlights the significant advancements offered by SAMPS, positioning it as a highly promising solution in the field of point cloud sampling.

Overall, our method not only outperforms its predecessor in terms of quantitative evaluation but also exhibits superior qualitative characteristics. These findings collectively demonstrate the efficacy and capability of our proposed approach, reinforcing its potential for various applications and its relevance to the broader research community.

5.5.2 Segmentation

Network Architecture Design

To ensure a fair comparison, we adopt the same segmentation network architecture as shown in Figure 4.14 for APES. However, we introduce our proposed new downsampling layer and replace the upsampling layer with an interpolation-based approach. The network structures of two alternative downsampling layers are illustrated in Figure 5.9 and Figure 5.10.

Most other point cloud network models apply neighbor-based interpolation [Qi17b, Li18b, Wu19, Qia22, Zha21c] for upsampling since FPS is used in the downsampling process. However, since APES focuses on edge points during the sampling, the neighbor-based interpolation operation is not feasible anymore, especially for those points that are far from edges. To circumvent this issue, APES proposes a cross-attention layer for upsampling and achieves reasonably satisfactory results. In this chapter, with the improved global uniformity achieved during sampling using our proposed downsampling layer, we can now leverage the benefits of interpolation-based upsampling. The network structure of the interpolation-based upsampling layer is depicted in Figure 5.13.

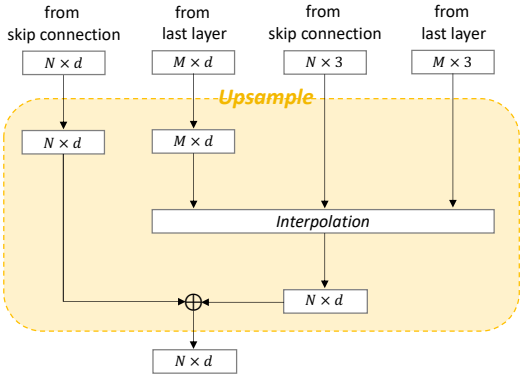


Figure 5.13: The network structure of the interpolation-based upsampling layer.

Training Details

All the training settings are kept consistent with APES. AdamW is used as the optimizer. The learning rate starts from 1×10^{-4} and decays to 1×10^{-8} with a cosine annealing schedule. The weight decay hyperparameter for network weights is 1×10^{-4}. Dropout with a probability of 0.5 is used in the last two fully connected layers. We use indexing mode vii for computing point-wise sampling scores and a temperature parameter of $\tau = 0.05$ is used during Boltzmann sampling. The network is trained with a batch size of 16 for 200 epochs.

Quantitative and Qualitative Results

The quantitative results are given in Table 5.2, providing a comprehensive analysis of the performance of our method.

Table 5.2: Performance of SAMPS on the ShapeNet Part Segmentation benchmark.

Method	Cat. mIoU	Ins. mIoU
PointNet [Qi17a]	80.4%	83.7%
PointNet++ [Qi17b]	81.9%	85.1%
SpiderCNN [Xu18]	82.4%	85.3%
DGCNN [Wan19]	82.3%	85.2%
SPLATNet [Su18]	83.7%	85.4%
PointCNN [Li18b]	84.6%	86.1%
PointConv [Wu19]	82.8%	85.7%
KPConv [Tho19]	85.0%	86.2%
PT[1] [Eng21]	-	85.9%
PT[2] [Zha21c]	83.7%	86.6%
PCT [Guo21]	-	86.4%
PRA-Net [Che21b]	83.7%	86.3%
PAConv [Xu21]	84.6%	86.1%
CurveNet [Muz20]	-	86.6%
DeltaConv [Wie22]	-	86.6%
StratifiedTransformer [Lai22]	**85.1%**	86.6%
PointNeXt [Qia22]	84.4%	**86.7%**
PointMLP [Ma22]	84.6%	86.1%
PointMetaBase [Lin23]	84.3%	**86.7%**
APES (local-based) [Wu23b]	83.1%	85.6%
APES (global-based) [Wu23b]	83.7%	85.8%
SAMPS	84.2%	86.5%

The quantitative results presented in Table 5.2 demonstrate that SAMPS surpasses its predecessor, showcasing improved performance across various metrics. However, SAMPS does not yet achieve the performance levels of the top-performing methods in the field. While SAMPS exhibits notable advancements and outperforms its predecessor, there is still room for further improvement to match or exceed the state-of-the-art results.

In addition, a study of applying different upsampling layers and interpolating with different K_{up} is also performed and the results are reported in Table 5.3. From it, we can observe a performance decrease in APES when interpolation is used for upsampling instead of cross-attention, while SAMPS achieves better performance with interpolation. On the other hand, as K_{up} increases, APES reports an overall performance increase, while a performance decrease is observed in SAMPS. This is due to that when a point cloud is uniformly sampled, too many neighbors do not always contribute to a better interpolation result. Hence many other papers also adopt $K_{up} = 3$ for interpolation. In conclusion, our method overcomes the limitation of APES and enables the application of interpolation for upsampling.

Table 5.3: Segmentation results with different upsampling layers on ShapeNet Part. The number before "/" is the category mIoU, and the number after is the instance mIoU.

Upsample	Interpolation			Cross Attention
	$K_{up} = 3$	$K_{up} = 8$	$K_{up} = 16$	
APES (local)	82.89 / 85.40	82.95 / 85.44	82.96 / 85.42	83.11 / 85.58
APES (global)	83.16 / 85.53	83.19 / 85.59	83.17 / 85.55	83.67 / 85.81
SAMPS	**84.22 / 86.46**	84.08 / 86.19	84.02 / 86.23	84.10 / 86.24

The qualitative results are depicted in Figure 5.14. Once again, SAMPS successfully strikes a better trade-off between sampling edge points and preserving global uniformity, as demonstrated across various object classes. For example, in the case of chair segmentation, SAMPS exhibits a notable improvement over APES by sampling more points on the chair seats. This increased sampling density enables effective interpolation operations during the subsequent upsampling stage. As a result, SAMPS captures finer details and contours, enhancing the overall quality of the segmentation output.

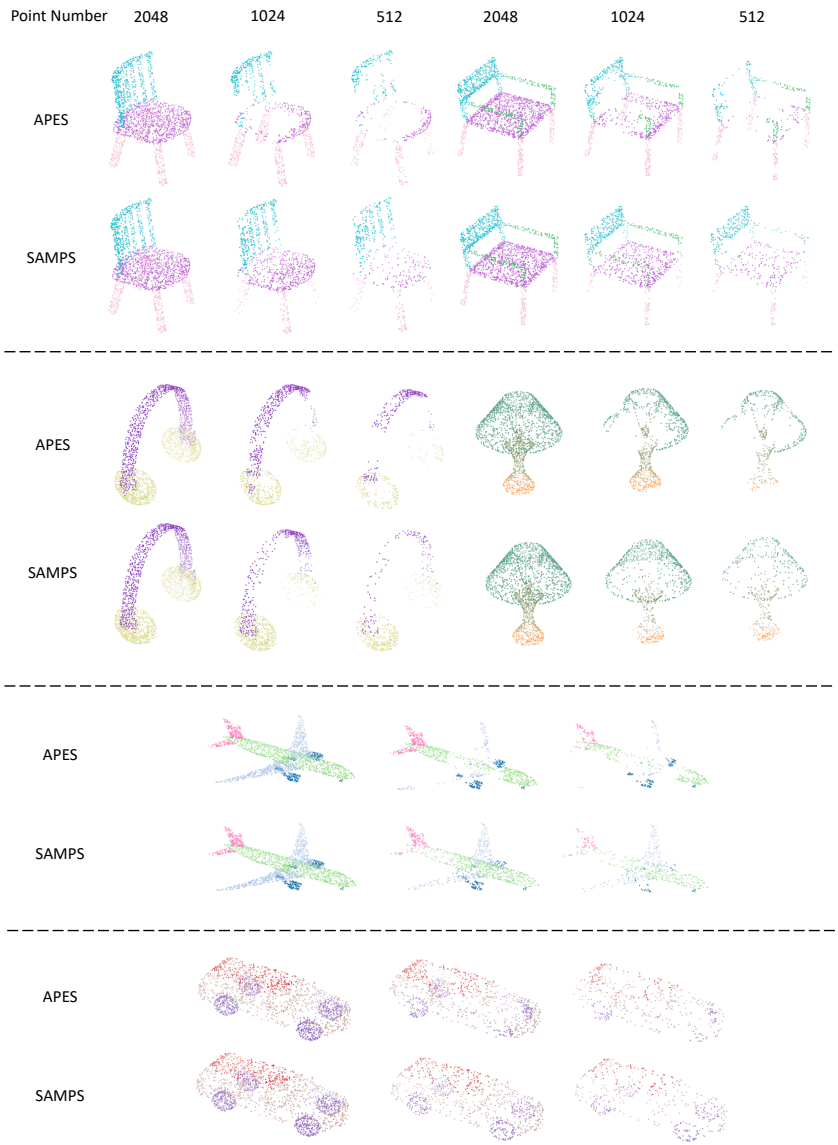


Figure 5.14: Segmentation results from SAMPS, in comparison with the result from APES. All shapes are from the test set.

5.5.3 Few Point Sampling

Experiment Setting

Apart from APES, we additionally compare our sampling method to previous work including RS, FPS, and the more recent learning-based S-Net, SampleNet, LighTN, etc. The same evaluation framework from [Dov19, Wan23, Wu23b] is used. The task here is the ModelNet40 Classification, and the task network is PointNet. Sampling methods are evaluated with multiple sampling sizes.

Moreover, it is important to note that APES addresses its limitations in few-point sampling by employing FPS to pre-sample the input into $2\boldsymbol{M}$ points [Wu23b]. However, this pre-processing step introduces an additional bias and complexity that may not be present in other methods, potentially affecting the fairness of the comparison. To ensure a fair and unbiased evaluation, we conducted experiments specifically on APES without the FPS pre-processing step. By removing this additional step, we can assess the performance of APES under more comparable conditions, aligning it with the evaluation setup of other methods.

Quantitative and Qualitative Results

We present the numerical results in Table 5.4, which clearly demonstrate that our SAMPS method achieves state-of-the-art performance in the few-point sampling task. Notably, when only a small number of points (less than approximately 100) are sampled from the input, SAMPS outperforms APES even when APES utilizes the pre-processing operation.

Due to the aforementioned limitations of APES, for few-point sampling, it uses FPS to pre-sample the input into $2\boldsymbol{M}$ points [Wu23b]. Notably, without this pre-processing step, the performance of APES experiences a significant decrease, as demonstrated in Table 5.4. In contrast, our SAMPS method preserves better global uniformity, allowing for satisfactory sampled results even when directly sampling a few points from the input. This distinction is evident in Figure 5.15, where a visual comparison between APES and SAMPS is presented.

Table 5.4: Comparison with other sampling methods. Evaluated on the ModelNet40 classification benchmark with multiple sampling sizes. For APES, we additionally report its performance when pre-processing is not performed.

M	Voxel	RS	FPS [Eld97]	S-NET [Dov19]
512	73.82	87.52	88.34	87.80
256	73.50	77.09	83.64	82.38
128	68.15	56.44	70.34	77.53
64	58.31	31.69	46.42	70.45
32	20.02	16.35	26.58	60.70
M	PST-NET [Wan21]	SampleNet [Lan20]	MOPS-Net [Qia20]	DA-Net [Lin21]
512	87.94	88.16	86.67	89.01
256	83.15	84.27	86.63	86.24
128	80.11	80.75	86.06	85.67
64	76.06	79.86	85.25	85.55
32	63.92	77.31	84.28	85.11
M	LighTN [Wan23]	APES (w/ pre-pro.)	APES (w/o pre-pro.)	SAMPS
512	89.91	**90.81**	89.81	90.46
256	88.21	**90.40**	86.78	90.12
128	86.26	89.77	84.87	**89.81**
64	86.51	89.57	79.23	**89.66**
32	86.18	88.56	75.63	**89.25**

From the visualization, it is apparent that when APES samples a very limited number of points directly from the input, it tends to concentrate those samples in the most salient or sharp locations. In contrast, SAMPS maintains better global uniformity in its sampling, ensuring a more balanced representation of the input data. The ability of SAMPS to preserve global uniformity during few-point sampling is a significant advantage. It enables the method to capture a more comprehensive understanding of the input, even with a limited number of samples.

The comparison between APES and SAMPS in terms of few-point sampling highlights the superior performance and global uniformity achieved by our proposed method. These findings reinforce the efficacy and versatility of SAMPS across various sampling scenarios and emphasize its potential for practical applications in situations involving limited resources or sparse data distributions.

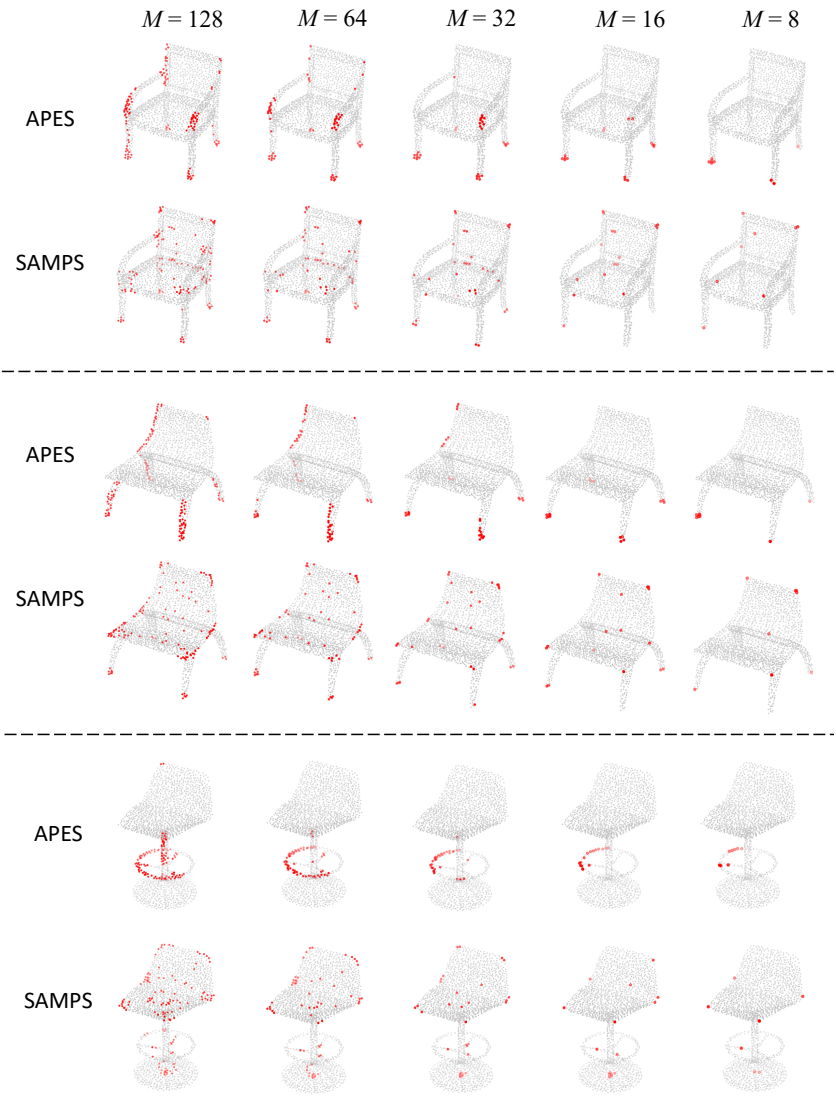


Figure 5.15: Sampled results of few-point sampling. No pre-FPS into $2M$ points was performed.

5.5.4 Ablation Study

Since the comparison with APES is of more importance, in this ablation study subsection, our emphasis is directed towards the novel designs introduced within this paper. Common topics that have been previously investigated in APES are briefly explored in this subsection.

Feature Dimension and Choice of k

In our experimental setup, we adopt a default embedding dimension of 128 and a neighbor number of 32 for SAMPS, consistent with the configuration used in APES. Furthermore, we conduct additional experiments to explore the impact of varying the embedding dimension d and the neighbor number k on SAMPS' performance. The numerical results tested on the ModelNet40 classification benchmark are presented in Table 5.5.

Table 5.5: Model performance on the ModelNet40 classification benchmark under different layer parameters, including embedding dimension d and neighbor number k.

d	64	128	192
OA (%)	93.68	94.05	94.13

k	8	16	32	64	128
OA (%)	93.54	93.88	94.05	94.12	94.16

From the results, it is evident that increasing the embedding dimension d or the neighbor number k generally leads to improved performance. This observation aligns with the notion that higher-dimensional embeddings and a larger number of neighbors provide richer and more comprehensive information for the sampling process. However, it is important to note that the performance gains become progressively marginal as these parameters increase. This diminishing improvement suggests that there may be diminishing returns beyond a certain point. Therefore, careful consideration should be given to strike a balance between performance and computational complexity when selecting the optimal values for d and k.

Different Indexing Modes

In Section 5.5, we use indexing mode vii as the method for computing point-wise sampling scores in the SAMPS framework. This indexing mode, combined

with the carve-based SAM, offers promising results in our experiments. To provide a comprehensive analysis, in addition to the visualized heatmaps in Figure 5.5 and Figure 5.6, the corresponding experimental results are presented in Table 5.6. In all experiments, we maintain a consistent temperature value of $\tau = 0.05$.

Table 5.6: Classification and segmentation performance of different indexing modes with different SAMs.

SAM	Indexing Mode	Cls. OA (%)	Seg. Cat. mIoU (%)	Seg. Ins. mIoU (%)
Carve	i	93.92	83.98	86.16
	ii	93.78	83.85	85.99
	iii	93.63	83.62	85.74
	iv	93.66	83.51	85.60
	v	93.40	83.47	85.49
	vi	**94.11**	84.12	86.38
	vii	94.08	**84.22**	**86.46**
Insert	iii	93.67	83.71	85.86
	v	93.44	83.42	85.51
	vi	93.46	83.64	85.78
	vii	**93.83**	**84.09**	**86.15**

The visualized heatmaps in Figures 5.5 and 5.6 provide a visual representation of the sampling scores generated by the respective indexing modes. On the other hand, the numerical results in Table 5.6 confirm that indexing modes vi and vii, when used in conjunction with the carve-based SAM method, consistently demonstrate superior performance compared to other indexing modes. These findings validate the effectiveness of these indexing modes in achieving the desired sampling outcomes.

Temperature Parameter

In addition to the aforementioned analyses, we also conducted an ablation study to investigate the effect of different values of τ in SAMPS. The results are reported in Table 5.7, and to complement these findings, we visualized

the sampling score heatmap, sampled result, and the post-softmax sampling probability heatmap in Figure 5.16.

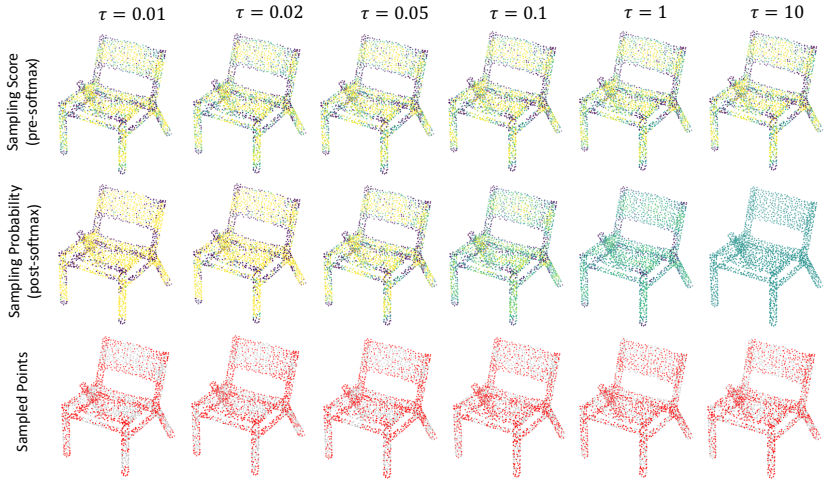


Figure 5.16: Different sampling results using different τ in the softmax with temperature during the sampling process. The indexing mode is the sparse column square-divided.

Table 5.7: Classification and segmentation performance of the model with different temperature τ value.

	τ	0.01	0.02	0.05	0.1	1	10
Cls.	OA (%)	93.74	93.90	**94.05**	93.98	93.72	93.41
Seg.	Cat. mIoU (%)	83.85	83.94	**84.22**	84.18	83.63	83.27
	Ins. mIoU (%)	86.04	86.27	**86.46**	86.36	86.02	85.82

From the visualization, it is evident that as τ increases, the sampling probability of points transitions from having a significant deviation to being more uniformly distributed, as intended by our design. This observation aligns with our objective of achieving global uniformity in the sampling process. Regarding the numerical results, we find that $\tau = 0.05$ yields the best performance in our experiments. However, it is worth noting that a smaller τ, which corresponds to a sampling strategy closer to Top-M sampling, does

not consistently guarantee better performance. This observation is consistent with the conclusion that solely sampling edge points can have detrimental effects on the overall performance.

The ablation study on the value of τ provides valuable insights into the influence of temperature in the sampling process of SAMPS. The visualization of the sampling probability heatmap and the quantitative results help us understand the optimal range of τ for achieving the desired balance between global uniformity and capturing important details.

Model Complexity

Model complexity significantly influences a model's ability to learn from varied and complex data patterns. Generally, more complex models, which typically feature more layers or neurons, are capable of capturing subtler nuances within the data. However, this increased complexity can also heighten the risk of overfitting, where a model becomes overly attuned to noise and specific details in the training data that do not generalize well to new data.

To evaluate SAMPS' practicality, we provide an analysis of model complexity in Table 5.8. This includes details on model parameters and floating-point operations per second (FLOPs) for both the entire model and a single downsampling layer. In order to assess inference efficiency, experiments were carried out using a trained ModelNet40 classification model on a single NVIDIA GeForce RTX 3090. The tests were conducted with a batch size of 8, evaluating a total of 2468 shapes from the test set.

Table 5.8: For model Complexity, we report the number of model parameters and FLOPs for both the full model and one downsampling layer. The inference throughput (instances per second) is also reported.

Method	Params.		FLOPs		Throughput
	Full Model	One DS Layer	Full Model	One DS layer	(ins./sec.)
APES (local)	4.47M	49.15k	4.59G	1.09G	488
APES (global)	4.47M	49.15k	3.03G	0.05G	520
SAMPS (insert)	4.47M	49.15k	4.59G	1.09G	446
SAMPS (carve)	4.47M	49.15k	3.03G	0.05G	473

In Table 5.8, it is evident that SAMPS shares the same number of model parameters as APES, as both employ an attention-based module for downsampling. Despite potential variations in tensor sizes resulting from the use of self-attention or cross-attention mechanisms, the fundamental model parameters remain consistent, primarily residing in the layers responsible for computing the $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ tensors.

On a different note, downsampling layers featuring carve-based SAM exhibit significantly lower FLOPs compared to those utilizing insert-based SAM. This efficiency stems from carve-based SAM's utilization of global information as its basis, involving point-to-point self-attention. In contrast, insert-based SAM relies on local information, incorporating point-to-neighbor cross-attention, which typically leads to higher FLOPs.

Lastly, while SAMPS introduces additional operations, its inference throughput experiences a slight decrease as a result.

5.6 Summary

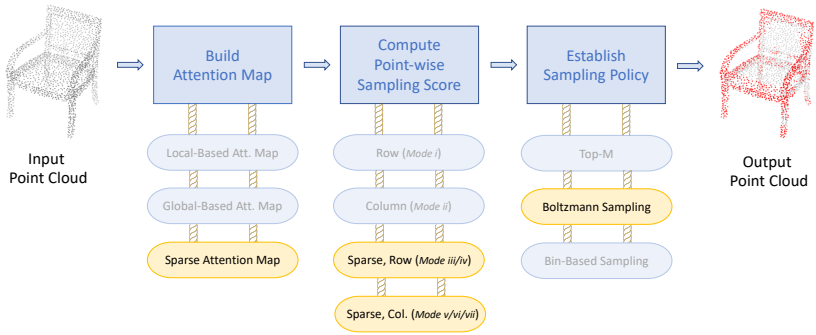


Figure 5.17: In this chapter, SAMPS is proposed by leveraging SAM and exploring various indexing modes. A new sampling policy of Boltzmann sampling is employed. It successfully achieves a better trade-off between sampling edge points and preserving global uniformity of the input point cloud.

In this chapter, a new point cloud sampling method called SAMPS has been proposed based on APES to overcome the limitations of the predecessor. Based on a sparse attention map that combines knowledge from both local and global information, multiple indexing modes have been designed and explored. Mapping the point-wise sampling scores into sampling probabilities, the softmax function with a temperature parameter is used to achieve simple tunable sampling. With the proposed methods, a better trade-off between sampling edge points and preserving global uniformity of the input point cloud has been achieved. This enables the utilization of interpolation-based upsampling layers and successfully leads to significantly improved sampling results, especially when only a few points are sampled.

However, the challenge of balancing edge sampling in point clouds could be further explored. One area that requires further exploration is the sampling policy. Regarding the sampling policy, both APES and SAMPS demonstrate a limitation in their approach to handling the inherent variations in point distributions observed across various shapes. Consequently, these methods do not adequately consider the unique characteristics exhibited by different shapes and instead employ the same sampling strategy applied uniformly to all point clouds. In the upcoming chapter, our objective is to develop an innovative sampling policy that can learn shape-specific sampling strategies for diverse point clouds.

6 SAMBLE: Learning Shape-Specific Sampling Strategies with Bin Partitioning

6.1 Going Beyond SAMPS

While current learning-based point cloud sampling methods are mostly generative-based [Dov19, Lan20, Che22], APES pioneers the direction of combining neural network-based learning and mathematical statistics-based direct point selection. This method identifies salient edge points in the point cloud outline by computing sampling scores that are derived from normalized attention maps. It demonstrates superior performance on multiple common benchmark tasks. Subsequently, we extend it to SAMPS, which introduces a Sparse Attention Map (SAM) to integrate information from both local and global, with multiple point-wise sampling score computation methods proposed and evaluated. SAMPS further achieves a better trade-off between sampling edge points and preserving the global uniformity of the input point cloud.

However, there is still room for improvement in both APES and SAMPS. These methods calculate sampling scores using mathematical statistics-based computations derived from the learned attention map(s). Subsequently, a sampling policy, such as top-M or Boltzmann sampling, is applied uniformly to all point clouds. In essence, APES and SAMPS manually establish the mapping between sampling scores and sampling probabilities. However, while the current mapping is competitive, it may not be optimal for all shapes. Considering the diverse features and distinct distribution of sampling scores

across different shapes, it is necessary to develop a shape-specific mapping between sampling scores and sampling probabilities.

To address these limitations, the objective of this chapter is to introduce a novel approach that enables the deep learning model to learn shape-specific sampling strategies. By doing so, we can overcome the drawbacks of the existing methods and achieve improved results by tailoring the sampling policies to the unique characteristics of each shape.

From Point Categories to Bin Partitioning

The idea originates from a thoughtful analysis of the illustrative demo figure that introduces point categories in SAMPS, as depicted once again in Figure 6.1. In it, various point categories are introduced, including Edge points, Close-to-edge points, and Non-edge points. Each category further contains multiple sub-categories.

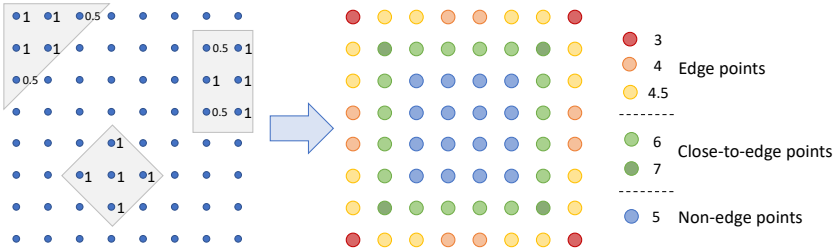


Figure 6.1: When selecting an equal number of neighbors for each point in the input point cloud, points at different positions are chosen as neighbors with varying frequencies.

Drawing inspiration from it, SAMPS achieves a better sampling trade-off between edge details and global uniformity. We further build upon that. Points within a specific category share similar sampling scores, suggesting that this partitioning can be efficiently achieved through bin partitioning based on the sampling score. Therefore, the mapping from sampling scores to sampling probabilities can be described by the bin partitioning process and the weight

associated with each bin. By leveraging this approach, shape-specific sampling strategies can be derived, leading to improved sampling outcomes.

Sampling pipeline

In this chapter, a new method of **S**parse **A**ttention **M**ap and **B**in-based **Le**arning method (termed SAMBLE) [Wu25] is proposed for point cloud sampling. A brief sampling pipeline is provided in Figure 6.2.

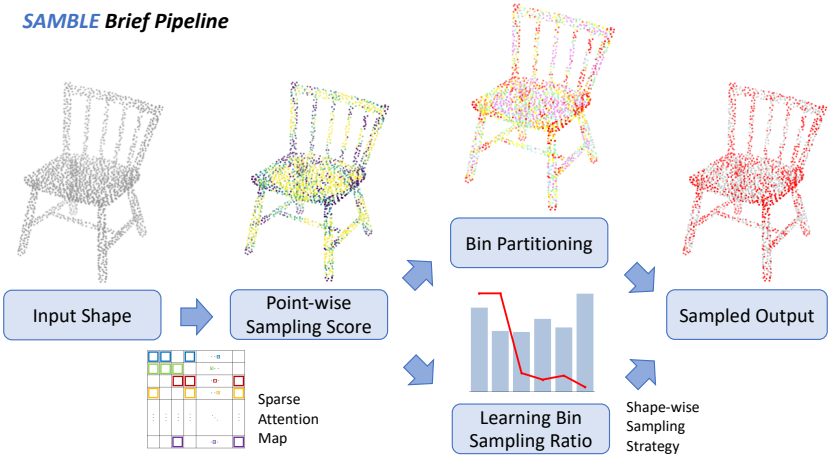


Figure 6.2: A brief pipeline of our proposed method SAMBLE to learn shape-specific sampling strategies for point cloud shapes.

This pipeline builds upon SAMPS by introducing a bin-based sampling method, which replaces the conventional sampling policy. In this method, the mapping from sampling scores to sampling probabilities is achieved through the bin partitioning process and the assignment of weights to each bin. It is important to note that when taking into account the actual network-based learning process, it is not as straightforward to define the boundaries for bin partitioning as illustrated in the demonstration shown in Figure 6.1. The boundaries for partitioning the points still need to be learned during the training process.

On the other hand, to learn the bin weights, inspired by ViT[Dos21], VilT[Kim21], and PointCAT [Yan23] — which leverage additional tokens during the computation of attention maps to extract and convey information across the entire feature map or specific groups of points or pixels — we introduce additional tokens specifically for learning bin sampling weights. In our case, attention maps are computed shape-wise during the downsampling process, facilitating the learning of bin sampling weights also in a shape-wise manner.

6.2 Bin-Based Sampling

After point-wise sampling scores are computed with SAM, points are sampled based on certain rules. The simplest way is to sample points with larger scores, i.e. top-M. In our case, as we aim to enhance the neural network's flexibility by integrating learned point cloud features into the sampling process and allowing for the selection of certain close-to-edge points or even non-edge points, we suggest employing a bin-based sampling strategy.

6.2.1 Bin Partitioning

Equal or Unequal Partitioning

To partition the points into bins, the simplest and most intuitional method is equal-sized bin partitioning. All N points are sorted by their downsampling scores from highest to lowest and then divided into n_b bins, with each bin containing an equal number of points.

However, the distribution of points among categories within different shapes may not necessarily be uniform. Equal-sized partitioning fails to account for shape-wise differences, potentially leading to points from different categories being grouped into the same bin or points from the same category being distributed across different bins. Therefore, we have adopted an unequal-sized partitioning approach that allows for an adaptive shape-wise distribution of points among the bins.

Instead of sorting points by sampling score and placing points with specific ranks into specific bins, they are partitioned into n_b bins according to the $n_b - 1$ boundary values, which are represented as a vector $\boldsymbol{v}_t = (v_1, v_2, \cdots, v_{n_b-1})$ in the tth iteration. These boundary values are meticulously designed and statistically computed to ensure that each bin contains approximately the same number of points across all point clouds from the training set. It is important to note that although $\boldsymbol{v}_t$ facilitates an approximate even division across all the shapes as the model is iteratively trained, for each individual shape, points are not evenly partitioned with the acquired boundary values at the final iteration.

Fixed or Dynamic Bin Boundary Values

To ensure an average distribution of points of all shapes among n_b bins, it's necessary to establish boundary values based on the collective statistics of all sampling scores of all points. In this case, using fixed boundary values for the bin partitioning becomes unfeasible since obtaining these values during runtime is not possible. This is because the model requires the boundary values from the very beginning, including the first training batch, and the model hasn't processed the complete dataset at that early stage. Therefore, relying on fixed boundary values obtained during runtime is impractical in this context.

The only possible way to use fixed boundary values is to pre-train the feature learning layers on a degraded model in which no bin partitioning is involved. Given the similarity of our overall model structure and attention score calculation method to SAMPS, it was expected that the attention scores obtained from it would have a distribution similar to that of SAMPS. Therefore, several SAMPS models were trained and saved, and the attention scores of all points in the training set within these models were statistically analyzed. The boundary values are obtained by equally partitioning the points of the entire dataset into n_b bins based on their sampling scores. These boundary values are fixed and used for the training of a SAMBLE model subsequently.

However, as research progressed and more improvements were introduced, the gap between the new and old sampling score distributions became increasingly significant. Using fixed boundary values in such cases resulted in instability during the training process and led to slightly worse performance. Relying

solely on SAMPS became less ideal. As a result, a new scheme with dynamic updates was adopted, allowing for adjustments and adaptations to the evolving sampling score distributions.

Dynamic Bin Boundary Values Updating

The process begins with the processing of the distribution of normalized point-wise sampling scores across the shapes within the current batch. Denoting n_b as the number of bins we used for partitioning, $n_b - 1$ boundary values are computed based on this distribution. In each training step, a vector $\boldsymbol{\nu}_c = (\nu_1, \nu_2, \cdots, \nu_{n_b-1})$ that ensures the equitable division of points among all shapes within the current batch is computed based on the point score distribution. Note that even while $\boldsymbol{\nu}_c$ enables an even division across the shapes of the current batch, for each individual shape, points are not evenly partitioned with the acquired batch-based boundary values.

During the training, for the first iteration, we directly use the boundary values derived from the first batch of data as the dynamic boundary values. Subsequently, since the second iteration, boundaries are updated adaptively in a momentum-based manner:

$$\boldsymbol{\nu}_t = \gamma \boldsymbol{\nu}_{t-1} + (1 - \gamma)\boldsymbol{\nu}_c \,, \tag{6.1}$$

where $\boldsymbol{\nu}_{t-1}$ stands for the bin partitioning boundaries used in the last iteration, and $\boldsymbol{\nu}_t$ is the updated dynamic boundaries used for the current iteration. $\boldsymbol{\nu}_c$ is a vector of boundary values that ensures the equitable division of points among all shapes within the current batch. $\gamma \in (0, 1)$ is the momentum update factor. With updated boundary values $\boldsymbol{\nu}_t$, points in each shape are divided into n_b subsets of $\{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{n_b}\}$ based on their sampling scores.

The principle idea presented here is the adaptive learning of boundary values, which are derived from the entirety of shapes within the training dataset. These values aim to evenly partition the distribution of point sampling scores across all shapes and points in the training data. Consequently, for each individual shape, the acquired boundary values can effectively partition its points into bins with a shape-specific strategy, capturing the unique characteristics of the shape while maintaining a degree of proximity to other shapes within the dataset.

6.2.2 Tokens for Learning Bin Weights

With points already being partitioned into bins for each shape, the next step is to learn a shape-specific sampling strategy, i.e., to learn shape-specific sampling weights in each bin for each shape. Inspired by ViT [Dos21], VilT [Kim21], and PointCAT [Yan23] — which leverage additional tokens during the computation of attention maps to extract and convey information across the entire feature map or specific groups of points or pixels — we introduce additional tokens specifically for learning bin sampling weights. In our case, attention maps are computed shape-wise during the downsampling process, facilitating the learning of bin sampling weights also in a shape-wise manner.

Using the former proposed bin partitioning method, points in each shape are partitioned into n_b subsets of $\{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_{n_b}\}$. The sampling weight ω_j for bin $\mathcal{B}_j (j = 1, 2, \dots, n_b)$ is established based on the distinctive features of each shape.

Network Structure for Bin Weights Learning

Figure 6.3 gives the network structure of our proposed downsampling layer and illustrates the idea of using additional tokens. n_b bin tokens are introduced during the attention computation, where each token corresponds to a specific bin. As shown in Figure 6.3, the bin tokens are initially concatenated with the input point-wise features for Key and $Value$. Subsequently, the combined features are subjected to a cross-attention mechanism with the original point-wise features as $Query$.

The attention map is split into two parts of a point-to-point sub-attention map and a point-to-token sub-attention map. For the point-to-point attention map, the methods proposed in Section 5.2 and Section 5.3 are applied to it to obtain point-wise sampling scores. Note that in this case, the row-wise sum is not exactly equal to 1 but still very close to 1 since n_b is of a very small quantity compared to N. With computed point scores, dynamic boundary values $\boldsymbol{v}_t$ are obtained for bin partitioning.

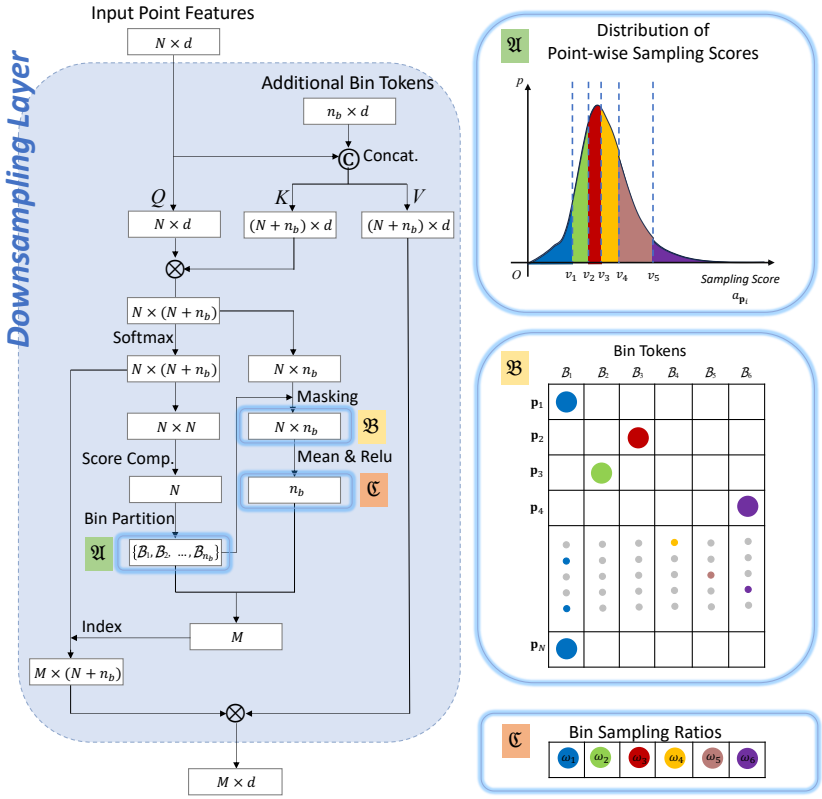


Figure 6.3: Left: network structure of downsampling layer. Block $\mathfrak{A}$: Points in each shape are partitioned into n_b bins. Block $\mathfrak{B}$: Masking the split-out point-to-token sub-attention map. Block $\mathfrak{C}$: Learned bin sampling weights.

The point-to-token sub-attention map is dimensioned at $n_b \times N$. Each column contains features not only from points in the corresponding bin but also from points in the other bins. However, the significance of each bin should be solely determined by the features of the points within that specific bin. To eliminate interference from points irrelevant to the given bin, within each column, the values associated with points from other bins are set to 0. In contrast, the values of points that belong to the corresponding bin are kept unchanged.

In short, a mask operation is performed on the point-to-token sub-attention map as illustrated in Block $\mathfrak{B}$ of Figure 6.3.

The sampling weights ω_j are then subsequently acquired with

$$\omega_j = \text{ReLU}\left(\frac{1}{\beta_j}\sum_{\mathbf{p}_i \in \mathcal{B}_j} m_{\mathbf{p}_i,\mathcal{B}_j}\right), \tag{6.2}$$

where β_j stands for the number of points in bin $\mathcal{B}_j$, and $m_{\mathbf{p}_i,\mathcal{B}_j}$ represents the element in the energy matrix corresponding to point $\mathbf{p}_i$ in row and $\mathcal{B}_j$ in column.

Adding Bin Tokens to Q or K/V?

A critical point in the idea of bin tokens lies in determining the specific branches to which the tokens should be concatenated. In order to match the tensor dimension for later computation in the attention mechanism, the tensor size of Key and Value should be the same. Hence if tokens are being added to the Key branch, they also have to be added to the Value branch. Overall, there are two possibilities of adding bin tokens to (i) the Query branch, or (ii) the Key and the Value branches.

It is crucial to emphasize that, due to the nature of the sampling operation where indexes are selected, gradients cannot be propagated back through the sampling operation during the backward propagation process, as discussed in Section 4.4.1. As a result, regardless of the selected structure, it is essential to establish an alternative pathway to convey the information contained within the bin tokens, which have a size of $n_b \times N$, to the downsampled features, which have a size of $M \times d$. This pathway should ensure the flow of relevant information despite the inability to directly backpropagate gradients through the sampling operation.

As illustrated in the left of Figure 6.4, in the former case, an attention map of tensor size $(N+n_b)\times N$ is obtained. After M indexes of the points to be sampled are learned with SAMBLE, M rows in the attention map are extracted to form

a new tensor for the next steps. However, note that the sub-tensor of $n_b \times N$ will never be delivered to the next steps since they do not correspond to points, hence no gradient will be backpropagated to the tokens during the training.

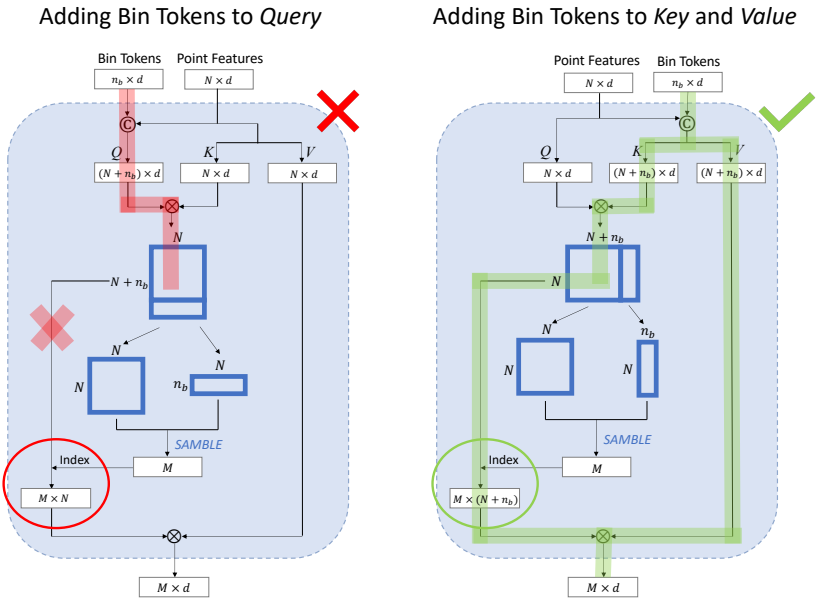


Figure 6.4: Adding bin tokens to Query leads to no gradient being backpropagated to the tokens, while adding bin tokens to Key and Value enables the gradient backpropagation.

On the other hand, as illustrated in the right of Figure 6.4, adding bin tokens to the Key and Value branches does not have this problem and successfully enables gradient backpropagation. One thing worth mentioning is that in this scenario, the row-wise sum is not exactly equal to 1 but still very close to 1 due to the significantly smaller magnitude of n_b relative to N. Therefore, this is unlikely to significantly impact the calculation of point-wise sampling scores. Concerning the design of adding bin tokens to all branches of Query, Key, and Value, it is equivalent to case ii since the sub-tensor of n_b rows in the attention map will never be sampled and propagated.

Pre-softmax or Post-softmax Attention Map for Splitting The Point-to-Token Sub-Attention Map

When addressing the bin tokens, our initial approach involved splitting the point-to-token sub-attention map from the post-softmax attention map $\mathbf{M}_{\text{post}}$, which seemed intuitively appropriate. Furthermore, all elements within $\mathbf{M}_{\text{post}}$ are inherently positive, eliminating any concern for negative sampling weights and obviating the need for an additional ReLU operation. However, experimental findings revealed that this method proved ineffective, as it resulted in overly uniform sampling weights across different bins.

The underlying cause of this issue was identified after we explored the underlying mathematical principles and examined the values in the tensors during runtime. Tensors in a well-trained network tend to exhibit diminutive feature values as they propagate through layers. Denote m_{ij} as one element in the pre-softmax attention map $\mathbf{M}_{\text{pre}}$, given its minute magnitude, we apply the Taylor expansion formula to yield:

$$e^{m_{ij}} = 1 + m_{ij} + \frac{m_{ij}^2}{2} + \cdots \approx 1 + m_{ij}\,. \tag{6.3}$$

Therefore, the corresponding element m'_{ij} in the post-softmax attention map is

$$m'_{ij} = \frac{e^{m_{ij}}}{\sum_{j=1}^{N+n_b} e^{m_{ij}}} \approx \frac{1 + m_{ij}}{N + n_b + \sum_{j=1}^{N+n_b} m_{ij}}\,. \tag{6.4}$$

In our case, the values of the elements m_{ij} in $\mathbf{M}_{\text{pre}}$ are approximately within the magnitude of 10^{-3} to 10^{-5}. After a softmax operation, the resultant values m'_{ij} in $\mathbf{M}_{\text{post}}$ exhibit minimal variation, leading to closely similar sampling weights across bins in a later step.

Efforts were undertaken to address this issue before we turned to using $\mathbf{M}_{\text{pre}}$ for sampling weights acquisition. We attempted to use the logarithmic operation

to restore the lost information:

$$\ln(m'_{ij}) = \ln\left(\frac{e^{m_{ij}}}{\sum_{j=1}^{N+n_b} e^{m_{ij}}}\right) = m_{ij} - \ln\left(\sum_{j=1}^{N+n_b} e^{m_{ij}}\right) \tag{6.5}$$

After the logarithmic operation, every value in the sub-attention map is negative. Therefore, a normalization operation is necessary. However, as shown in 6.5, the common normalization methods, such as z-score and centering, will result in too many negative elements (more than half), leading to too much information loss when passing through subsequent ReLU modules. Even if we successfully identify or meticulously design a superior normalization method that enables manual control over the proportion of negative elements to an applicable value, such manual intervention strays from the original intention of this thesis, which is to discover a learning-based mapping from sampling score to sampling probability.

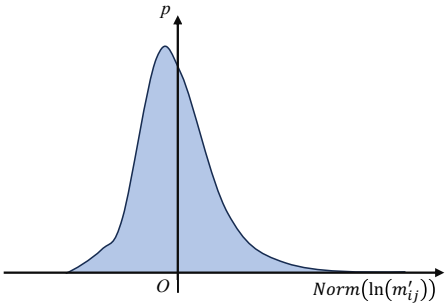


Figure 6.5: Illustrative figure of the distribution of the element values in the post-softmax attention map, after normalization.

Through the analysis, we observed that the term m_{ij} in Equation (6.5) is exactly the elements in the pre-softmax attention map and is what we are interested in. Therefore, to avoid the potential loss of information that could arise from the softmax operation, we opted to directly use the results from $\mathbf{M}_{\text{pre}}$ for bin sampling weights acquisition.

Order of Mean-pooling and ReLU Operations

Within our design, the ReLU operation is used to prevent the learned sampling weight from being negative. It can be performed after Mean-pooling, as shown in Equation (6.2), or performed before Mean-pooling:

$$\omega_j = \frac{1}{\beta_j} \sum_{\mathbf{p}_i \in \mathcal{B}_j} \mathrm{ReLU}(m_{\mathbf{p}_i, \mathcal{B}_j}) \,. \tag{6.6}$$

However, the inherent distribution of values within tensors often results in a non-negligible proportion being negative, especially those corresponding to points of lower importance. Directly setting too many values to zero would result in a significant loss of features, which is regrettable considering the potential information discarded.

Therefore, instead of performing the ReLU operation before the mean-pooling operation, we do it the other way around, i.e., first mean-pooling, then, after this information fusion, ReLU is performed over the pooled results. As detailed in Section 6.3.4, this mean-first approach has been adopted based on the results and analysis of our experiments.

6.2.3 Sampling in Each Bin

After completing the evaluation of the points' importance by calculating the sampling score, performing bin partitioning, and assessing the importance of each bin, points can then be downsampled. This process is carried out in two steps: (1) determining the number of points that need to be selected from each bin, and (2) selecting these points within each bin according to specific rules.

Determining Numbers of Sampled Points for Bins

For each shape, by considering the number of points contained within bins $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{n_b})$ alongside the determined bin sampling weights $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_{n_b})$, the specific numbers of points to be selected from each bin $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \ldots, \kappa_{n_b})$ are computed as follows.

Direct multiplication of $\boldsymbol{\beta}$ and $\boldsymbol{\omega}$ does not yield a sum that aligns with the total number of down-sampled points M required by the network structure. To address this discrepancy, a scaling method is applied to first scale bin sampling weights ω_j. Furthermore, to prevent κ_j from surpassing the available number β_j in any bin, any excess points are proportionately redistributed to other bins that have not been fully selected. The detailed pipeline is described in Algorithm 1. Here, $\circ$ denotes the Hadamard (element-wise) product, and ε is a small positive constant added to ensure that the denominator in line 5 remains strictly positive.

Algorithm 1 Determining $\boldsymbol{\kappa}$ from $\boldsymbol{\beta}$ and $\boldsymbol{\omega}$

Require: number of total points to be selected: M, Sampling weights $\boldsymbol{\omega} : [\omega_1, \omega_2, \ldots, \omega_{n_b}]$, number of points in bins $\boldsymbol{\beta} : [\beta_1, \beta_2, ..., \beta_{n_b}]$

1: $\boldsymbol{\kappa} \leftarrow \mathbf{0}$
2: $\mathbf{x} \leftarrow \boldsymbol{\omega} \circ \boldsymbol{\beta} + \varepsilon$
3: $M_r \leftarrow M$
4: **while** $M_r > 0$ **do**
5: $\quad s \leftarrow \frac{M_r}{\sum x_j}$
6: $\quad$ **for** $j = 1$ to n_b **do**
7: $\quad\quad \kappa_j \leftarrow \text{round}(\kappa_j + s x_j)$
8: $\quad\quad$ **if** $\kappa_j \geq \beta_j$ **then**
9: $\quad\quad\quad \kappa_j \leftarrow \beta_j$
10: $\quad\quad\quad x_j \leftarrow 0$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad M_r \leftarrow M - \sum \kappa_j$
14: **end while**
15: **return** $\boldsymbol{\kappa}$

Sampling Within Bins

Finally, within each bin $\mathcal{B}_j$, a total of κ_j points are sampled based on the normalized point-wise sampling scores $a''_{\mathbf{p}_i}$ computed from Equation (5.15). It is important to note that these scores are normalized within each bin other

than the whole point cloud. In this case, there are several options available for the in-bin sampling policy. Alongside the commonly used Top-M sampling and the previously explored fixed-temperature Boltzmann sampling, we also investigate the effectiveness of bin-wise adaptive-temperature Boltzmann sampling as an additional method.

(i) Top-M sampling. The Top-M sampling represents the most fundamental approach, wherein the points with the top-κ_j highest downsampling scores are selected from a total of β_j points in j-th bin $\mathcal{B}_j$. This method proves particularly effective for sampling salient points in the point cloud outline.

(ii) Uniform sampling. From a set of β_j points in bin $\mathcal{B}_j$, a total of κ_j are randomly selected without replacement. Within each bin, points have a uniform probability of being selected. This method excels at maintaining the overall shape of the point cloud.

(iii) Fixed-temperature Boltzmann sampling. From a set of β_j points in bin $\mathcal{B}_j$, a total of κ_j points are selected through random sampling with priors, i.e. with point-wise sampling probabilities. Within each bin, to process the sampling scores $a_{\mathbf{p}_i}$ into sampling probabilities $\rho_{\mathbf{p}_i}$, the computations described in Section 5.4 are executed:

$$a'_{\mathbf{p}_i} = \frac{a_{\mathbf{p}_i} - f_{\text{mean}}(\{a_{\mathbf{p}_i} | i = 1,2,\dots,\beta_j\})}{f_{\text{std}}(\{a_{\mathbf{p}_i} | i = 1,2,\dots,\beta_j\})}, \tag{6.7}$$

$$a''_{\mathbf{p}_i} = \tanh(a'_{\mathbf{p}_i}), \tag{6.8}$$

$$\rho_{\mathbf{p}_i} = \frac{e^{a''_{\mathbf{p}_i}/\tau}}{\sum_{\mathbf{p}_i \in \mathcal{B}_j} e^{a''_{\mathbf{p}_i}/\tau}}. \tag{6.9}$$

f_{mean} and f_{std} are the functions that compute the mean and the standard deviation of a set of values respectively. The tanh function is used to avoid the long-tail problem as discussed in Section 5.4. Drawing inspiration from the Boltzmann Distribution [Bol68], a temperature parameter τ is added to control the sampling probability distribution. The point $\mathbf{p}_i$ in bin $\mathcal{B}_j$ is randomly selected with the probability $\rho_{\mathbf{p}_i}$. Totally κ_j points are selected.

As τ approaches 0, the sampling outcome converges towards top-M sampling. Conversely, as τ approaches infinity ($+\infty$), the result converges towards uniform sampling. Specifically, when $\tau = 1$, the sampling method aligns with Softmax-based sampling. This parameter allows seamlessly adjusting the sampling approach from uniform sampling to Softmax-based, and ultimately to top-M sampling. Sampling with τ can be seen as a compromise between the top-M sampling, Softmax-based sampling and uniform sampling.

(iv) Adaptive-temperature Boltzmann sampling. The distributions of post-Boltzmann-softmax sampling probabilities depend on the total number of points β_j in each bin $\mathcal{B}_j$. Considering the variable number of points in bins caused by unequal partitioning, it is worth exploring the adaptation of the temperature parameter τ based on the number of points β_j in the bin $\mathcal{B}_j$ instead of using a fixed value. Based on this idea, the adaptive temperature τ_j for each bin $\mathcal{B}_j$ is determined as follows:

$$\tau_j = \frac{N/n_b}{\beta_j}\tau\,, \tag{6.10}$$

where N/n_b is the average number of points over all bins. This implies that when a bin contains a higher number of points, a relatively smaller value of τ_j is used, emphasizing the selection of salient points during the sampling process. Conversely, when a bin contains a lower number of points, a relatively larger τ_j is employed, promoting a more uniform sampling across the bin. It is worth noting that when $n_b = 1$ and thus $\beta_j = N$, Equation (6.10) essentially simplifies to the fixed-temperature Boltzmann sampling.

Subsequently, the sampling probability ρ_{p_i} of point $\mathbf{p}_i$ in each bin is determined as follows:

$$\rho_{\mathbf{p}_i} = \frac{e^{a''_{\mathbf{p}_i}/\tau_j}}{\sum_{\mathbf{p}_i \in \mathcal{B}_j} e^{a''_{\mathbf{p}_j}/\tau_j}}\,. \tag{6.11}$$

Experiments concerning the in-bin sampling policies outlined above, along with the hyperparameters, are detailed in Section 6.3.4. Drawing from the

findings of these experiments, sampling with a fixed temperature parameter τ is found to be the most effective and is applied in most of the following experiments.

6.3 Experimental Results

6.3.1 Classification

Training Details

For a fair comparison, we use the same network architecture in SAMPS but with our proposed new sampling layer. All the training settings are kept consistent. AdamW is used as the optimizer. The learning rate starts from 1×10^{-4} and decays to 1×10^{-8} with a cosine annealing schedule. The weight decay hyperparameter for network weights is set as 1. Dropout with a probability of 0.5 is used in the last two fully connected layers. We use $n_b = 6$ bins for point partitioning. The momentum update factor $\gamma = 0.99$ for updating boundary values. A fixed temperature parameter of $\tau = 0.1$ is used. The network is trained with a batch size of 8 for 200 epochs.

For bin tokens, each element x_b of bin tokens is so initialized, that:

$$x_b \sim \mathcal{N}\left(0, \frac{1}{\sqrt{d}}\right), \tag{6.12}$$

where d is the dimension of bin tokens. According to the design, d equals the dimension of the input features of the attention block and is 128 in our case.

Quantitative and Qualitative Results

The quantitative evaluation results are presented in Table 6.1, offering a comprehensive analysis of the performance of our method. Our proposed approach outperforms previous methods, including its predecessor, and attains state-of-the-art performance levels.

Table 6.1: Classification results on ModelNet40. In comparison with other SOTA methods that also only use raw point clouds as input.

Method	Overall Accuracy
PointNet [Qi17a]	89.2%
PointNet++ [Qi17a]	91.9%
SpiderCNN [Xu18]	92.4%
DGCNN [Wan19]	92.9%
PointCNN [Li18b]	92.2%
PointConv [Wu19]	92.5%
PVCNN [Liu19c]	92.4%
KPConv [Tho19]	92.9%
PointASNL [Yan20a]	93.2%
PT[1] [Eng21]	92.8%
PT[2] [Zha21c]	93.7%
PCT [Guo21]	93.2%
PRA-Net [Che21b]	93.7%
PAConv [Xu21]	93.6%
CurveNet [Muz20]	93.8%
DeltaConv [Wie22]	93.8%
PointNeXt [Qia22]	93.2%
APES (local) [Wu23b]	93.5%
APES (global) [Wu23b]	93.8%
SAMPS	94.1%
SAMBLE	**94.2%**

The qualitative results of SAMBLE are presented in Figure 6.6, which includes sampling score heatmaps, the employed sampling strategies, and the final sampled results. Results from APES and SAMPS are also included for comparison and analysis. In the logged shape bin histograms from SAMBLE, the learned bin partitioning strategy (blue bins) with bin sampling ratios (red polyline) are both presented. It is evident that SAMBLE has successfully learned shape-specific sampling strategies.

Comparing SAMBLE to APES, as shown in Figure 6.6, SAMBLE effectively samples an adequate number of edge points that form the overall structure of the shape. It also demonstrates improved global uniformity by not excessively focusing on edge points, particularly for thin or detailed parts. For instance, it selectively samples less points on chair legs to allocate more points for sampling from the chair seats, resulting in a more balanced representation.

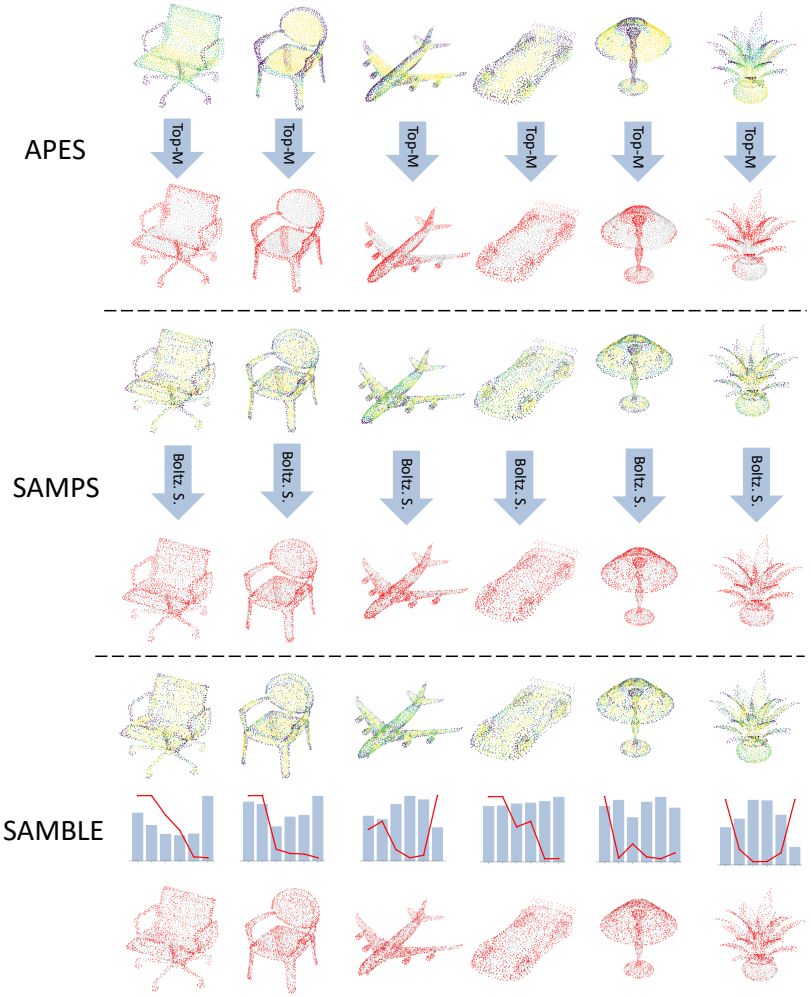


Figure 6.6: Qualitative results of the proposed SAMBLE, in comparison with the results from APES and SAMPS. Apart from the sampled results, sampling score heatmaps and bin histograms along with bin sampling ratios are also given. All shapes are from the test set. APES adopts the Top-M sampling policy, SAMPS uses the Boltzmann sampling policy, and SAMBLE employs the bin-based sampling policy.

In comparison to SAMPS, SAMBLE exhibits similar heatmaps as they share the same attention map construction and point-wise sampling score computation design. However, SAMBLE incorporates a bin-based sampling policy, enabling the learning of shape-specific sampling strategies. While SAMBLE still prioritizes sampling edge points, which correspond to the points in the first bin, it also leverages other point categories. For example, in the case of the airplane category, SAMBLE adopts a different sampling strategy than SAMPS by sampling more points from the last bin, which contains points with the smallest sampling scores and corresponds to the close-to-edge point category. This choice results in a slightly more uniformly distributed sampling output, which the model recognizes as a better sampling strategy.

Overall, SAMBLE successfully achieves a favorable balance between sampling edge points and preserving global uniformity. Additionally, it introduces a shape-specific sampling strategy that considers the unique characteristics exhibited by different shapes, resulting in an enhanced sampling approach.

Relationship between Bin Sampling Weights and Ratios

For the sake of brevity and improved visual clarity, in Figure 6.6, the axis labels of the histograms have been omitted. We further provide the full version of the histogram, in which the number of points and the sampling ratio in each bin are given. An example is provided in Figure 6.7. More detailed histogram results are provided later.

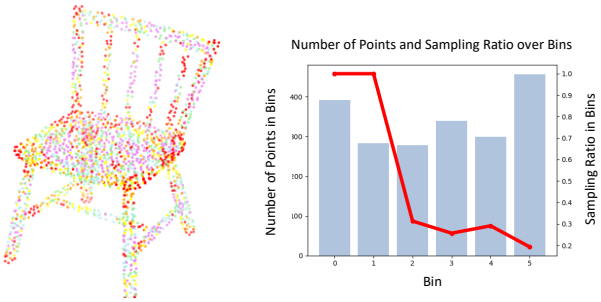


Figure 6.7: Left: bin partitioning, each color represents the points belonging to this bin. Right: the learned sampling strategy.

One thing worth noting is that the indicated sampling ratios $\mathbf{r}$ in the histogram are not simply re-scaled sampling weights $\boldsymbol{\omega}$. As in the algorithm we presented in this chapter, apart from the re-scaling operation, a redistribution operation is also applied to prevent κ_j surpassing the available point number β_j in one bin. Given the point counts in each bin $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_{n_b})$ and the numbers of points to be sampled from each bin $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \dots, \kappa_{n_b})$, the sampling ratios presented in the histogram are given by $\mathbf{r} = \boldsymbol{\kappa} \oslash \boldsymbol{\beta}$, where $\oslash$ denotes elment-wise division and $\mathbf{r} \in [0, 1]$.

The redistribution operation only happens when κ_j is about to surpass β_j, this means all points in jth bin have been selected and $r_j = 1$. We additionally count and document the likelihood of this occurrence for all bins across all test shapes. The numbers are reported in Table 6.2, for which we can see that for around 54% of the shapes, all points in the first bin are selected and sampled. Note that the first bin corresponds to the points of higher sampling scores which are mostly edge points with indexing mode vii. This observation underscores the significance of edge points. On the other hand, there are still around 46% shapes that do not sample all edge points. It suggests that an excessive emphasis on edge points might have adverse effects on subsequent downstream tasks, which also aligns with the conclusion drawn by APES.

Table 6.2: Possibilities of all points being sampled in bins, across all test shapes.

Bin Index	0	1	2	3	4	5
Possibilities of All Points Being Sampled	53.69%	27.11%	8.02%	2.11%	0.85%	4.98%

Further Visualizations of Learned Shape-specific Sampling Strategies

We present additional extensive results in Figure 6.8, Figure 6.9, Figure 6.10, and Figure 6.11 with various categories. The presented shape-specific bin partitioning histograms and bin sampling ratios reveal that SAMBLE has effectively learned shape-specific sampling strategies.

Moreover, from those qualitative figures, we can observe that shape edge points are mostly partitioned into the first two bins. Furthermore, in addition

to learning shape-wise sampling strategies for individual shapes, it is observed that analogous shapes within the same category exhibit similar histogram distributions and sampling strategies. Conversely, point clouds from different shape categories are sampled by distinct sampling strategies.

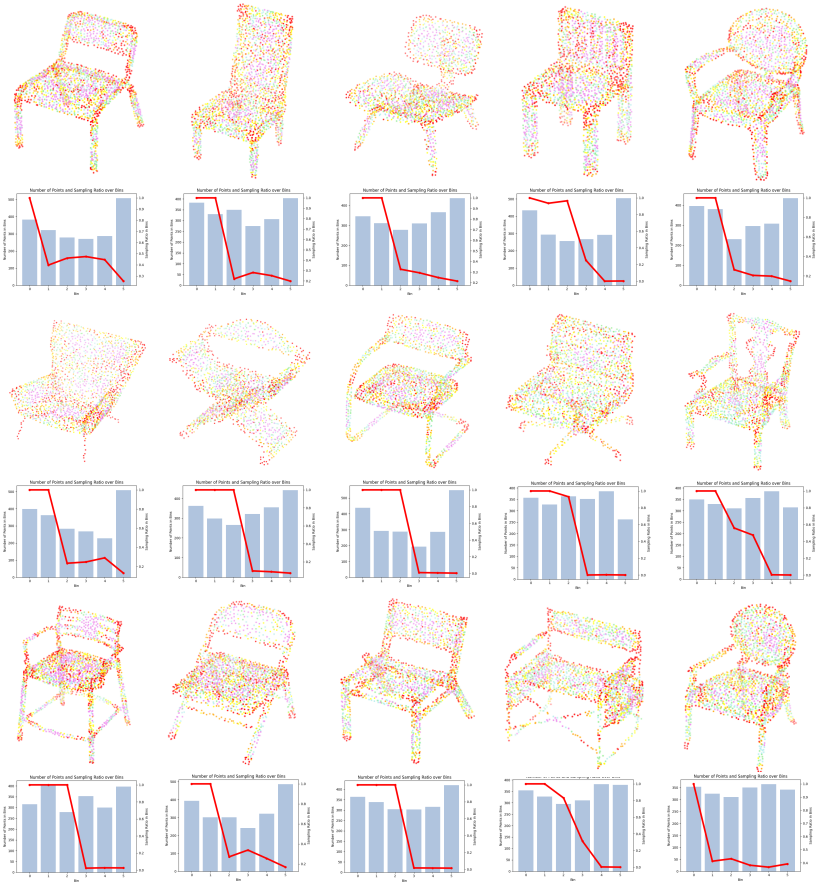


Figure 6.8: Additional visualizations of bin partitioning and learned shape-wise sampling strategies for the chair category. Zoom in for optimal visual clarity.

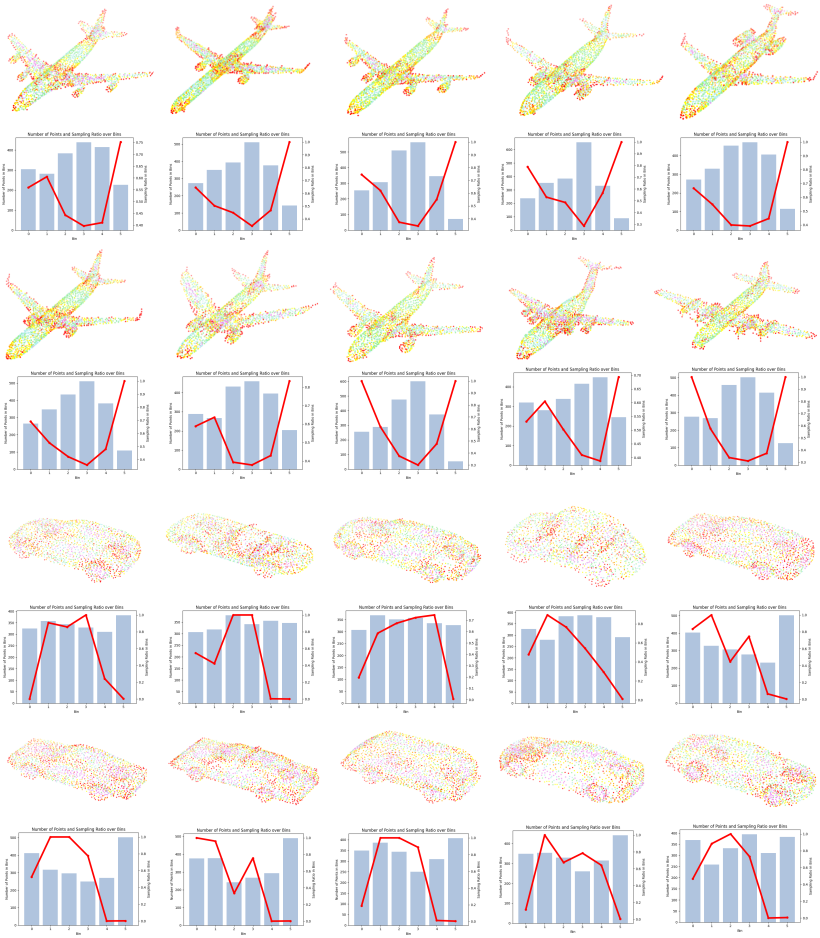


Figure 6.9: Additional visualizations of bin partitioning and learned shape-wise sampling strategies for the airplane and car categories. Zoom in for optimal visual clarity.

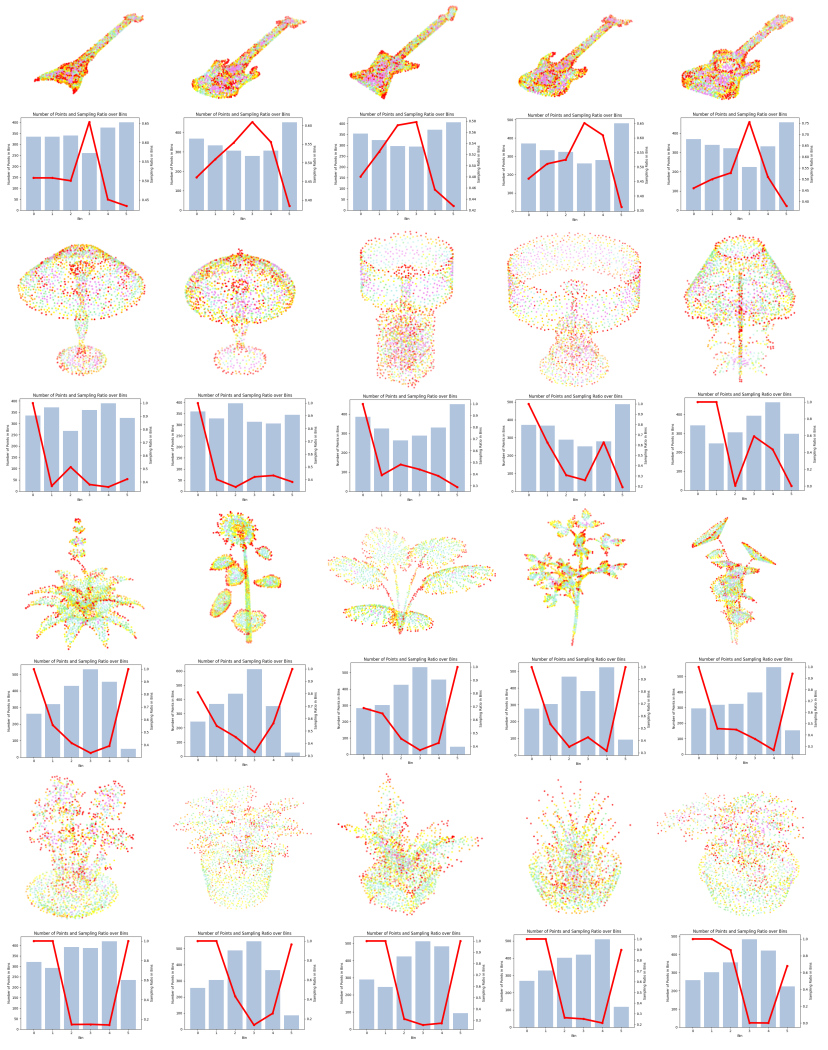


Figure 6.10: Additional visualizations of bin partitioning and learned shape-wise sampling strategies for the guitar, lamp, plant, and flower pot categories. Zoom in for optimal visual clarity.

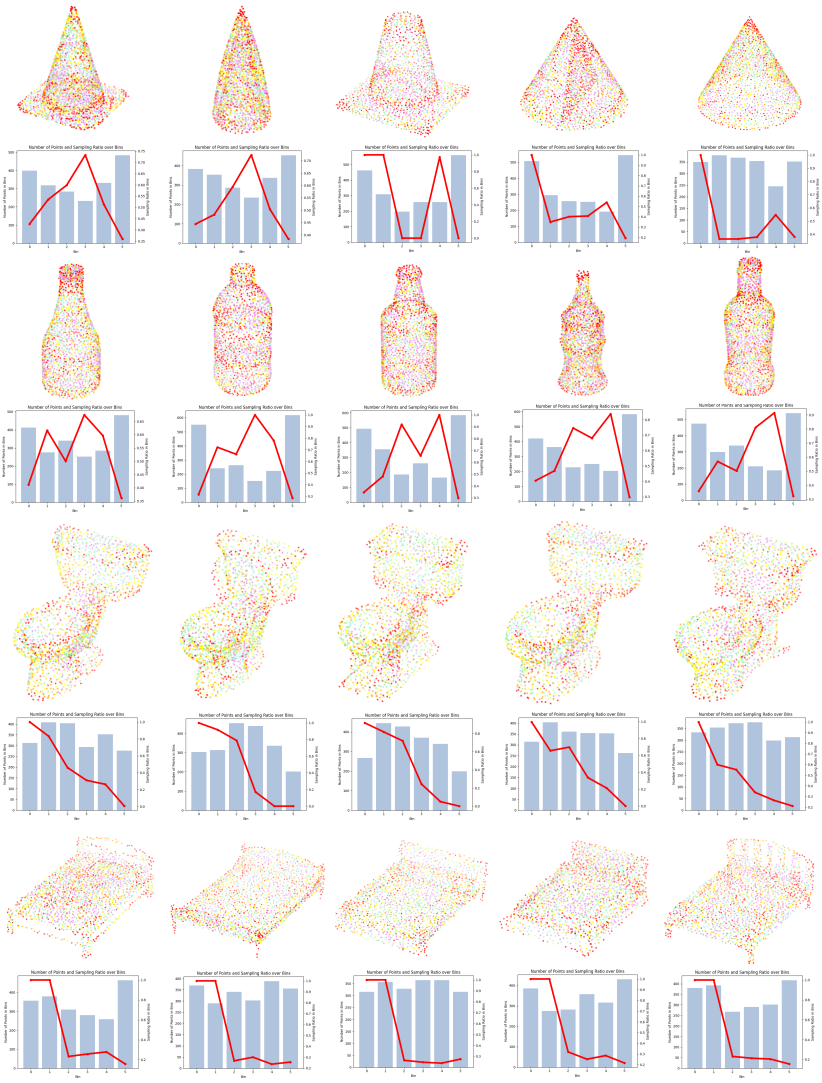


Figure 6.11: More visualization results of bin partitioning and learned shape-wise sampling strategies. The cone, bottle, toilet, and bed categories. Zoom in for optimal visual clarity.

6.3.2 Segmentation

Training Details

For a fair comparison, we use the same network architecture in SAMPS but with our proposed new downsampling layer. All the training settings are kept consistent. AdamW is used as the optimizer. The learning rate starts from 1×10^{-4} and decays to 1×10^{-8} with a cosine annealing schedule. The weight decay hyperparameter for network weights is 1×10^{-4}. We use $n_b = 4$ bins for point partitioning. The momentum update factor $\gamma = 0.99$ for updating boundary values. The temperature parameter $\tau = 0.1$. The network is trained with a batch size of 16 for 200 epochs. Each element x_b of bin tokens is initialized as Equation (6.12) and d is 128 in our case.

Quantitative and Qualitative Results

Table 6.3: Performance of SAMBLE on the ShapeNet Part Segmentation benchmark.

Method	Cat. mIoU	Ins. mIoU
PointNet [Qi17a]	80.4%	83.7%
PointNet++ [Qi17b]	81.9%	85.1%
SpiderCNN [Xu18]	82.4%	85.3%
DGCNN [Wan19]	82.3%	85.2%
SPLATNet [Su18]	83.7%	85.4%
PointCNN [Li18b]	84.6%	86.1%
PointConv [Wu19]	82.8%	85.7%
KPConv [Tho19]	85.0%	86.2%
PT[1] [Eng21]	-	85.9%
PT[2] [Zha21c]	83.7%	86.6%
PCT [Guo21]	-	86.4%
PRA-Net [Che21b]	83.7%	86.3%
PAConv [Xu21]	84.6%	86.1%
CurveNet [Muz20]	-	86.6%
DeltaConv [Wie22]	-	86.6%
StratifiedTransformer [Lai22]	**85.1%**	86.6%
PointNeXt [Qia22]	84.4%	**86.7%**
PointMLP [Ma22]	84.6%	86.1%
PointMetaBase [Lin23]	84.3%	**86.7%**
APES [Wu23b]	83.7%	85.8%
SAMPS	84.2%	86.4%
SAMBLE	84.5%	**86.7%**

The quantitative results are presented in Table 6.3. SAMBLE not only achieves superior quantitative results compared to APES and SAMPS but also attains state-of-the-art performance. In terms of the instance mIoU metric, SAMBLE performs on par with other state-of-the-art methods. Although it falls slightly short of achieving the SOTA results in the category mIoU metric, its performance is comparable to the latest prominent methods, including PointNeXt [Qia22], PointMLP [Ma22], and PointMetaBase [Lin23].

For the part segmentation benchmark, we further report the performance on the intermediate downsampled sub-point clouds in Table 6.4. Additionally, results from PointNeXt [Qia22] are also presented, which is a prominent point cloud learning method that employs FPS for downsampling.

It is evident that FPS-based methods exhibit poorer performance when applied to downsampled sub-point clouds. In contrast, our SAMBLE approach demonstrates improved performance with downsampled sub-point clouds. Additionally, SAMBLE employs the same interpolation-based upsampling layer as SAMPS. Although it outperforms the upsampling layer used in APES, there is potential for further enhancement by designing a more meticulously crafted upsampling layer. Such an upsampling layer could better reconstruct the features of the discarded points, leading to further improvements in SAMBLE's performance.

Table 6.4: Segmentation performances on downsampled point clouds.

Method	PointNeXt			SAMBLE		
Point Number	2048	1024	512	2048	1024	512
Cat. mIoU (%)	84.40	83.79	82.77	84.51	84.84	**85.04**
Ins. mIoU (%)	86.70	86.18	85.18	86.67	86.93	**87.12**

The qualitative results, showcased in Figure 6.12, provide valuable insights into the performance of SAMBLE. These qualitative findings in Figure 5.14 align with the quantitative results presented in Table 5.2, further solidifying the superiority of our proposed method. Upon closer examination, several notable observations can be made.

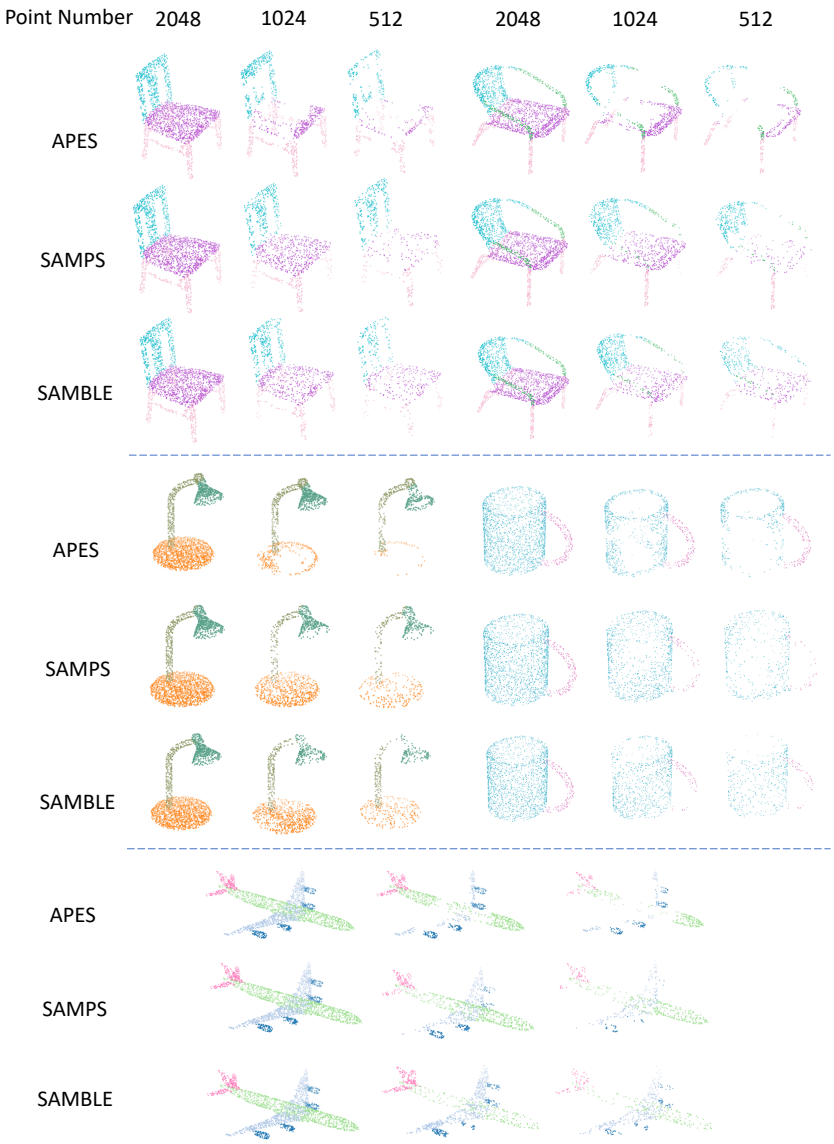


Figure 6.12: Segmentation results of the proposed SAMBLE, in comparison with APES and SAMPS. All shapes are from the test set.

Firstly, when comparing SAMBLE to SAMPS, SAMBLE exhibits a more balanced utilization of non-edge points, as exemplified by the chair seat. It demonstrates a thoughtful sampling strategy that takes into account both edge and non-edge points, resulting in a more comprehensive representation of the shape. Moreover, SAMBLE excels at preserving intricate details while simultaneously striving for a more uniform sampling distribution. This is evident in the case of airplane engines, where SAMBLE effectively captures the fine features while maintaining a more even distribution of sampled points across the shape. Overall, these qualitative results effectively showcase the strengths of SAMBLE, which can be attributed to the learned shape-specific sampling strategies.

6.3.3 Few Point Sampling

Experiment Setting

Apart from APES and SAMPS, we additionally compare our sampling method to previous work including RS, FPS, and the more recent learning-based S-Net, SampleNet, LighTN, etc. The same evaluation framework from [Dov19, Wan23, Wu23b] is used. The task here is the ModelNet40 Classification, and the task network is PointNet. Sampling methods are evaluated with multiple sampling sizes.

Moreover, it is important to note that APES addresses its limitations in few-point sampling by employing FPS to pre-sample the input into $2\boldsymbol{M}$ points [Wu23b]. However, this pre-processing step introduces an additional bias and complexity that may not be present in other methods, potentially affecting the fairness of the comparison. To ensure a fair and unbiased evaluation, we conducted experiments specifically on APES without the FPS pre-processing step. By removing this additional step, we can assess the performance of APES under more comparable conditions, aligning it with the evaluation setup of other methods.

Quantitative and Qualitative Results

The numerical results are showcased in Table 6.5. For a fair comparison, no FPS pre-sampling is employed. The results distinctly illustrate that our SAMBLE method attains state-of-the-art performance in the few-point sampling task, particularly noteworthy when the number of sampled points diminishes to smaller quantities.

Table 6.5: Comparison with other sampling methods. Evaluated on the ModelNet40 classification benchmark with multiple sampling sizes. For APES, we report its performance when pre-processing is not performed.

M	Voxel	RS	FPS [Eld97]	S-NET [Dov19]
512	73.82	87.52	88.34	87.80
256	73.50	77.09	83.64	82.38
128	68.15	56.44	70.34	77.53
64	58.31	31.69	46.42	70.45
32	20.02	16.35	26.58	60.70
M	PST-NET [Wan21]	SampleNet [Lan20]	MOPS-Net [Qia20]	DA-Net [Lin21]
512	87.94	88.16	86.67	89.01
256	83.15	84.27	86.63	86.24
128	80.11	80.75	86.06	85.67
64	76.06	79.86	85.25	85.55
32	63.92	77.31	84.28	85.11
M	LighTN [Wan23]	APES (w/o pre-pro.)	SAMPS	SAMBLE
512	89.91	89.81	90.46	**90.58**
256	88.21	86.78	90.12	**90.18**
128	86.26	84.87	89.81	**90.02**
64	86.51	79.23	89.66	**89.81**
32	86.18	75.63	89.25	**89.45**

Qualitative results are presented in Figure 6.13, Figure 6.14 and Figure 6.15. For few-point sampling, APES has to use FPS to pre-sample the input into $2M$ points due to its limitations [Wu23b]. In contrast to that, since our method preserves better global uniformity, direct few-point sampling from the input still yields satisfactory sampled results.

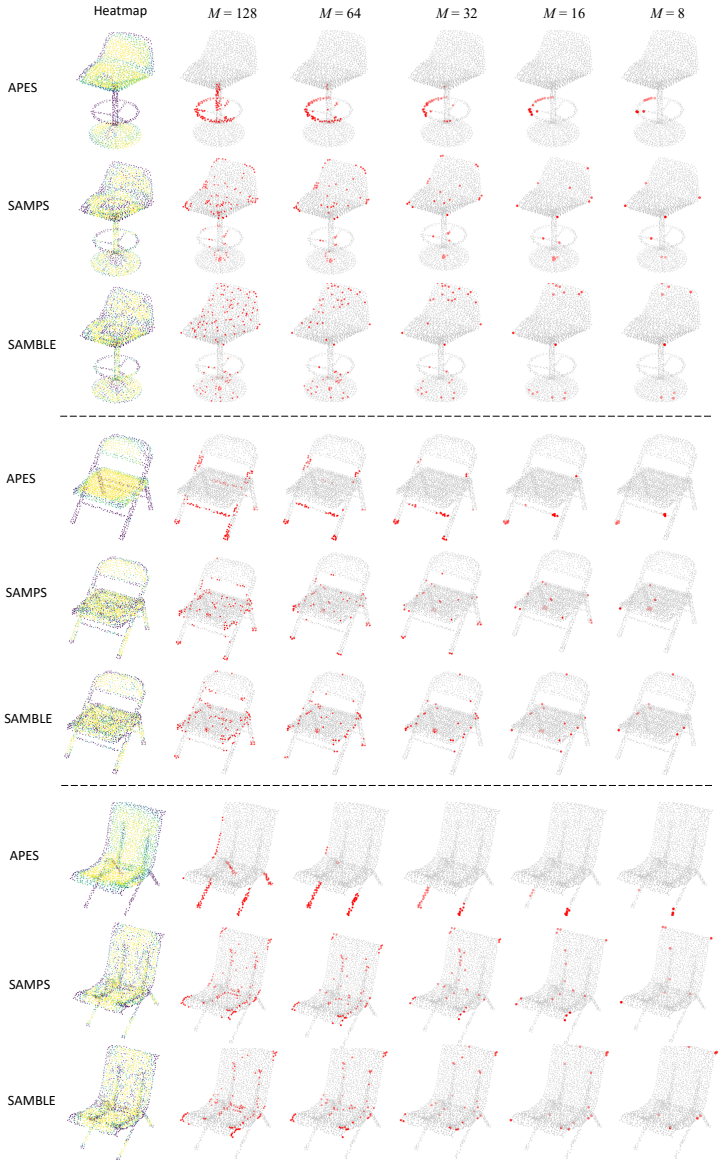


Figure 6.13: Sampled results of few-point sampling with SAMBLE, in comparison with APES and SAMPS. No pre-FPS into $2M$ points was performed.

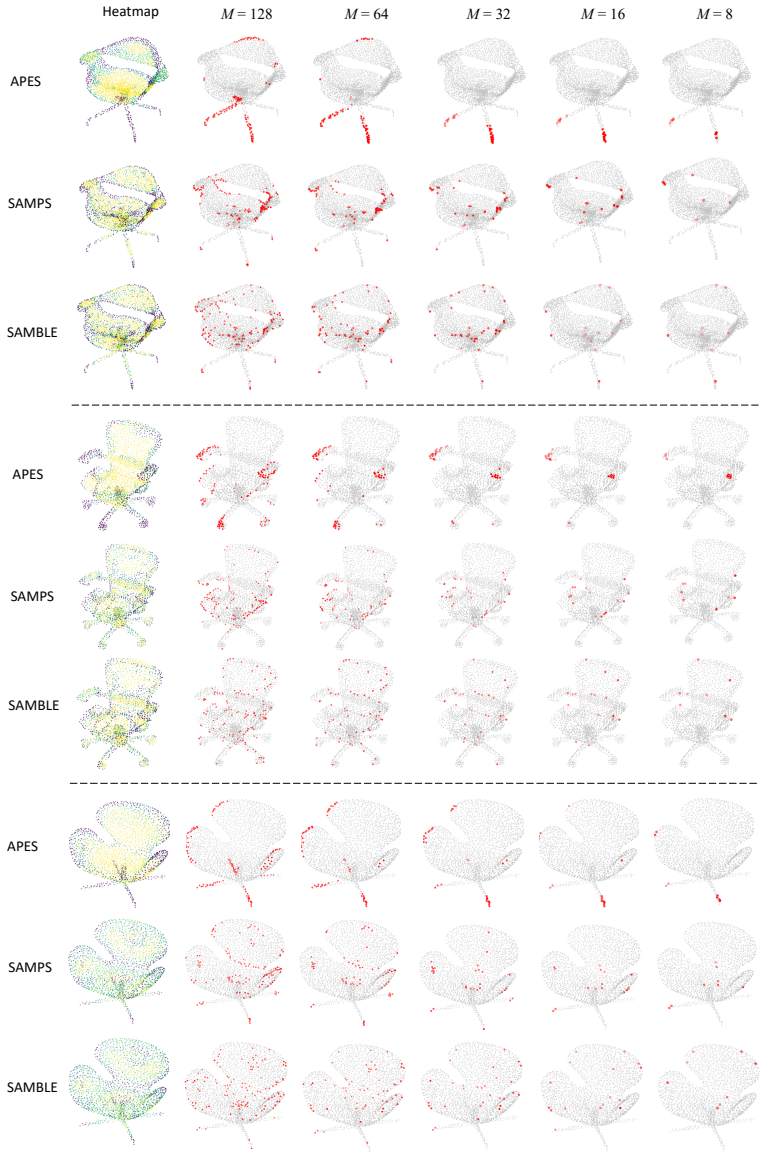


Figure 6.14: Sampled results of few-point sampling with SAMBLE, in comparison with APES and SAMPS. No pre-FPS into $2M$ points was performed.

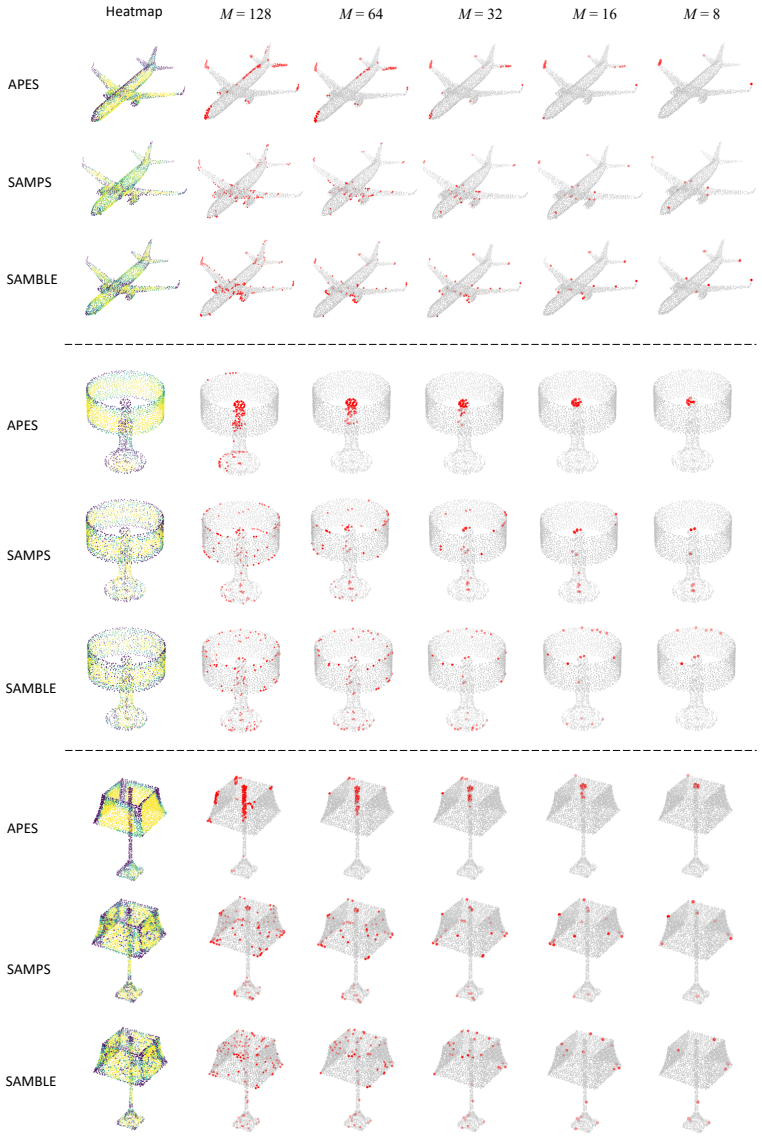


Figure 6.15: Sampled results of few-point sampling with SAMBLE, in comparison with APES and SAMPS. No pre-FPS into $2M$ points was performed.

From Figure 6.13, Figure 6.14 and Figure 6.15, it is evident that when sampling very few points directly from the input, APES tends to concentrate its sampling on the sharpest locations, leading to a skewed distribution, whereas our SAMBLE method maintains superior global uniformity in its sampling approach.

The comparison with the few-point sampling results from SAMPS offers an intriguing insight. SAMPS and SAMBLE exhibit similar heatmaps due to their shared attention map construction and point-wise sampling score computation design. However, as evident in Figure 6.13, SAMPS excels when shapes feature straight outlines but shows a slight decrease in performance with shapes characterized by predominantly round outlines, as illustrated in Figure 6.14. In contrast, SAMBLE maintains better global uniformity when sampling few points, regardless of the shapes' outline characteristics. This distinction is particularly pronounced when examining the lamp objects presented in Figure 6.15.

The reason behind this difference lies in SAMPS' utilization of a Boltzmann sampling policy, which, while achieving a commendable balance between capturing details and preserving overall structure compared to APES, still maintains relatively low probabilities for sampling non-edge points in comparison to edge points. Consequently, crucial non-edge points may not be sampled, potentially impacting downstream tasks that rely on these points. Conversely, SAMBLE integrates a bin-based sampling policy, enabling the learning of shape-specific sampling strategies. This approach allows for non-edge points to have a higher probability of being sampled, enhancing task performance. As a result, SAMBLE achieves a more refined balance between capturing shape details and preserving the overall structure compared to SAMPS.

6.3.4 Ablation Study

Within this subsection, our primary emphasis is placed on elucidating the innovative design features that have been introduced in this chapter. We purposefully steer clear of reiterating common topics that have been exhaustively covered in APES and SAMPS.

Number of Bins

As a key parameter in SAMBLE, an ablation study is performed over the number of bins n_b. The results are presented in table 6.6. Remarkably, increasing the number of bins does not yield improved performance. This phenomenon is likely attributable to the subdivision of shapes into an excessive number of point categories, leading to the gradual diminishment of score disparities across the bins. In our case, $n_b = 6$ and $n_b = 4$ yield the best performance for the classification and segmentation tasks respectively, and we use it for the corresponding experiments.

Table 6.6: Classification and segmentation performance with different number of bins.

Number of Bins		1	2	4	6	8	10	12
Cls.	OA (%)	93.95	93.91	93.98	**94.18**	94.02	93.80	93.84
Seg.	Cat. mIoU (%)	84.22	84.14	**84.51**	84.40	84.19	83.98	84.36
	Ins. mIoU (%)	86.46	86.28	**86.67**	86.61	86.48	86.23	86.43

Momentum Update Factor

The momentum update strategy is widely used within contrastive learning frameworks in self-supervised learning. In our case, we aim to derive the bin boundary values $\boldsymbol{\nu}$ from the entirety of shapes within the training dataset. These values aim to evenly partition the distribution of point sampling scores across all shapes and points in the training data. Hence such an adaptive learning method is used.

An ablation study over the momentum update parameter γ is performed and the numerical results are reported in Table 6.7. From it, we can see that $\gamma = 0.99$ yields the best performance. This actually aligns with most current contrastive learning frameworks, where a majority use a value of $\gamma = 0.99$.

Table 6.7: Classification performance with different values of the momentum update factor γ.

γ		0.9	0.99	0.999	0.9999
Cls.	OA (%)	93.80	**94.18**	94.02	93.95

We additionally provide the bin partitioning results over the test dataset with the learned boundary values $\boldsymbol{\nu}$ in Figure 6.16. It demonstrates that the boundary values adaptively learned from the training dataset can also effectively partition the distribution of point sampling scores evenly across all shapes and points in the test dataset.

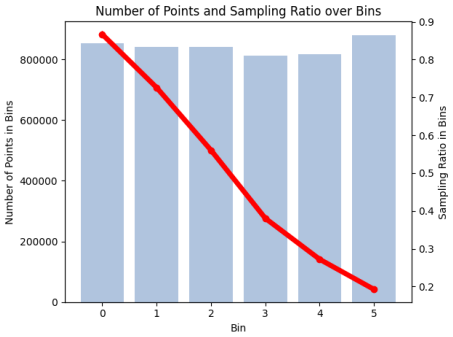


Figure 6.16: Partitioning the distribution of point sampling scores of all shapes and points in the test dataset into bins with the learned boundary values.

In-Bin Sampling policy and Temperature Parameter

In Section 6.2.3, four sampling approaches are described: TopM, Uniform, Sampling with fixed τ, and Sampling with adaptive τ. The Top-M and Uniform approaches can be regarded as two extremes of sampling with fixed τ, where τ equals 0 and $+\infty$, respectively. To evaluate performance, experiments were conducted, with results presented in Table 6.8 and Table 6.9.

Table 6.8: Classification and segmentation performance of the model with different temperature τ value.

	τ	0.01	0.02	0.05	0.1	0.2	0.5	1	10
Cls.	OA (%)	93.84	93.96	94.06	**94.18**	93.89	93.84	93.74	93.70
Seg.	Cat. mIoU (%)	84.10	84.23	84.38	**84.51**	84.26	84.13	84.02	83.88
	Ins. mIoU (%)	86.44	86.48	86.60	**86.67**	86.51	86.42	86.29	86.23

Table 6.9: Classification and segmentation performance of the model with different in-bin sampling policies. The temperature value $\tau = 0.1$.

In-Bin Sampling Policy		Top-M	Uniform	Fixed τ	Adaptive τ
Cls.	OA (%)	93.82	93.72	**94.18**	94.05
Seg.	Cat. mIoU (%)	84.14	83.86	**84.51**	84.32
	Ins. mIoU (%)	86.43	86.24	**86.67**	86.61

Upon reviewing Table 6.8 and Table 6.9, it becomes apparent that employing a fixed temperature parameter as the in-bin sampling policy, specifically setting $\tau = 0.1$, yields optimal performance. Consequently, this configuration is adopted for the majority of our experiments.

Interestingly, the results indicate that performance variations are minimal when adjusting τ or the in-bin sampling policy. This phenomenon can be attributed to the efficacy of the learned shape-specific sampling strategy, which effectively regulates the number of points to be sampled within each bin, acting as a holistic control mechanism over the entire shape. For instance, if the learned strategy dictates sampling 100% of points for the first bin (associated with edge points), altering the in-bin sampling policy will not impact the sampling outcome for this bin. Thus, while attaining shape-specific sampling strategies, the proposed bin-based sampling policy concurrently ensures comprehensive control over the entirety of the shape, enhancing the stability of the sampling process in comparison to SAMPS.

Order of Mean-pooling and ReLU operation

Within our design, the ReLU operation is used to prevent the learned sampling weight from being negative. However, the inherent distribution of values within tensors often results in a non-negligible proportion being negative. Directly setting too many values to zero would result in a significant loss of features, which is regrettable considering the potential information discarded. Hence, instead of performing the ReLU operation before the mean-pooling operation, we do it the other way around, i.e., first mean-pooling, then ReLU is performed over the pooled results.

Figure 6.17 gives the learned sampling strategies with the mean-pooling and ReLU operations applied in different orders. Although both orders yield shape-wise sampling strategies, the sampling ratios over bins learned with the order of ReLU first are mostly around 40% - 60%, leading to a worse sampling performance. On the other hand, the order of mean-pool first yields better sampling strategies as less potential information is discarded.

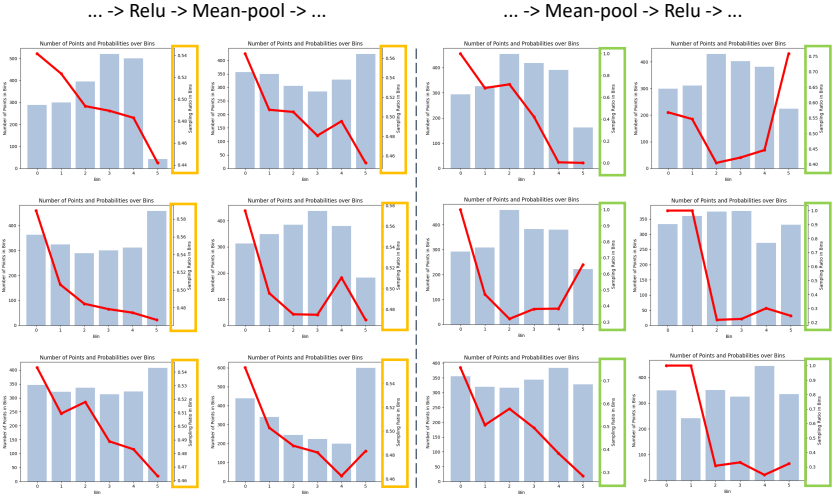


Figure 6.17: Learned sampling strategies with the mean-pooling and ReLU operations applied in different orders.

We additionally count and document the likelihood of ReLU being effective, which indicates the former pooled result is negative, for all bins across all test shapes. From the numbers reported in Table 6.10, we can see that the likelihood of the pooled results being negative is extremely small (less than 1%) for the first half of bins, while it goes higher for the latter bins yet the number is still relatively acceptable.

Table 6.10: Possibilities of ReLU being effective in bins, across all test shapes.

Bin Index	0	1	2	3	4	5
Possibilities of ReLU Being effective	0.45%	0.28%	0.57%	4.25%	11.63%	13.53%

Model Complexity

To evaluate SAMBLE's practicality, we assess its complexity in comparison with APES and SAMPS and report the results in Table 6.11. This includes details on model parameters and FLOPs for both the entire model and a single downsampling layer. In order to assess inference efficiency, experiments were carried out using a trained ModelNet40 classification model on a single NVIDIA GeForce RTX 3090. The tests were conducted with a batch size of 8, evaluating a total of 2468 shapes from the test set.

Table 6.11: For model Complexity, we report the number of model parameters and FLOPs for both the full model and one downsampling layer. The inference throughput (instances per second) is also reported.

Method	Params.		FLOPs		Throughput
	Full Model	One DS Layer	Full Model	One DS layer	(ins./sec.)
FPS + kNN	4.43M	20.90k	7.02G	2.71G	102
APES (local)	4.47M	49.15k	4.59G	1.09G	488
APES (global)	4.47M	49.15k	3.03G	0.05G	520
SAMPS (insert)	4.47M	49.15k	4.59G	1.09G	446
SAMPS (carve)	4.47M	49.15k	3.03G	0.05G	473
SAMBLE	4.48M	49.92k	3.56G	0.38G	125

As shown in Table 6.11, SAMBLE has a slightly larger number of model parameters compared to APES and SAMPS, primarily due to the incorporation of additional bin tokens. Furthermore, while SAMBLE also employed carve-based SAM, the integration of bin-based sampling policy results in slightly higher FLOPs compared to SAMPS with carve-based SAM. Despite this, SAMBLE's FLOPs remain lower than those of insert-based methods that necessitate computing attention between points and neighbors.

On the other hand, SAMBLE's inference throughput is reduced compared to SAMPS due to the introduction of bin partitioning operations. Notably, the process of determining the number of points to be sampled within each bin involves a CPU-intensive loop computation, which can lead to increased inference time. Moreover, SAMBLE consistently outperforms FPS in both performance and speed. Note that the FPS results presented here already use

a GPU-accelerated version, whereas its standard implementation achieves a much lower throughput (around 12). Overall, despite the computational considerations, SAMBLE's enhanced performance across downstream tasks underscores the effectiveness of its innovative sampling strategy.

6.4 Summary

In this chapter, a novel point cloud sampling method called SAMBLE is introduced based on SAMPS to learn shape-specific sampling strategies. SAMBLE computes point-wise sampling scores using the SAM proposed in Chapter 5, which enhances the discrimination among different point categories, enabling the partitioning of points into bins based on these scores. By adaptively learning bin boundaries for each shape and determining sampling ratios for each bin using additional tokens during attention computation, SAMBLE customizes sampling strategies for individual point cloud shapes. Multiple in-bin sampling policies have also been explored in this chapter. Overall, SAMBLE achieves a refined balance between sampling edge points and maintaining global shape uniformity through tailoring shape-specific sampling strategies, resulting in improved performance on downstream tasks.

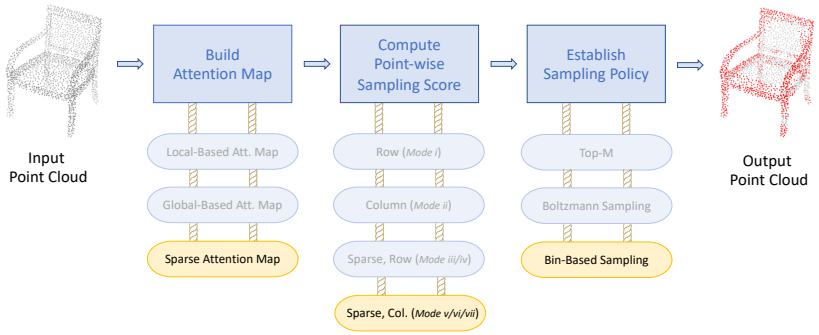


Figure 6.18: In this chapter, SAMBLE is introduced with a novel bin-based sampling policy for attention-based point cloud sampling. It achieves better performance through learning shape-specific sampling strategies.

Looking ahead, the integration of SAMBLE's shape-specific sampling strategies opens up promising avenues for further advancements in point cloud processing. However, challenges remain in fully resolving the edge sampling trade-off problem for point clouds. Currently, a point's sampling probability primarily hinges on its scalar sampling score rather than its latent feature vector, indicating room for enhancement in the sampling process. Moreover, the potential for refining more advanced upsampling layers by incorporating previously deduced information has not been fully explored. The intricate interplay between the downsampling and upsampling layers presents a rich domain for future research, aiming to push the boundaries of performance and versatility in point cloud analysis and processing.

7 Concluding Remarks

7.1 Conclusion

In this dissertation, we have explored the domain of point cloud sampling with a focus on integrating task-oriented learning and mathematical statistics-based direct point selection. Recognizing the inherent limitations of traditional and existing learning-based sampling methods, we proposed novel techniques to enhance sampling effectiveness while preserving point traceability and geometric fidelity, improving performance on point cloud downstream tasks.

Three novel methods are proposed and explored for point cloud edge sampling, each addressing specific challenges and improving upon its predecessor. These methods—APES, SAMPS, and SAMBLE—represent a progression in the sophistication and effectiveness of point cloud sampling techniques, offering valuable contributions to the field of 3D data processing and presenting invaluable insights to the field.

The introduction of APES marked a significant advancement in point cloud edge sampling, leveraging the well-established principles of the Canny edge detection algorithm adapted for 3D point cloud data. APES employs attention-based mechanisms to compute correlation maps, effectively identifying and sampling edge points within point clouds. The attention mechanism in APES allows it to focus on critical areas, thereby enhancing its capability to detect edge points in a high-dimensional space. This method pioneered the integration of task-oriented learning and mathematically traceable direct point selection, enabling gradient backpropagation during training while maintaining the traceability of sampled points.

APES introduced two variations: local-based APES and global-based APES, tailored to different attention modes. Both qualitative and quantitative results demonstrated that APES achieved excellent performance on common point cloud benchmark tasks. However, findings indicated that while increasing the number of sampled edge points generally enhanced downstream task performance, an excessive focus on edge points could detract from overall shape uniformity, potentially leading to a negative impact on overall performance. This highlighted the necessity for a more sophisticated sampling method that balances edge point sampling with shape uniformity preservation, setting the stage for further developments.

Building on the foundation laid by APES, SAMPS was developed to overcome its limitations. SAMPS introduces a sparse attention map (SAM) that combines local and global information, enhancing its ability to capture a broader range of relevant features within the point cloud. One of the key innovations in SAMPS is the design of multiple indexing modes, which provide multiple ways to compute point-wise sampling scores. Instead of adopting a sampling policy of Top-M, SAMPS maps point-wise sampling scores into probabilities using the softmax function with a temperature parameter. This tunable sampling mechanism provides greater control over the process, achieving a more balanced trade-off between edge point sampling and maintaining global uniformity.

The efficacy of SAMPS is particularly evident in its ability to leverage interpolation-based upsampling layers. Moreover, it significantly enhances the quality of sampled points, especially when only a limited number of points are sampled. However, SAMPS also reveals an ongoing challenge in the field: the uniform application of the same sampling strategy to all point clouds fails to account for the inherent variations in point distributions across different shapes. This limitation underscores the need for more adaptive and shape-specific sampling strategies, which can better address the unique characteristics of diverse point clouds. This insight sets the stage for further advancements, driving the development of methods that can achieve more nuanced and effective sampling strategies tailored to specific shapes.

To address the limitations of SAMPS, SAMBLE was introduced as a novel point cloud sampling method designed to learn shape-specific sampling strategies.

SAMBLE also employs the sparse attention map proposed in SAMPS to enhance point category discrimination, allowing for the partitioning of points into bins based on sampling scores. This binning process is crucial as it tailors the sampling strategy to the unique characteristics of each shape within the point cloud. SAMBLE adaptively learns the boundaries of these bins for each shape and determines appropriate sampling ratios for each bin. This adaptive learning is facilitated by additional tokens during attention computation, which provide the flexibility needed to customize the shape-specific sampling strategy dynamically.

Within each bin, SAMBLE explores multiple in-bin sampling policies, ensuring that the selected points maintain a refined balance between sampling edge points and maintaining global shape uniformity. This approach allows SAMBLE to address the inherent variations in point distributions across different shapes more effectively than its predecessors, which applied a uniform sampling strategy to all point clouds. By focusing on learning shape-specific sampling strategies, SAMBLE achieves a higher degree of precision in point sampling, enhancing the overall quality of the sampled point cloud data and achieving better performance on point cloud downstream tasks.

Through extensive experimentation and evaluation on benchmark datasets, our proposed methods demonstrated significant improvements in point cloud sampling and downstream tasks such as shape classification, part segmentation, and few-point sampling. The qualitative and quantitative results validate the effectiveness of our approaches in capturing essential geometric features and enhancing overall performance. Notably, SAMPS and SAMBLE demonstrate a marked improvement in the quality of sampled points when only a limited number of points are sampled, highlighting their ability to capture specific geometric patterns during the learning process.

To conclude, this dissertation has presented a comprehensive exploration of point cloud sampling, introducing novel methods that integrate task-oriented learning with mathematical statistics-based direct point selection. The development of SAMBLE, along with APES and SAMPS, represents a significant progression in the field, each method building upon the insights and innovations of its predecessors to advance the state of point cloud sampling techniques.

By addressing key challenges and proposing innovative solutions, this work advances the state-of-the-art in point cloud sampling and paves the way for more efficient and effective 3D data processing in various applications. The proposed methods not only enhance the performance of downstream tasks but also provide a foundation for future research in this dynamic and evolving field.

7.2 Outlook

The methodologies developed in this research—APES, SAMPS, and SAMBLE—mark significant strides in point cloud sampling and present numerous promising directions for future work. One promising direction is the integration of latent feature vectors into the sampling process. Current methods primarily rely on scalar sampling scores, which, while effective, do not fully capture the rich, multidimensional nature of point cloud data. By incorporating latent feature vectors, we can develop sampling strategies that consider a broader spectrum of data attributes, leading to more nuanced and precise sampling decisions. This enhancement could significantly improve the performance of downstream tasks, such as classification and segmentation, by ensuring that the most informative points are retained in the sampled data.

The refinement of upsampling layers also presents a valuable direction for further research. Current methods have not fully explored the potential of using previously deduced information during the downsampling process to enhance upsampling processes. By developing more advanced upsampling techniques that leverage this information, we can create more seamless and accurate reconstructions of point clouds after sampling. The interplay between downsampling and upsampling layers offers a rich domain for innovation, with the potential to yield significant advances in both computational efficiency and the accuracy of 3D reconstructions.

Moreover, one exciting future avenue for exploration is the application of these techniques to point cloud scenes rather than isolated shapes. Point cloud scenes, such as those captured in autonomous driving or large-scale urban modeling, encompass more complex structures and a higher degree

of variability. Extending our sampling methods to these scenes can enhance the capability of current algorithms to handle more comprehensive datasets, leading to improved detection and recognition of objects within these scenes. However, this shift introduces the challenge of scene boundary points being incorrectly prioritized as important, necessitating the development of more sophisticated sampling algorithms that can accurately distinguish between boundary points and key internal features.

Applying our sampling methods to point cloud scenes can revolutionize how we approach detection tasks in various fields. For instance, in autonomous driving, the ability to accurately sample and interpret point cloud scenes can lead to better obstacle detection and navigation decisions. Similarly, in robotics, more effective point cloud sampling can improve the robot's ability to understand and interact with its environment, enhancing performance in tasks such as object manipulation and path planning. By tailoring sampling strategies to consider the unique characteristics of entire scenes, we can ensure that crucial features are preserved, while less important data is efficiently filtered out, leading to more robust and reliable detection systems.

Finally, the practical applications of improved point cloud sampling methods extend far beyond academic research. In industries ranging from autonomous driving and robotics to virtual reality and urban planning, the ability to effectively sample and analyze 3D data is crucial. By addressing the challenges outlined and refining these methodologies, we can develop tools that not only advance our understanding of point cloud data but also have a profound impact on real-world applications. The journey ahead is filled with opportunities to enhance our capability to navigate and interpret complex 3D environments, leading to more intelligent and capable systems that can operate with greater accuracy and reliability in a wide range of scenarios.

Bibliography

[Ahn22] Ahn, Pyunghwan; Yang, Juyoung; Yi, Eojindl; Lee, Chanho and Kim, Junmo: "Projection-Based Point Convolution for Efficient Point Cloud Segmentation". In: *IEEE Access* 10 (2022), pp. 15348–15358 (cit. on p. 30).

[Aud16] Audebert, Nicolas; Saux, Bertrand Le and Lefèvre, Sébastien: "Semantic Segmentation of Earth Observation Data using Multimodal and Multi-scale Deep Networks". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Springer. 2016, pp. 180–196 (cit. on p. 27).

[Bah14] Bahdanau, Dzmitry; Cho, Kyunghyun and Bengio, Yoshua: "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv preprint arXiv:1409.0473* (2014) (cit. on pp. 17, 19).

[Bha21] Bhattacharyya, Prarthana; Huang, Chengjie and Czarnecki, Krzysztof: "SA-Det3D: Self-Attention Based Context-Aware 3D Object Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 3022–3031 (cit. on p. 35).

[Bol68] Boltzmann, Ludwig: Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten. K. und k. Hof-und Staatsdr., 1868 (cit. on pp. 100, 110, 145).

[Bou17] Boulch, Alexandre; Le Saux, Bertrand and Audebert, Nicolas: "Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks." In: *3DOR Eurographics* 3 (2017) (cit. on p. 27).

[Bou20] Boulch, Alexandre: "ConvPoint: Continuous convolutions for point cloud processing". In: *Computers and Graphics* 88 (2020), pp. 24–34 (cit. on p. 30).

[Bro20] Brown, Tom B; Mann, Benjamin; Ryder, Nick; Subbiah, Melanie; Kaplan, Jared; Dhariwal, Prafulla; Neelakantan, Arvind; Shyam, Pranav; Sastry, Girish; Askell, Amanda et al.: "Language Models are Few-Shot Learners". In: *arXiv preprint arXiv:2005.14165* (2020) (cit. on p. 20).

[Bru13] Bruna, Joan; Zaremba, Wojciech; Szlam, Arthur and LeCun, Yann: "Spectral Networks and Locally Connected Networks on Graphs". In: *arXiv preprint arXiv:1312.6203* (2013) (cit. on p. 34).

[Can86] Canny, John: "A computational approach to edge detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 6 (1986), pp. 679–698 (cit. on p. 8).

[Cha15] Chang, Angel X; Funkhouser, Thomas; Guibas, Leonidas; Hanrahan, Pat; Huang, Qixing; Li, Zimo; Savarese, Silvio; Savva, Manolis; Song, Shuran; Su, Hao et al.: "ShapeNet: An Information-Rich 3D Model Repository". In: *arXiv preprint arXiv:1512.03012* (2015) (cit. on pp. 60, 61).

[Che21a] Chen, Can; Fragonara, Luca Zanotti and Tsourdos, Antonios: "GAPointNet: Graph Attention Based Point Neural Network for Exploiting Local Feature of Point Cloud". In: *Neurocomputing* 438 (2021), pp. 122–132 (cit. on pp. 33, 99).

[Che21b] Cheng, Silin; Chen, Xiwu; He, Xinwei; Liu, Zhe and Bai, Xiang: "PRA-Net: Point Relation-Aware Network for 3D Point Cloud Analysis". In: *IEEE Transactions on Image Processing* 30 (2021), pp. 4436–4448 (cit. on pp. 81, 85, 114, 118, 148, 156).

[Che22] Cheng, Ta-Ying; Hu, Qingyong; Xie, Qian; Trigoni, Niki and Markham, Andrew: "Meta-Sampler: Almost-Universal yet Task-Oriented Sampling for Point Clouds". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2022, pp. 694–710 (cit. on p. 131).

[Cla19] Clark, Kevin; Khandelwal, Urvashi; Levy, Omer and Manning, Christopher D.: "What Does BERT Look at? An Analysis of BERT's Attention". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019, pp. 2764–2773 (cit. on p. 20).

[Coh18] Cohen, Taco S; Geiger, Mario; Köhler, Jonas and Welling, Max: "Spherical CNNs". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2018 (cit. on p. 31).

[Dai19] Dai, Zihang; Yang, Zhilin; Yang, Yiming; Carbonell, Jaime; Le, Quoc V and Salakhutdinov, Ruslan: "Transformer-XL: Attentive Language Models beyond A Fixed-Length Context". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019 (cit. on p. 20).

[Def16] Defferrard, Michaël; Bresson, Xavier and Vandergheynst, Pierre: "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in Neural Information Processing Systems (NeurIPS)* 29 (2016) (cit. on p. 34).

[Dev19] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton and Toutanova, Kristina: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2019 (cit. on pp. 20, 24).

[Dos21] Dosovitskiy, Alexey; Beyer, Lucas; Kolesnikov, Alexander; Weissenborn, Dirk; Zhai, Xiaohua; Unterthiner, Thomas; Dehghani, Mostafa; Minderer, Matthias; Heigold, Georg; Gelly, Sylvain et al.: "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2021 (cit. on pp. 24, 25, 35, 134, 137).

[Dov19] Dovrat, Oren; Lang, Itai and Avidan, Shai: "Learning to Sample". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2760–2769 (cit. on pp. 2, 39, 40, 64, 80, 87, 88, 121, 122, 131, 159, 160).

[Dow14] Downie, Bill; Stephenson, Ian and Jackson, Paul: "Voxelization and Grid-Based Methods for Point Clouds using the CUDA Architecture". In: *Computers and Graphics* 38 (2014), pp. 262–271 (cit. on p. 37).

[Eld97] Eldar, Yuval; Lindenbaum, Michael; Porat, Moshe and Zeevi, Yehoshua Y: "The Farthest Point Strategy for Progressive Image Sampling". In: *IEEE Transactions on Image Processing* 6.9 (1997), pp. 1305–1315 (cit. on pp. 1, 37, 38, 88, 122, 160).

[Eng21] Engel, Nico; Belagiannis, Vasileios and Dietmayer, Klaus: "Point Transformer". In: *IEEE Access* 9 (2021), pp. 134826–134840 (cit. on pp. 35–37, 69, 81, 85, 99, 114, 118, 148, 156).

[Est18] Esteves, Carlos; Allen-Blanchette, Christine; Makadia, Ameesh and Daniilidis, Kostas: "Learning SO (3) Equivariant Representations with Spherical CNNs". In: *Proceedings of the 2018 European Conference on Computer Vision (ECCV)*. 2018, pp. 52–68 (cit. on p. 31).

[Fen19] Feng, Yifan; You, Haoxuan; Zhang, Zizhao; Ji, Rongrong and Gao, Yue: "Hypergraph Neural Networks". In: *Proceedings of the AAAI conference on artificial intelligence (AAAI)*. Vol. 33. 01. 2019, pp. 3558–3565 (cit. on p. 34).

[Gei19] Geirhos, Robert; Rubisch, Patricia; Michaelis, Claudio; Bethge, Matthias; Wichmann, Felix A and Brendel, Wieland: "ImageNet-Trained CNNs Are Biased towards Texture; Increasing Shape Bias Improves Accuracy and Robustness". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019 (cit. on p. 68).

[Gro18] Groh, Fabian; Wieschollek, Patrick and Lensch, Hendrik PA: "Flex-Convolution: Million-Scale Point-Cloud Learning beyond Grid-Worlds". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Springer. 2018, pp. 105–122 (cit. on pp. 2, 37–39).

[Guo20] Guo, Yulan; Wang, Hanyun; Hu, Qingyong; Liu, Hao; Liu, Li and Bennamoun, Mohammed: "Deep Learning for 3D Point Clouds: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 43.12 (2020), pp. 4338–4364 (cit. on pp. 30, 33).

[Guo21] Guo, Meng-Hao; Cai, Jun-Xiong; Liu, Zheng-Ning; Mu, Tai-Jiang; Martin, Ralph R and Hu, Shi-Min: "PCT: Point Cloud Transformer". In: *Computational Visual Media* 7 (2021), pp. 187–199 (cit. on pp. 35, 81, 85, 114, 118, 148, 156).

[He16] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing and Sun, Jian: "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778 (cit. on p. 68).

[Her18] Hermosilla, Pedro; Ritschel, Tobias; Vázquez, Pere-Pau; Vinacua, Àlvar and Ropinski, Timo: "Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds". In: *ACM Transactions On Graphics (TOG)* 37.6 (2018), pp. 1–12 (cit. on p. 31).

[Hou22] Hou, Zhixing; Yan, Yan; Xu, Chengzhong and Kong, Hui: "HiTPR: Hierarchical Transformer for Place Recognition in Point Cloud". In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2612–2618 (cit. on p. 99).

[Hu20] Hu, Qingyong; Yang, Bo; Xie, Linhai; Rosa, Stefano; Guo, Yulan; Wang, Zhihua; Trigoni, Niki and Markham, Andrew: "RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11108–11117 (cit. on p. 35).

[Hua18] Hua, Binh-Son; Tran, Minh-Khoi and Yeung, Sai-Kit: "Pointwise Convolutional Neural Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 984–993 (cit. on p. 32).

[Hua23] Huang, Zhuoxu; Zhao, Zhiyou; Li, Banghuai and Han, Jungong: "LCPformer: Towards Effective 3D Point Cloud Analysis via Local Context Propagation in Transformers". In: *IEEE Transactions on Circuits and Systems for Video Technology* 33.9 (2023), pp. 4985–4996 (cit. on p. 35).

[Jar09] Jarrett, Kevin; Kavukcuoglu, Koray; Ranzato, Marc'Aurelio and LeCun, Yann: "What Is the Best Multi-Stage Architecture for Object Recognition?" In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. 2009, pp. 2146–2153 (cit. on p. 14).

[Jia18] Jiang, Mingyang; Wu, Yiran; Zhao, Tianqi; Zhao, Zelin and Lu, Cewu: "Pointsift: A Sift-like Network Module for 3D Point Cloud Semantic Segmentation". In: *arXiv preprint arXiv:1807.00652* (2018) (cit. on p. 27).

[Kim21] Kim, Wonjae; Son, Bokyung and Kim, Ildoo: "ViLT: Vision-and-language Transformer without Convolution or Region Supervision". In: *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR. 2021, pp. 5583–5594 (cit. on pp. 25, 134, 137).

[Kip17] Kipf, Thomas N and Welling, Max: "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017 (cit. on p. 33).

[Kom19] Komarichev, Artem; Zhong, Zichun and Hua, Jing: "A-CNN: Annularly Convolutional Neural Networks on Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7421–7430 (cit. on p. 30).

[Kov19] Kovaleva, Olga; Romanov, Alexey; Rogers, Anna and Rumshisky, Anna: "Revealing the Dark Secrets of BERT". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019, pp. 4365–4374 (cit. on p. 20).

[Kri17] Krizhevsky, Alex; Sutskever, Ilya and Hinton, Geoffrey E: "ImageNet Classification with Deep Convolutional Neural Networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90 (cit. on p. 68).

[Kum19] Kumawat, Sudhakar and Raman, Shanmuganathan: "LP-3DCNN: Unveiling Local Phase in 3D Convolutional Neural Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4903–4912 (cit. on p. 32).

[Lai22] Lai, Xin; Liu, Jianhui; Jiang, Li; Wang, Liwei; Zhao, Hengshuang; Liu, Shu; Qi, Xiaojuan and Jia, Jiaya: "Stratified Transformer for 3D Point Cloud Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8500–8509 (cit. on pp. 36, 85, 86, 118, 156).

[Lan19] Lan, Shiyi; Yu, Ruichi; Yu, Gang and Davis, Larry S: "Modeling Local Geometric Structure of 3D Point Clouds using Geo-CNN". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 998–1008 (cit. on p. 32).

[Lan20] Lang, Itai; Manor, Asaf and Avidan, Shai: "SampleNet: Differentiable Point Cloud Sampling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 7578–7588 (cit. on pp. 2, 40, 64, 80, 87, 88, 122, 131, 160).

[Law17] Lawin, Felix Järemo; Danelljan, Martin; Tosteberg, Patrik; Bhat, Goutam; Khan, Fahad Shahbaz and Felsberg, Michael: "Deep Projective 3D Semantic Segmentation". In: *Proceedings of the International Conference on Computer Analysis of Images and Patterns (CAIP)*. Springer. 2017, pp. 95–107 (cit. on p. 27).

[Le18] Le, Truc and Duan, Ye: "Pointgrid: A Deep Network for 3D Shape Understanding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 9204–9214 (cit. on p. 27).

[Leh03] Lehtinen, Jaakko; Aila, Timo; Miettinen, Ville; Kautz, Jan and Akenine-Möller, Tomas: "Gradient-Domain Painting with Poisson Disk Sampling". In: *Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques*. ACM/SIGGRAPH, 2003, pp. 741–748 (cit. on p. 39).

[Lei19] Lei, Huan; Akhtar, Naveed and Mian, Ajmal: "Octree Guided CNN with Spherical Kernels for 3D Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9631–9640 (cit. on p. 32).

[Li18a] Li, Ruoyu; Wang, Sheng; Zhu, Feiyun and Huang, Junzhou: "Adaptive Graph Convolutional Neural Networks". In: *Proceedings of the AAAI conference on artificial intelligence (AAAI)*. Vol. 32. 1. 2018 (cit. on p. 34).

[Li18b] Li, Yangyan; Bu, Rui; Sun, Mingchao; Wu, Wei; Di, Xinhan and Chen, Baoquan: "PointCNN: Convolution on x-Transformed Points". In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018) (cit. on pp. 2, 30, 32, 38, 41, 81, 85, 99, 114, 117, 118, 148, 156).

[Li20] Li, Zirui; Xu, Junyu; Zhao, Yue; Li, Wenhui and Nie, Weizhi: "MPAN: Multi-Part Attention Network for Point Cloud Based 3D Shape Retrieval". In: *IEEE Access* 8 (2020), pp. 157322–157332 (cit. on p. 36).

[Lin20a] Lin, Yiqun; Yan, Zizheng; Huang, Haibin; Du, Dong; Liu, Ligang; Cui, Shuguang and Han, Xiaoguang: "FPconv: Learning Local Flattening for Point Convolution". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4293–4302 (cit. on p. 30).

[Lin20b] Lin, Zhi-Hao; Huang, Sheng-Yu and Wang, Yu-Chiang Frank: "Convolution in The Cloud: Learning Deformable Kernels in 3D Graph Convolution Networks for Point Cloud Analysis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 1800–1809 (cit. on pp. 33, 34).

[Lin21] Lin, Yanan; Huang, Yan; Zhou, Shihao; Jiang, Mengxi; Wang, Tianlong and Lei, Yunqi: "DA-Net: Density-Adaptive Downsampling Network for Point Cloud Classification via End-to-End Learning". In: *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*. IEEE. 2021, pp. 13–18 (cit. on pp. 2, 40, 80, 88, 122, 160).

[Lin23] Lin, Haojia; Zheng, Xiawu; Li, Lijiang; Chao, Fei; Wang, Shanshan; Wang, Yan; Tian, Yonghong and Ji, Rongrong: "Meta Architecture for Point Cloud Analysis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 17682–17691 (cit. on pp. 29, 118, 156, 157).

[Liu19a] Liu, Yongcheng; Fan, Bin; Meng, Gaofeng; Lu, Jiwen; Xiang, Shiming and Pan, Chunhong: "Densepoint: Learning Densely Contextual Representation for Efficient Point Cloud Processing". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 5239–5248 (cit. on p. 30).

[Liu19b] Liu, Yongcheng; Fan, Bin; Xiang, Shiming and Pan, Chunhong: "Relation-Shape Convolutional Neural Network for Point Cloud Analysis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8895–8904 (cit. on p. 33).

[Liu19c] Liu, Zhijian; Tang, Haotian; Lin, Yujun and Han, Song: "Point-Voxel CNN for Efficient 3D Deep Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)* 32 (2019) (cit. on pp. 81, 114, 148).

[Liu22] Liu, Haotian; Cai, Mu and Lee, Yong Jae: "Masked Discrimination for Self-Supervised Learning on Point Clouds". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2022, pp. 657–675 (cit. on p. 36).

[Lu22a] Lu, Dening; Gao, Kyle; Xie, Qian; Xu, Linlin and Li, Jonathan: "3DPCT: 3D Point Cloud Transformer with Dual Self-Attention". In: *arXiv preprint arXiv:2209.11255* (2022) (cit. on p. 35).

[Lu22b] Lu, Dening; Xie, Qian; Gao, Kyle; Xu, Linlin and Li, Jonathan: "3DCTN: 3D Convolution-Transformer Network for Point Cloud Classification". In: *IEEE Transactions on Intelligent Transportation Systems (TITS)* 23.12 (2022), pp. 24854–24865 (cit. on pp. 35, 99).

[Ma22] Ma, Xu; Qin, Can; You, Haoxuan; Ran, Haoxi and Fu, Yun: "Rethinking Network Design and Local Geometry in Point Cloud: A Simple Residual MLP Framework". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2022 (cit. on pp. 29, 118, 156, 157).

[Maa13] Maas, Andrew L; Hannun, Awni Y; Ng, Andrew Y et al.: "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: *Proceedings of the International Conference on Machine Learning (ICML)*. Vol. 30. 1. Atlanta, GA. 2013, p. 3 (cit. on p. 15).

[Mao19] Mao, Jiageng; Wang, Xiaogang and Li, Hongsheng: "Interpolated convolutional networks for 3d point cloud understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 1578–1587 (cit. on p. 30).

[Mat15] Maturana, Daniel and Scherer, Sebastian: "Voxnet: A 3D Convolutional Neural Network for Real-Time Object Recognition". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 922–928 (cit. on p. 27).

[McC92] McCool, Michael and Fiume, Eugene: "Hierarchical Poisson Disk Sampling Distributions". In: *Proceedings of Graphics Interface*. 1992, pp. 94–105 (cit. on p. 39).

[Moe03] Moenning, Carsten and Dodgson, Neil A: Fast Marching Farthest Point Sampling. Tech. rep. University of Cambridge, Computer Laboratory, 2003 (cit. on pp. 1, 37).

[Muz20] Muzahid, AAM; Wan, Wanggen; Sohel, Ferdous; Wu, Lianyao and Hou, Li: "CurveNet: Curvature-Based Multitask Learning Deep Networks for 3D Object Recognition". In: *IEEE/CAA Journal of Automatica Sinica* 8.6 (2020), pp. 1177–1187 (cit. on pp. 34, 81, 85, 114, 118, 148, 156).

[Nai10] Nair, Vinod and Hinton, Geoffrey E: "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2010, pp. 807–814 (cit. on p. 14).

[Nez20] Nezhadarya, Ehsan; Taghavi, Ehsan; Razani, Ryan; Liu, Bingbing and Luo, Jun: "Adaptive Hierarchical Down-sampling for Point Cloud Classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 12956–12964 (cit. on p. 41).

[Ope23] OpenAI: "GPT-4 Technical Report". In: *arXiv preprint arXiv:2303.08774* (2023) (cit. on p. 20).

[Pan21] Pan, Xuran; Xia, Zhuofan; Song, Shiji; Li, Li Erran and Huang, Gao: "3d Object Detection with Pointformer". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 7463–7472 (cit. on p. 35).

[Pan22] Pang, Yatian; Wang, Wenxiao; Tay, Francis EH; Liu, Wei; Tian, Yonghong and Yuan, Li: "Masked Autoencoders for Point Cloud Self-Supervised Learning". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2022, pp. 604–621 (cit. on p. 36).

[Pau02] Pauly, Mark; Gross, Markus and Kobbelt, Leif P.: "Efficient Simplification of Point-Sampled Surfaces". In: *Proceedings of the Conference on Visualization*. IEEE Computer Society, 2002, pp. 163–170 (cit. on p. 39).

[Pau03] Pauly, Mark; Keiser, Richard and Gross, Markus: "Multi-Scale Feature Extraction on Point-Sampled Surfaces". In: *Computer Graphics Forum*. Vol. 22. 3. Wiley Online Library. 2003, pp. 281–289 (cit. on p. 2).

[Pou19] Poulenard, Adrien; Rakotosaona, Marie-Julie; Ponty, Yann and Ovsjanikov, Maks: "Effective Rotation-Invariant Point CNN with Spherical Harmonics Kernels". In: *Proceedings of the International Conference on 3D Vision (3DV)*. IEEE. 2019, pp. 47–56 (cit. on p. 31).

[Qi17a] Qi, Charles R; Su, Hao; Mo, Kaichun and Guibas, Leonidas J: "Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 652–660 (cit. on pp. 27, 28, 61, 81, 84, 85, 99, 114, 118, 148, 156).

[Qi17b] Qi, Charles Ruizhongtai; Yi, Li; Su, Hao and Guibas, Leonidas J: "Pointnet++: Deep Hierarchical Feature Learning on Point Sets in A Metric Space". In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017) (cit. on pp. 2, 27–29, 33, 38, 81, 85, 86, 99, 117, 118, 156).

[Qi20] Qi, Haozhe; Feng, Chen; Cao, Zhiguo; Zhao, Feng and Xiao, Yang: "P2B: Point-to-Box Network for 3D Object Tracking in Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 6329–6338 (cit. on p. 38).

[Qia20] Qian, Yue; Hou, Junhui; Zhang, Qijian; Zeng, Yiming; Kwong, Sam and He, Ying: "MOPS-Net: A Matrix Optimization-Driven Network for Task-Oriented 3D Point Cloud Downsampling". In: *arXiv preprint arXiv:2005.00383* (2020) (cit. on pp. 2, 41, 80, 88, 122, 160).

[Qia22] Qian, Guocheng; Li, Yuchen; Peng, Houwen; Mai, Jinjie; Hammoud, Hasan; Elhoseiny, Mohamed and Ghanem, Bernard: "PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies". In: *Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022), pp. 23192–23204 (cit. on pp. 29, 38, 114, 117, 118, 148, 156, 157).

[Rad18] Radford, Alec; Narasimhan, Karthik; Salimans, Tim and Sutskever, Ilya: "Improving Language Understanding by Generative Pre-Training". In: *OpenAI Blog* 1 (2018), pp. 1–12 (cit. on p. 20).

[Rad19] Radford, Alec; Wu, Jeffrey; Child, Rewon; Luan, David; Amodei, Dario and Sutskever, Ilya: "Language Models are

Unsupervised Multitask Learners". In: *OpenAI Blog*. Vol. 1. 8. 2019, p. 9 (cit. on p. 20).

[Rao19] Rao, Yongming; Lu, Jiwen and Zhou, Jie: "Spherical Fractal Convolutional Neural Networks for Point Cloud Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 452–460 (cit. on p. 32).

[Ron15] Ronneberger, Olaf; Fischer, Philipp and Brox, Thomas: "U-net: Convolutional Networks for Biomedical Image Segmentation". In: *Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer. 2015, pp. 234–241 (cit. on p. 35).

[San19] Sanh, Victor; Debut, Lysandre; Chaumond, Julien and Wolf, Thomas: "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019 (cit. on p. 20).

[Ser19] Serrano, Sofia and Smith, Noah A.: "Is Attention Interpretable?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019, pp. 2931–2951 (cit. on p. 20).

[Sim17] Simonovsky, Martin and Komodakis, Nikos: "Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 3693–3702 (cit. on p. 33).

[Su18] Su, Hang; Jampani, Varun; Sun, Deqing; Maji, Subhransu; Kalogerakis, Evangelos; Yang, Ming-Hsuan and Kautz, Jan: "SplatNet: Sparse Lattice Networks for Point Cloud Processing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2530–2539 (cit. on pp. 85, 118, 156).

[Sun01] Sun, Wei; Bradley, Colin; Zhang, YF and Loh, Han Tong: "Cloud Data Modeling Employing A Unified, Non-Redundant Triangular Mesh". In: *Computer-Aided Design* 33.2 (2001), pp. 183–193 (cit. on p. 37).

[Sun20a] Sun, Zhiqing; Yu, Hongkun; Song, Xiaodan; Liu, Renjie; Yang, Yiming and Zhou, Denny: "MobileBERT: A Compact Task-Agnostic BERT for Resource-Limited Devices". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020 (cit. on p. 20).

[Sun20b] Sun, Zhiqing; Yu, Hongkun; Song, Xiaodan; Liu, Renjie; Yang, Yiming and Zhou, Denny: "TinyBERT: Distilling BERT for Natural Language Understanding". In: *Annual Conference of the Association for Computational Linguistics (ACL)*. 2020 (cit. on p. 20).

[Tat18] Tatarchenko, Maxim; Park, Jaesik; Koltun, Vladlen and Zhou, Qian-Yi: "Tangent Convolutions for Dense Prediction in 3D". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3887–3896 (cit. on p. 27).

[Te18] Te, Gusi; Hu, Wei; Zheng, Amin and Guo, Zongming: "RGCNN: Regularized Graph CNN for Point Cloud Segmentation". In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018, pp. 746–754 (cit. on p. 34).

[Tho19] Thomas, Hugues; Qi, Charles R; Deschaud, Jean-Emmanuel; Marcotegui, Beatriz; Goulette, François and Guibas, Leonidas J: "KPconv: Flexible and Deformable Convolution for Point Clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 6411–6420 (cit. on pp. 30, 31, 81, 85, 99, 114, 118, 148, 156).

[Vas17] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz and Polosukhin, Illia: "Attention Is All You Need". In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017) (cit. on pp. 8, 18, 19, 35, 74, 102).

[Vig19] Vig, Jesse: "Analyzing the Structure of Attention in a Transformer Language Model". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2019, pp. 63–76 (cit. on p. 20).

[Wan18] Wang, Chu; Samari, Babak and Siddiqi, Kaleem: "Local Spectral Graph Convolution for Point Set Feature Learning". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 52–66 (cit. on p. 34).

[Wan19] Wang, Yue; Sun, Yongbin; Liu, Ziwei; Sarma, Sanjay E; Bronstein, Michael M and Solomon, Justin M: "Dynamic Graph CNN for Learning on Point Clouds". In: *ACM Transactions on Graphics (TOG)* 38.5 (2019), pp. 1–12 (cit. on pp. 33, 34, 81, 84, 85, 99, 114, 118, 148, 156).

[Wan21] Wang, Xu; Jin, Yi; Cen, Yigang; Lang, Congyan and Li, Yidong: "PST-Net: Point Cloud Sampling via Point-Based Transformer". In: *Proceedings of the 11th International Conference on Image and Graphics (ICIG)*. Springer. 2021, pp. 57–69 (cit. on pp. 2, 40, 80, 88, 122, 160).

[Wan22] Wang, Shuaiqing; Hu, Qijun; Xiao, Dongsheng; He, Leping; Liu, Rengang; Xiang, Bo and Kong, Qinghui: "A New Point Cloud Simplification Method with Feature and Integrity Preservation by Partition Strategy". In: *Measurement* 197 (2022), p. 111173 (cit. on p. 38).

[Wan23] Wang, Xu; Jin, Yi; Cen, Yigang; Wang, Tao; Tang, Bowen and Li, Yidong: "LighTN: Light-Weight Transformer Network for Performance-Overhead Tradeoff in Point Cloud Downsampling". In: *IEEE Transactions on Multimedia (TMM)* (2023) (cit. on pp. 2, 40, 64, 80, 87, 88, 121, 122, 159, 160).

[Wie19] Wiegreffe, Sarah and Pinter, Yuval: "Attention is not not Explanation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019, pp. 11–20 (cit. on p. 20).

[Wie22] Wiersma, Ruben; Nasikun, Ahmad; Eisemann, Elmar and Hildebrandt, Klaus: "DeltaConv: Anisotropic Operators for Geometric Deep Learning on Point Clouds". In: *ACM Transactions on Graphics (TOG)* 41.4 (2022), pp. 1–10 (cit. on pp. 31, 81, 114, 118, 148, 156).

[Wu15] Wu, Zhirong; Song, Shuran; Khosla, Aditya; Yu, Fisher; Zhang, Linguang; Tang, Xiaoou and Xiao, Jianxiong: "3D ShapeNets: A Deep Representation for Volumetric Shapes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920 (cit. on pp. 57, 58, 60).

[Wu19] Wu, Wenxuan; Qi, Zhongang and Fuxin, Li: "PointConv: Deep Convolutional Networks on 3D Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9621–9630 (cit. on pp. 2, 30, 31, 38, 81, 85, 114, 117, 118, 148, 156).

[Wu20] Wu, Chengzhi; Pfrommer, Julius; Beyerer, Jürgen; Li, Kangning and Neubert, Boris: "Object Detection in 3D Point Clouds via Local Correlation-Aware Point Embedding". In: *Proceedings of the Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. IEEE. 2020, pp. 1–8 (cit. on p. 33).

[Wu22] Wu, Xiaoyang; Lao, Yixing; Jiang, Li; Liu, Xihui and Zhao, Hengshuang: "Point Transformer V2: Grouped Vector Attention and Partition-Based Pooling". In: *Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022), pp. 33330–33342 (cit. on p. 35).

[Wu23a] Wu, Chengzhi; Bi, Xuelei; Pfrommer, Julius; Cebulla, Alexander; Mangold, Simon and Beyerer, Jürgen: "Sim2real Transfer Learning for Point Cloud Segmentation: An Industrial Application Case on Autonomous Disassembly". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2023, pp. 4531–4540 (cit. on p. 35).

[Wu23b] Wu, Chengzhi; Zheng, Junwei; Pfrommer, Julius and Beyerer, Jürgen: "Attention-Based Point Cloud Edge Sampling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 5333–5343 (cit. on pp. 95, 106, 114, 118, 121, 148, 156, 159, 160).

[Wu23c] Wu, Wenxuan; Fuxin, Li and Shan, Qi: "PointConvFormer: Revenge of The Point-Based Convolution". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 21802–21813 (cit. on p. 30).

[Wu24a] Wu, Chengzhi; Fu, Hao; Kaiser, Jan-Philipp; Barczak, Erik Tabuchi; Pfrommer, Julius; Lanza, Gisela; Heizmann, Michael and Beyerer, Jürgen: "6D Pose Estimation on Point Cloud Data through Prior Knowledge Integration: A Case Study in Autonomous Disassembly". In: *Procedia CIRP* 122 (2024), pp. 193–198 (cit. on p. 35).

[Wu24b] Wu, Chengzhi; Qianliang, Huang; Kun, Jin; Pfrommer, Julius and Beyerer, Jürgen: "A Cross Branch Fusion-based Contrastive Framework for Point Cloud Self-supervised Learning". In: *Proceedings of the International Conference on 3D Vision (3DV)*. 2024 (cit. on p. 36).

[Wu24c] Wu, Chengzhi; Wang, Kaige; Zhong, Zeyun; Fu, Hao; Zheng, Junwei; Zhang, Jiaming; Pfrommer, Julius and Beyerer, Jürgen: "Rethinking Attention Module Design for Point Cloud Analysis". In: *arXiv preprint arXiv:2407.19294* (2024) (cit. on p. 36).

[Wu24d] Wu, Xiaoyang; Jiang, Li; Wang, Peng-Shuai; Liu, Zhijian; Liu, Xihui; Qiao, Yu; Ouyang, Wanli; He, Tong and Zhao, Hengshuang: "Point Transformer V3: Simpler Faster Stronger". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 4840–4851 (cit. on p. 35).

[Wu25] Wu, Chengzhi; Wan, Yuxin; Fu, Hao; Pfrommer, Julius; Zhong, Zeyun; Zheng, Junwei; Zhang, Jiaming and Beyerer, Jürgen: "SAMBLE: Shape-Specific Point Cloud Sampling for an Optimal Trade-Off Between Local Detail and Global Uniformity". In:

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2025, pp. 1342–1352 (cit. on p. 133).

[Xu18] Xu, Yifan; Fan, Tianqi; Xu, Mingye; Zeng, Long and Qiao, Yu: "SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 87–102 (cit. on pp. 31, 81, 85, 114, 118, 148, 156).

[Xu20] Xu, Qiangeng; Sun, Xudong; Wu, Cho-Ying; Wang, Panqu and Neumann, Ulrich: "Grid-GCN for Fast and Scalable Point Cloud Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5661–5670 (cit. on p. 33).

[Xu21] Xu, Mutian; Ding, Runyu; Zhao, Hengshuang and Qi, Xiaojuan: "PAconv: Position Adaptive Convolution with Dynamic Kernel Assembling on Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 3173–3182 (cit. on pp. 32, 81, 85, 114, 118, 148, 156).

[Yan19a] Yang, Jiancheng; Zhang, Qiang; Ni, Bingbing; Li, Linguo; Liu, Jinxian; Zhou, Mengdie and Tian, Qi: "Modeling Point Clouds with Self-Attention and Gumbel Subset Sampling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3323–3332 (cit. on p. 41).

[Yan19b] Yang, Zhilin; Dai, Zihang; Yang, Yiming; Carbonell, Jaime; Salakhutdinov, Russ R and Le, Quoc V: "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *Advances in Neural Information Processing Systems (NeurIPS)* 32 (2019) (cit. on p. 20).

[Yan20a] Yan, Xu; Zheng, Chaoda; Li, Zhen; Wang, Sheng and Cui, Shuguang: "PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5589–5598 (cit. on pp. 2, 41, 81, 114, 148).

[Yan20b] Yang, Dinghao and Gao, Wei: "Pointmanifold: Using Manifold Learning for Point Cloud Classification". In: *arXiv preprint arXiv:2010.07215* (2020) (cit. on p. 34).

[Yan21] Yang, Shuohang; Liu, Peng; Ye, Zihao and Lin, Jimmy: "Tiny-Transformer: A Lightweight Transformer for Neural Machine Translation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2021 (cit. on p. 20).

[Yan23] Yang, Xincheng; Jin, Mingze; He, Weiji and Chen, Qian: "Point-CAT: Cross-Attention Transformer for Point Cloud". In: *arXiv preprint arXiv:2304.03012* (2023) (cit. on pp. 26, 134, 137).

[Yi16] Yi, Li; Kim, Vladimir G; Ceylan, Duygu; Shen, I-Chao; Yan, Mengyan; Su, Hao; Lu, Cewu; Huang, Qixing; Sheffer, Alla and Guibas, Leonidas: "A Scalable Active Framework for Region Annotation in 3D Shape Collections". In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), pp. 1–12 (cit. on pp. 61, 62).

[Yos15] Yosinski, Jason; Clune, Jeff; Nguyen, Anh; Fuchs, Thomas and Lipson, Hod: "Understanding Neural Networks through Deep Visualization". In: *Proceedings of the International Conference on Machine Learning Workshops (ICMLW)*. 2015 (cit. on p. 4).

[Yu18] Yu, Lequan; Li, Xianzhi; Fu, Chi-Wing; Cohen-Or, Daniel and Heng, Pheng-Ann: "PU-Net: Point Cloud Upsampling Network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2790–2799 (cit. on p. 2).

[Yu22] Yu, Xumin; Tang, Lulu; Rao, Yongming; Huang, Tiejun; Zhou, Jie and Lu, Jiwen: "Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 19313–19322 (cit. on p. 36).

[Zha18] Zhang, Yingxue and Rabbat, Michael: "A Graph-CNN for 3D Point Cloud Classification". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 6279–6283 (cit. on p. 34).

[Zha19] Zhao, Hengshuang; Jiang, Li; Fu, Chi-Wing and Jia, Jiaya: "PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5565–5573 (cit. on p. 33).

[Zha21a] Zhang, Kuangen; Hao, Ming; Wang, Jing; Chen, Xinxing; Leng, Yuquan; Silva, Clarence W de and Fu, Chenglong: "Linked Dynamic Graph CNN: Learning through Point Cloud by Linking Hierarchical Features". In: *Proceedings of the 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. IEEE. 2021, pp. 7–12 (cit. on p. 33).

[Zha21b] Zhang, Zihan; Wang, Huan; Xue, Naiwen; Sun, Jian; Xie, Xing and Han, Wentao: "SqueezeBERT: Lightweight and Efficient Transformer for Natural Language Processing". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2021 (cit. on p. 20).

[Zha21c] Zhao, Hengshuang; Jiang, Li; Jia, Jiaya; Torr, Philip HS and Koltun, Vladlen: "Point Transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16259–16268 (cit. on pp. 23, 35, 38, 81, 85, 86, 114, 117, 118, 148, 156).

[Zha22a] Zhang, Cheng; Wan, Haocheng; Shen, Xinyi and Wu, Zizhao: "PatchFormer: An Efficient Point Transformer with Patch Attention". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 11799–11808 (cit. on pp. 35, 36).

[Zha22b] Zhang, Renrui; Guo, Ziyu; Gao, Peng; Fang, Rongyao; Zhao, Bin; Wang, Dong; Qiao, Yu and Li, Hongsheng: "Point-M2AE: Multi-Scale Masked Autoencoders for Hierarchical Point Cloud Pre-raining". In: *Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022), pp. 27061–27074 (cit. on p. 36).

[Zho18] Zhou, Yin and Tuzel, Oncel: "VoxelNet: End-to-End Learning for Point Cloud-Based 3D Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4490–4499 (cit. on p. 38).

[Zhu23] Zhu, Wei; Ying, Yue; Zhang, Jin; Wang, Xiuli and Zheng, Yayu: "Point Cloud Registration Network Based on Convolution Fusion and Attention Mechanism". In: *Neural Processing Letters* (2023), pp. 1–21 (cit. on p. 30).

Own publications

This section contains a complete list of the author's own publications that were created during the doctoral period.

[1] Wu, Chengzhi; Wan, Yuxin; Fu, Hao; Pfrommer, Julius; Zhong, Zeyun; Zheng, Junwei; Zhang, Jiaming and Beyerer, Jürgen: "SAMBLE: Shape-Specific Point Cloud Sampling for an Optimal Trade-Off Between Local Detail and Global Uniformity". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2025, pp. 1342–1352.

[2] Wu, Chengzhi; Zheng, Junwei; Pfrommer, Julius and Beyerer, Jürgen: "Attention-Based Point Cloud Edge Sampling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 5333–5343.

[3] Zheng, Junwei; Liu, Ruiping; Chen, Yufan; Peng, Kunyu; Wu, Chengzhi; Yang, Kailun; Zhang, Jiaming and Stiefelhagen, Rainer: "Open Panoramic Segmentation". In: *2024 European Conference on Computer Vision (ECCV)*. 2024.

[4] Wu, Chengzhi; Huang, Qianliang; Jin, Kun; Pfrommer, Julius and Beyerer, Jürgen: "A Cross Branch Fusion-Based Contrastive Learning Framework for Point Cloud Self-supervised Learning". In: *2024 International Conference on 3D Vision (3DV)*. IEEE. 2024, pp. 528–538.

[5] Wu, Chengzhi; Fu, Hao; Kaiser, Jan-Philipp; Barczak, Erik Tabuchi; Pfrommer, Julius; Lanza, Gisela; Heizmann, Michael and Beyerer, Jürgen: "6D Pose Estimation on Point Cloud Data through Prior Knowledge Integration: A Case Study in Autonomous Disassembly". In: *Procedia CIRP* 122 (2024), pp. 193–198.

[6] Wu, Chengzhi; Zheng, Junwei; Pfrommer, Julius and Beyerer, Jürgen: "Attention-based Part Assembly for 3D Volumetric Shape Modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023, pp. 2717–2726.

[7] Wu, Chengzhi; Pfrommer, Julius; Zhou, Mingyuan and Beyerer, Jürgen: "Self-Supervised Generative-Contrastive Learning of Multi-Modal Euclidean Input for 3D Shape Latent Representations: A Dynamic Switching Approach". In: *IEEE Transactions on Multimedia (TMM)* (2023).

[8] Wu, Chengzhi; Bi, Xuelei; Pfrommer, Julius; Cebulla, Alexander; Mangold, Simon and Beyerer, Jürgen: "Sim2real Transfer Learning for Point Cloud Segmentation: An Industrial Application Case on Autonomous Disassembly". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2023, pp. 4531–4540.

[9] Wu, Chengzhi; Qiu, Linxi; Zhou, Kanran; Pfrommer, Julius and Beyerer, Jürgen: "Synmotor: A Benchmark Suite for Object Attribute Regression and Multi-task Learning". In: *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*. 2023, pp. 529–540.

[10] Wu, Chengzhi; Zhou, Kanran; Kaiser, Jan-Philipp; Mitschke, Norbert; Klein, Jan-Felix; Pfrommer, Julius; Beyerer, Jürgen; Lanza, Gisela; Heizmann, Michael and Furmans, Kai: "MotorFactory: A Blender Add-on for Large Dataset Generation of Small Electric Motors". In: *Procedia CIRP* 106 (2022), pp. 138–143.

[11] Lanza, Gisela et al.: "Agiles Produktionssystem mittels lernender Roboter bei ungewissen Produktzuständen am Beispiel der Anlasser-Demontage". In: *at-Automatisierungstechnik* 70.6 (2022), pp. 504–516.

[12] Wu, Chengzhi; Pfrommer, Julius; Beyerer, Jürgen; Li, Kangning and Neubert, Boris: "Object Detection in 3D Point Clouds via Local Correlation-Aware Point Embedding". In: *2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. IEEE. 2020, pp. 1–8.

[13] Wu, Chengzhi: "Attention Mechanism in Computer Vision: Current Status and Prospect". In: *Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory*. 2022, pp. 207–221.

[14] Wu, Chengzhi: "Learning Universal Vector Representation for Objects of Different 3D Euclidean formats". In: *Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory*. 2021, pp. 155–170.

[15] Wu, Chengzhi: "Learning with Latent Representations of 3D Data: from Classical Methods to 3D Deep Learning". In: *Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory*. 2020, pp. 133–149.

Supervised student theses

This section contains the bachelor's or master's theses of students supervised by the author during the doctoral period.

[1] Fu, Hao: "Bin-Based Learnable Sampling Strategy for Point Cloud Edge Sampling". Master's Thesis. Karlsruhe Institute of Technology, 2024.

[2] Wang, Kaige: "A Study of Attention Module Variations in Point Cloud Feature Learning". Master's Thesis. Karlsruhe Institute of Technology, 2023.

[3] Huang, Qianliang: "Exploring Contrastive Learning on 3D Point Clouds with Cross Attention". Master's Thesis. Karlsruhe Institute of Technology, 2023.

[4] Bi, Xuelei: "Attention-based 3D Point Cloud Segmentation on Bosch Motors and Its Visualization Interface Design". Master's Thesis. Karlsruhe Institute of Technology, 2022.

[5] Zheng, Junwei: "VoxAttention: Shape Synthesis via Part Assembly with Self-Attention". Master's Thesis. Karlsruhe Institute of Technology, 2022.

[6] Qiu, Linxi: "A Novel Multi-Attribute Regression Benchmark Suite on Bosch Motors". Master's Thesis. Karlsruhe Institute of Technology, 2022.

[7] Yu, Haodong: "3D Point Cloud semantic Segmentation on BOSCH Motors". Master's Thesis. Karlsruhe Institute of Technology, 2021.

[8] Zhou, Mingyuan: "SwitchVAE: Learning Better Latent Representations from Objects of Different 3D Euclidean Formats". Master's Thesis. Karlsruhe Institute of Technology, 2021.

[9] Mehrbrodt, Iris: “Improving Stylized View Synthesis of Image-based Reconstructions using Neural Networks”. Master’s Thesis. Karlsruhe Institute of Technology, 2019.

[10] Li, Kangning: “Object Detection In 3D Point Clouds By Leveraging Local-Spatial Correlations”. Master’s Thesis. Karlsruhe Institute of Technology, 2019.

List of Figures

List of Tables

Listings

Acronyms

APES	Attention-based Point cloud Edge Sampling
CE	Cross-entropy
CNN	Convolutional Neural Network
FLOPs	floating-point operations per second
FPS	Farthest Point Sampling
IDIS	Inverse Density Importance Sampling
IoU	Intersection-over-Union
***k*NN**	*k*-Nearest Neighbor
LReLU	Leaky Rectified Linear Units
mIoU	mean Intersection-over-Union
MLP	Multi-Layer Perceptron
N2P	neighbor-to-point
NLP	Natural Language Processing
OA	Overall Accuracy

P2P	point-to-point
ReLU	Rectified Linear Units
RS	Random Sampling
SAM	Sparse Attention Map
SAMBLE	Sparse Attention Map and Bin-based Learning
SAMPS	Sparse Attention Map-based Point cloud Sampling
SOTA	state-of-the-art
VRAM	Video Random-Access Memory