UNIVERSITY
OF TECHNOLOGY
SYDNEY

High-Performance and Interpretable 3D Point Cloud Analysis

by Tuo Feng

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Dr. Baihe Ma
and Prof. XiaoJun Chang

University of Technology Sydney
Faculty of Engineering and Information Technology

August 2025

Certificate of Original Authorship

I, **Tuo Feng**, declare that this thesis is submitted in fulfilment of the requirements for the award of **Doctor of Philosophy**, in the **Faculty of Engineering and Information Technology** at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: 18/08/2025

High-Performance and Interpretable 3D Point Cloud Analysis

by

Tuo Feng

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

The field of 3D computer vision has seen significant advancements, driven by the increasing availability of high-quality 3D data from Light Detection and Ranging (LiDAR) sensors, Red-Green-Blue with Depth (RGB-D) cameras, and other modality sensors. However, key challenges persist in efficiently processing and understanding large-scale, dynamic, and complex 3D environments. Problems such as inefficient feature extraction, poor scalability in clustering and subclass pattern discovery, computational overhead in large-scale scene processing, and the lack of model interpretability hinder the effectiveness of existing methods.

To address these challenges, this thesis introduces a series of novel methods spanning clustering-based learning, Dynamic Sparse Training (DST), large-kernel architectures, and interpretable model design. First, we propose a Clustering-based 3D Point Cloud Representation Learning Method (Cluster3D) that uncovers fine-grained patterns in 3D point clouds, enhancing representation learning in supervised settings. Second, we develop an Effective and Efficient 3D Perception Method with Large Sparse Kernels (LSK3DNet), a 3D backbone leveraging Spatial-wise Dynamic Sparsity (SDS) and Channel-wise Weight Selection (CWS) to improve computational efficiency and segmentation accuracy. Third, we introduce an Ad-Hoc Interpretable Classifier for 3D Point Clouds (Interpretable3D). It is a prototype-based interpretable classifier that integrates interpretability directly into its architecture, providing transparent decision-making processes. Lastly, we propose a Novel 3D Scene Representation Learning Method (Shape2Scene), a scalable pretraining strategy that bridges shape-level understanding with scene-level downstream tasks, demonstrating improved transfer learning capabilities.

Experimental results across multiple benchmarks validate the effectiveness of the proposed methods. Cluster3D achieves state-of-the-art performance in static and dynamic point cloud segmentation tasks. LSK3DNet demonstrates significant gains in semantic segmentation and object detection tasks while reducing computational costs. Interpretable3D not only maintains competitive accuracy in shape classification and part segmentation but also provides transparent and interpretable predictions. Shape2Scene outperforms existing pretraining strategies in both shape-level and scene-level tasks, highlighting its scalability and adaptability.

In conclusion, the contributions of this thesis collectively advance the state of the art in 3D computer vision by addressing critical problems in feature extraction, clustering, efficiency, and interpretability. The proposed methodologies pave the way for more robust, efficient, and interpretable 3D vision systems, enabling their application in real-world scenarios such as autonomous driving, robotics, and augmented reality.

Acknowledgements

I am deeply grateful to Prof. Yi Yang and Prof. Wenguan Wang for their invaluable guidance, encouragement, and prompt feedback throughout my time as their student. Their genuine concern for my progress has been a tremendous source of motivation.

I also wish to express my appreciation to every member of our research group who supported me in both my academic pursuits and day-to-day life. In particular, I would like to acknowledge Dr. Baihe Ma, Prof. Xiaojun Chang, and Prof. Linchao Zhu for their insightful suggestions that significantly shaped my research.

I appreciate the financial support from the CSC-UTS Program which was awarded to me in 2021. Without the scholarship support, this thesis would never have been finished.

Tuo FENG
Sydney, Australia, 2025.

Contents

List of Figures

List of Tables

Acronyms & Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AP	Average Precision
BEV	Bird’s-Eye View
BERT	Bidirectional Encoder Representations from Transformers
CE Loss	Cross-Entropy Loss
CNN	Convolutional Neural Network
CNNs	Convolutional Neural Networks
CWS	Channel-wise Weight Selection
Cluster3D	a Clustering-based 3D Point Cloud Representation Learning Method
DNNs	Deep Neural Networks
DST	Dynamic Sparse Training
ER	Erdős-Rényi
ERFs	Effective Receptive Fields
FLOPs	Floating Point Operations
FPS	Frames Per Second

GCN	Graph Convolutional Network
IoU	Intersection over Union
Interpretable3D	an Ad-Hoc Interpretable Classifier for 3D Point Clouds
KITTI	a Dataset for Autonomous Driving derived from the KITTI Vision Benchmark
KL	Kullback-Leibler Divergence
LiDAR	Light Detection and Ranging
LSK3DNet	an Effective and Efficient 3D Perception Method with Large Sparse Kernels
LVQ	Learning Vector Quantization
MAE	Masked Autoencoder
mAcc	Mean Accuracy
mAP	Mean Average Precision
mIoU	Mean Intersection over Union
MH-P	Multi-scale High-resolution Point-based Backbone
MH-V	Multi-scale High-resolution Voxel-based Backbone
MLPs	Multi-Layer Perceptrons
NAS	Neural Architecture Search
PPC	Point-Point Contrastive Loss
PCC	Point-Center Contrastive Loss
PTV1	Point Transformer v1
RGB-D	Red-Green-Blue with Depth
Shape2Scene	a Novel 3D Scene Representation Learning Method
S2SS	Shape-to-Scene Strategy

SDS	Spatial-wise Dynamic Sparsity
ScanNet	Richly-annotated 3D Reconstructions of Indoor Scenes
SemanticKITTI	a Dataset for Semantic Scene Understanding using LiDAR Sequences
ShapeNet	an Information-Rich 3D Model Repository
S3DIS	Stanford Large-Scale 3D Indoor Space Dataset
Synthia4D	a Large Synthetic Dataset for Autonomous Driving Scenarios
Sinkhorn-Knopp	an Iterative Algorithm for Optimal Transport Problems
SSL	Self-Supervised Learning
TTA	Test Time Augmentation
SOTA	state of the art
t-SNE	t-distributed Stochastic Neighbor Embedding
WTA	Winner-Takes-All

Nomenclature

Italic uppercase Latin letters	represent sets.
Bold lowercase letters	denote vectors and features
Bold uppercase letters	signify matrices.
Uppercase letters	indicate scalars.
$(\cdot)^T$	denotes the transpose operation.
$\langle\cdot,\cdot\rangle$	is cosine similarity.
$\langle\cdot\rangle_F$	is the Frobenius dot-product.
$\odot$	represents the Hadamard product.
$\mathbb{R}$	denotes the set of real numbers.

List of Publications

Related to the Thesis:

- **Tuo Feng**, Wenguan Wang, Xiaohan Wang, YiYang, and Qinghua Zheng. Clustering based point cloud representation learning for 3d analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8283–8294, 2023.
- **Tuo Feng**, Ruijie Quan, Xiaohan Wang, Wenguan Wang, and Yi Yang. Interpretable3d: An ad-hoc interpretable classifier for 3d point clouds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 1761–1769, 2024.
- **Tuo Feng**, Wenguan Wang, Fan Ma, and Yi Yang. Lsk3dnet: Towards effective and efficient 3d perception with large sparse kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14916–14927, 2024.
- **Tuo Feng**, Wenguan Wang, Ruijie Quan, and Yi Yang. Shape2scene: 3d scene representation learning through pre-training on shape data. In *Proceedings of European Conference on Computer Vision*. Springer, 2024.

Others:

- **Tuo Feng**, Wenguan Wang, and Yi Yang. Gaussian-based World Model: Gaussian Priors for Voxel-Based Occupancy Prediction and Future Motion Prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.
- **Tuo Feng**, Wenguan Wang, Ruijie Quan, Yi Yang. Clustering based Supervised Training for 3D Perception. Submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.
- **Tuo Feng**, Wenguan Wang, and Yi Yang. A survey of world models for autonomous driving. *arXiv preprint arXiv:2501.11260*, 2025.

Chapter 1

Introduction

1.1 Background

The rapid growth of 3D computer vision research has enabled a wide range of applications, from autonomous driving to robotics and augmented reality. With the advent of LiDAR, RGB-D sensors, and other 3D data acquisition methods, researchers now have access to massive, high-quality 3D datasets. However, the intrinsic properties of 3D data – such as sparsity, irregularity, and high dimensionality – introduce challenges in processing, understanding, and utilizing the data effectively.

2D computer vision methods often fail to generalize to 3D vision due to the fundamental differences between 2D images and 3D point clouds or voxel grids. This discrepancy has driven the development of novel algorithms and architectures designed specifically for 3D data, ranging from deep learning methods for segmentation and detection to clustering and representation learning strategies. Despite these advancements, numerous challenges remain, particularly in handling large-scale and dynamic environments, ensuring model interpretability, and maintaining computational efficiency.

1.2 Research Challenges

1.2.1 Challenges on Feature Extraction

Feature extraction forms the backbone of 3D vision tasks, yet it is fraught with challenges due to the irregular structure and sparsity of 3D data. Point-based methods, such as PointNet [14] and PointNet++ [15], directly operate on raw point clouds using permutation-invariant functions, but they often struggle to capture fine-grained geometric details in large-scale scenes. On the other hand, projection-based and voxel-based methods can leverage 2D or 3D convolutions to process structured representations of point clouds but are prone to information loss and computational inefficiencies.

The high dimensionality of 3D data further complicates feature extraction. Efficiently encoding both local and global geometric information remains a critical bottleneck. While attention-based and graph-based methods have shown promise in capturing contextual relationships, they come at the cost of increased computational overhead, which hinders their applicability to real-time systems.

1.2.2 Challenges on Clustering and Subclass Pattern Discovery

Clustering and pattern discovery are vital for understanding the latent structures within 3D data. However, the high dimensionality and sparsity of point clouds pose significant challenges for clustering algorithms. Most existing approaches rely on manual labeling and predefined class structures, which limits their ability to uncover fine-grained patterns or intra-class variations.

Our Cluster3D [16] aims to address this by leveraging clustering-based representation learning to discover unknown subclasses and refine feature spaces. Nonetheless, achieving accurate and efficient clustering in large-scale datasets remains an open problem, particularly when balancing computational feasibility with the need for precise subclass delineation.

1.2.3 Challenges on Large-scale Scene Processing

Processing large-scale 3D scenes, such as urban environments or complex indoor layouts [1, 9, 10], requires balancing computational efficiency and accuracy. Sparse convolutional networks [6, 17] have shown promise in handling large-scale data by focusing computations on non-empty regions.

However, these methods often rely on aggressive downsampling, which can degrade performance in tasks requiring high-resolution outputs.

Furthermore, dynamic scenes introduce additional complexities. Incorporating temporal information into segmentation and detection tasks demands models capable of fusing spatial and temporal data effectively. While early fusion and late fusion strategies [18–27] have been proposed, both approaches face limitations in scalability and generalization.

1.2.4 Challenges on Model Interpretability

The lack of interpretability in deep learning models is a pervasive issue across domains, and 3D vision is no exception. Most 3D models operate as black boxes, making it difficult to understand the rationale behind their predictions. Post-hoc interpretability methods, such as activation maximization [28] and saliency/attention maps [29–32], attempt to address this but often fail to provide reliable or consistent explanations.

Interpretable3D [33] takes a more proactive approach by integrating interpretability into the model design. This model relies on prototype-based mechanisms to align decisions with representative data points, offering a more transparent decision-making process. However, developing interpretable models that maintain competitive performance remains a significant challenge.

1.2.5 Summary

Addressing these challenges calls for a combination of innovative algorithms, efficient architectures, and principled training strategies. In this thesis, we pursue four complementary directions: (i) clustering-based representation learning (*i.e.*, Cluster3D [16]) to discover intra-class structure; (ii) large-kernel 3D backbones (*i.e.*, LSK3DNet [34]) to scale feature extraction and processing to large, dynamic scenes; (iii) built-in interpretability (*i.e.*, Interpretable3D [33]) via prototype-based decision mechanisms; and (iv) scalable pre-training (*i.e.*, Shape2Scene [35]) to improve transferability under data scarcity and heterogeneity. Together, these directions directly correspond to the challenges outlined and provide a cohesive path toward robust, efficient, and interpretable 3D vision.

1.3 Research Significance and Goals

The primary goal of this research is to advance the state of the art in 3D vision by addressing key challenges in feature extraction, clustering, large-scale processing, and interpretability. To achieve this, the thesis explores a range of methodologies, including:

- Clustering-based approaches for discovering fine-grained patterns in large-scale 3D datasets.
- Large-kernel and sparse convolutional methods to improve the efficiency and accuracy of 3D backbones.
- Interpretable models that integrate transparency into their design without sacrificing performance.
- Scalable pre-training strategies that enable shape-to-scene transfer learning.

These efforts aim to enhance the efficiency, accuracy, and interpretability of 3D vision systems while providing insights into the underlying structure of 3D data. The following contributions summarize the specific methods developed in this thesis to achieve these goals.

1.4 Research Contributions

This thesis makes the following key contributions:

1. Cluster3D: A clustering-based representation learning framework that uncovers fine-grained patterns in large-scale point clouds.
2. LSK3DNet: A novel 3D backbone that leverages spatial-wise dynamic sparsity and channel-wise weight selection to improve performance in large-scale scenes.
3. Interpretable3D: A prototype-based interpretable model that enhances transparency and decision-making in 3D classification tasks.
4. Shape2Scene: A multi-scale pre-training strategy that supports shape-level tasks and enables better generalization to scene-level downstream tasks.

1.5 Thesis Organization

The thesis is organized as follows:

- Chapter 2: A review of related work in 3D vision, including segmentation, clustering, representation learning, and interpretability.
- Chapter 3: An introduction to Cluster3D, detailing its clustering-based approach and applications in static and dynamic environments.
- Chapter 4: The proposal of LSK3DNet, a 3D backbone utilizing large kernels and dynamic sparse training.
- Chapter 5: A discussion on Interpretable3D, an interpretable classification framework for 3D perception tasks.
- Chapter 6: A description of Shape2Scene, a shape-to-scene pre-training method for scalable 3D understanding.
- Chapter 7: A summary of contributions and future directions.

Chapter 2

Literature Review

2.1 3D Backbones

3D Backbones. The design of backbones for 3D point cloud understanding has evolved along two main paradigms: **i)** *Point-based* methods [33, 36–38] directly operate on raw point sets in a permutation-invariant manner, as pioneered by PointNet/PointNet++ [14, 15]. Subsequent efforts have explored local feature pooling [39–48], graph convolutions [45, 49–57], kernel-based operations [58–65], and attention-based aggregation [7, 25, 66–68]. While these point-based models preserve fine-grained geometric and semantic information, they often face scalability challenges in large urban environments [69]. **ii)** *Projection-based* methods [70–73] map unstructured points to regular 2D grids [70, 73–79] or 3D voxels [6, 17, 71, 80–86], allowing standard 2D/3D convolutions. However, 2D projections risk discarding crucial geometric cues and demand costly back-projections, while voxel-based approaches suffer from heavy computational/memory overhead and performance degradation due to downsampling [87, 88]. *Voxel-based* backbones [6, 18, 81, 82, 89, 90] with sparse convolution [18, 82] restrict computation to non-empty voxels, improving efficiency and enabling high-performance 3D semantic segmentation [6, 89]. Recently, *fusion-based* methods [87, 91, 92] have integrated voxel- and point-based representations. The point-based high-resolution branch mitigates the performance degradation, which is caused by aggressive downsampling [87, 88] (*i.e.*, regular sparse convolution) of the voxel branch.

The 3D backbones diverge in network structure for shape-level and scene-level tasks. Networks for shape classification commonly employ successive downsampling to acquire high-level semantic features while maintaining a lower resolution. Conversely, for extracting deeper features with higher resolution, well-known networks for scene-level tasks often incorporate architectures like U-Net or hourglass-like networks [15, 18, 37–39, 70, 71, 81, 82, 89, 90]. For contrastive methods [93, 94] and multi-modal methods [95, 96], backbones originally designed for shape classification tasks are employed during pre-training. However, it has been demonstrated that the features extracted by these low-resolution backbones are not suitable for tasks that require high resolution representations [97]. Directly utilizing high-resolution shallow features for region and point level tasks, however, does not yield satisfactory results [97].

MH-P/V [35] in Shape2Scene present a multi-scale, high-resolution mechanism that preserves high-resolution features along with high-level semantic information. This approach improves the applicability of pre-trained high-level semantic features, making them more suitable for point-level tasks. This also ensures the preservation and mutual refinement of features across a wide range of scales, each offering distinct levels of abstraction for the point cloud. Retaining all these scale-specific features enhances diversity, and their complementary nature enables effective mutual refinement.

2.2 3D Perception

2.2.1 Static Point Cloud Segmentation

Generally speaking, current single-scan point cloud segmentation algorithms fall into two main categories based on data representation: **i)** *Projection-based* methods initially convert unstructured point sets into either a regular 2D grid [74, 75, 79] or a 3D voxel structure [6, 34, 82, 87, 92, 98], enabling the use of standard 2D/3D convolution operations. Although effective, 2D projection-based methods tend to disregard crucial geometric cues and require expensive back-projection steps, while voxel-based methods often incur heavy computational and memory overhead. **ii)** *Point-based* approaches, pioneered by PointNet++ [15], directly process point clouds through shared MLPs coupled with symmetric aggregation functions. Subsequent research has further enhanced this paradigm by exploring a variety of local aggregators, including 1) local feature pooling [38, 43, 44, 47], 2) graph convolution [52–54, 56, 57], 3) kernel-based convolution [59, 62–65], and 4) attention-based aggregation [7, 25, 67, 99, 100]. Compared to projection-based techniques, point-based methods excel at preserving point-wise semantics and capturing local geometric structures. Regrettably, many of these point-based solutions rely on point sampling strategies that are computationally expensive, memory-inefficient, and time-consuming [47], resulting in suboptimal performance in large-scale, urban environments [69].

2.2.2 Dynamic Point Cloud Segmentation

4D semantic segmentation presents significant challenges due to the spatial irregularity and temporal sequence of point cloud videos. Existing methods for segmenting dynamic point clouds can generally be classified into two main groups based on their spatial-temporal information fusion strategy: **i)** *Early fusion based* methods [18–21] directly handle point cloud sequences by adapting standard convolution to the heterogeneous characteristics of point clouds in spatial and temporal domains. **ii)** *Late fusion based* methods [22–27] typically extend existing single-scan point cloud processing models designed for spatial information extraction. They focus on integrating temporal information to enhance static features, thereby enhancing segmentation performance.

Despite their dazzling network designs, existing static and dynamic point cloud segmentation models typically adhere to a *scene-wise* training protocol. This methodology treats each point data as an independent training sample and accumulates all the point classification errors within each scene for network parameter optimization. Consequently, these models ignore the intricate relations between points across different scenes, and fail to regularize the feature embedding space from a comprehensive view. In contrast, Cluster3D [16] leverages automatic *class-wise* data clustering to capture the latent structure of the entire training dataset. This approach is based on the fundamental insight that meaningful latent data structures, such as subclass semantics, fine-grained patterns, and intra-class variation modes, are consistent and prevalent across scenes. As we will demonstrate, representations learned in this manner are well-suited for detailed analysis of point clouds.

2.2.3 Point Cloud Detection

Point cloud detection methods currently available can be categorized into two classes: **i)** *Projection-based methods* involve the transformation of 3D data points into either 2D grid (image [101, 102], spherical view [75, 103], cylindrical view [104, 105], and Bird's-Eye View (BEV) [106]) or 3D voxel [89, 107–110]. The method of 2D grid undergoes standard 2D processing and regression to derive 3D bounding boxes. Yet, detecting occluded objects in 2D planes poses challenges, and the 2D grid representations struggle to accurately preserve object dimensions. Conversely, 3D voxels resemble image pixels but in a 3D format, explicitly providing depth information. However, the voxel representation's reliance on 3D convolution escalates computational costs and demands a trade-off between resolution and efficiency. **ii)** *Point-based methods* [36, 111–114] directly handle point clouds, retaining vital spatial information. Nevertheless, these detectors are constrained by memory usage and computational expenses. Similar to point cloud segmentation, current methods primarily concentrate on network architecture and module design. They treat each annotated instance (or bounding box) as an training sample. However, these methods also tend to disregard intra-class variations for instances, stemming from factors like depth, occlusion, viewpoint, shape, *etc*. Furthermore, they overlook the intricate relationships between instances across different scenes and struggle to effectively regularize the instance embedding space from a holistic standpoint. Similar to our findings in segmentation, we reassess the impact of intra-class variation at the instance level in point cloud detection. Specifically, Cluster3D involves clustering and optimization techniques to create a robust embedding space accommodating diverse instance variations.

2.3 Self-supervised Representation Learning and Clustering

2.3.1 Self-supervised Representation Learning

Self-Supervised Learning (SSL) paradigms have substantially advanced point cloud understanding without costly annotations, as evidenced by extensive research [13, 110, 115–118]. A prominent approach builds upon the *instance discrimination* task [119, 120], where methods employ noise contrastive estimation [121] – a specialized form of contrastive learning [122, 123] – to enhance 2D/3D representations through feature comparisons. Notable implementations include

PointContrast [13], which leverages multi-viewpoint features, and DepthContrast [118] that uses augmented depth maps for improved instance discrimination and global feature extraction. These approaches have demonstrated effectiveness in dense representation learning [53, 124, 125] and scene understanding. However, pure instance discrimination methods often lack semantic structure despite their success. This limitation persists even as SSL frameworks expand to include diverse strategies [16, 40, 126–132]. The continued focus on contrastive learning paradigms highlights both their potential for annotation-free representation learning and the need to address their structural limitations through complementary approaches.

Generative SSL frameworks based on masked point modeling have emerged as a key research direction, with notable methods including Point-BERT [133], Point-MAE [134], and Point-M2AE [135]. These approaches adopt encoder-decoder architectures inspired by BERT [136] and MAE [137], employing masked region prediction to learn rich 3D representations. Three distinctive implementations demonstrate this paradigm: 1) Point-BERT combines BERT-style pre-training with dVAE concepts [138] for masked region prediction; 2) Point-MAE and Point-M2AE leverage MAE-inspired frameworks, with the latter implementing hierarchical transformers for multi-scale pre-training; 3) PointGPT [139] addresses shape leakage through auto-regressive pre-training. Additional innovations include OcCo [117], which focuses on reconstructing occluded point clouds from partial camera views. The collective success of these frameworks [140–147] underscores masked modeling's effectiveness in balancing structural awareness with state-of-the-art performance across diverse 3D understanding tasks.

Pre-training data for generative methods typically involves 3D shape data. Conversely, previous SSL methods for 3D scene understanding either utilize pre-training data derived from 3D scene data [13, 148] or consider 3D scene data as an indispensable component of pre-training [118]. In the realm of 3D scene understanding, our Shape2Scene fundamentally distinguishes itself from various aforementioned methods. Compared to scene-level methods [13, 118, 148], our shape-level pre-training approaches, *i.e.*, Shape2Scene, achieve comparable or superior performance on downstream scene tasks [35]. Despite attempts by 4DContrast [149] to generate pseudo-scenes by aggregating objects with ScanNet (scene), our Shape-to-Scene Strategy (S2SS) fundamentally differs from it, as we aggregate shapes independently of scene data. In summary, previous studies [13, 118, 148, 149] inevitably relied on substantial amounts of scene data (ScanNet v2) for pre-training.

2.3.2 Clustering-based Representation Learning

Another line of research moves beyond instance discrimination to discriminate between groups of data samples with similar features rather than individual instances [120, 140–146], jointly performing unsupervised representation learning and clustering. In this vein, Cluster3D [16] leverages clustering to automatically discover unknown subclasses within coarse-grained class labels, akin to self-supervised approaches that extract meaningful features from unlabeled data. By probing the underlying structure of large-scale point sets, Cluster3D reveals fine-grained patterns beneath manually-labeled, high-level semantic classes. Crucially, it reinforces the standard supervised paradigm for point/instance recognition with a clustering-based representation learning component, thereby regularizing the feature space to respect the inherent structure of the data. This marks one of the first attempts to explore automatic fine-grained pattern mining within a fully supervised setting for point cloud perception [16]. Moreover, by analyzing entire datasets instead of isolated scenes, clustering-based methods provide a holistic view, refining representation spaces for better segmentation.

2.4 Large-Kernel Models and Dynamic Sparse Training

2.4.1 Large-Kernel 2D/3D Models

In the 2010s, various large-kernel settings were investigated. LR-Net [150], Inceptions [151], and GCNs [152] explored 2D large kernels of 7×7, 11×11, and 15×15 respectively. Due to the widespread adoption [153, 154] of VGG [155], research into large kernels was largely overlooked in favour of multiple smaller kernels (1×1 or 3×3) to obtain a larger receptive field [156–158] during the past decade. Recently, certain studies have reintroduced large kernels in Convolutional Neural Networks (CNNs). RepLKNet [159] examines the effects of large kernels in CNNs, and for the first time it is able to increase kernel size to 31×31. It achieves comparable results to those of Swin Transformer [160]. Following *"use sparse groups, expand more"*, SLaK [161] has achieved an impressive kernel size of 51×51. SLaK has to strike a trade-off between sparsity and model width since increased model width leads to an increase in model size, a phenomenon that becomes even more pronounced in 3D vision. However, CWS effectively decouples the two objectives of improved performance and reduced model size, enabling expanded width without increasing the model size. Inspired by 2D large-kernel successes [159, 161], recent methods aim to overcome these barriers in 3D, striking a balance between kernel size, efficiency, and representational capacity. LargeKernel3D [3] has demonstrated that sizeable kernels can be successfully employed and bring positive results for 3D networks. *Spatial-wise Group Convolution* [3] enables to achieve a kernel size of $7\times7\times7$. However, it shares the weights within each spatial group during training, leading to redundant model weights. Moreover, the performance of LargeKernel3D drops when scaling up the kernel size over $7\times7\times7$.

2.4.2 Dynamic Sparse Training

Dynamic Sparse Training (DST) can train sparse neural networks from scratch, resulting in both a speedy training and prediction procedure. During training, DST [162–170] alters the location of non-zero weights with per pre-defined rules, thereby creating a sparse representation and cutting down the number of calculations. This paradigm commences without prior knowledge and simultaneously refines the non-zero locations and weights. The attractive aspect of DST is that it is sparse from the beginning, resulting in lower FLOPs and memory consumption for training and inference compared to a dense model. Generally, the pruning process [171] can be completed either by threshold-based pruning [162, 163, 167, 172] or by magnitude-based pruning [164, 173]. In addition, new weights are regrown with randomness growing [163, 167, 172], momentum growing [164], and gradient-based growing [165, 166, 173, 174]. Applied to large-kernel 3D networks, DST explores a broader parameter space without increasing model size, enabling expanded width and improved performance. Such synergy between DST and large kernels helps scale 3D models effectively [168, 170, 175].

2.5 Interpretable 3D Methods

Post-hoc *vs.* Ad-hoc Interpretability. In 3D vision, existing research of *post-hoc* analysis includes activation maximization [28] and saliency/attention maps [29–32]. However, *post-hoc* explanations are problematic and misleading [176–178] for the following reasons: ***i)*** *Post-hoc* analysis requires a separate modeling effort, which is not completely faithful to the original model, with unknown nuances [179]. Such a nuance makes it hard to guarantee the correctness of their interpretations [180]. ***ii)*** *Post-hoc* explanations can vary depending on the chosen explanation models [181]. This can lead to numerous conflicting yet seemingly convincing explanations for the same classification decision [181], none of which may actually be the correct reason for the classification [176, 182]. ***iii)*** *Post-hoc* explanations cannot provide a reasoning process for the network's decision-making [181, 183, 184]. For instance, saliency maps cannot explain how the highlighted pixels are used [179]. ***iv)*** *Post-hoc* methods may produce explanations that are not interpretable to humans, necessitating extra modeling to ensure understandability [181, 185]. In a sense, *post-hoc* explainability methods are often regarded as an excuse to deploy black-box models [186, 187], explaining and losing accuracy. Whereas an *ad-hoc* interpretable model does not. Interpretability should promote accuracy and not the other way around.

Interpretable3D [33] exemplifies *ad-hoc* interpretable models that intrinsically integrate interpretability into black-box architectures through prototype-based reasoning. Unlike opaque parametric classifiers, it defines prototypes as representative class centers [110, 188, 189] within the original data space, establishing transparent anchors for decision-making that mirror human-interpretable observations. This design enables three key advantages: (1) self-explanatory capabilities without requiring *post-hoc* analysis, (2) instance-based reasoning that enhances trustworthiness by revealing how representations influence predictions [176, 190], and (3) competitive accuracy comparable to softmax-based DNN models. By maintaining semantic alignment between prototypes and raw data features, the framework facilitates meaningful domain expert collaboration while addressing the interpretability limitations of conventional black-box approaches.

2.6 Towards Comprehensive and Interpretable 3D Understanding

Past research primarily focused on scene-wise supervised training, leaving inter-scene relationships and latent data structures underexplored. By incorporating clustering-based strategies (Cluster3D [16]), large-kernel 3D backbones [3], DST techniques, and *ad-hoc* interpretability frameworks (Interpretable3D [33]), the community can achieve a more holistic, scalable, and understandable 3D vision system. Additionally, emerging self-supervised and generative methods [133–135, 139] empower models with stronger representations transferable across tasks and domains. By uniting these diverse research threads, we drive progress toward more robust, efficient, and interpretable 3D understanding.

2.7 Positioning This Thesis within the 3D Vision Landscape

TABLE 2.1: **Design choices** in related works *vs* **this thesis**.

Methods	**Training Paradigm**	**Objective & Loss**	**Pre-training Data**	**Efficiency Details**	**Interpretability**	**Typical Works**
Point-based (*e.g.*, PointNet++, PTV1)	Scene-wise or Shape-wise	CE Loss	–	–	None (no built-in interpretability)	[7, 15, 36–38]
Voxel-based + SparseConv (*e.g.*, SPVCNN, MinkowskiNet)	Scene-wise	CE Loss	–	Sparse Convolution	None	[6, 18, 87, 89]
Scene-level SSL (*e.g.*, PointContrast, PointClustering)	Scene-wise	Contrastive Loss	Scene Level	–	None	[13, 118, 148]
LargeKernel3D (*e.g.*, Spatial-wise Group Conv)	Scene-wise	CE Loss	–	Large kernel (*e.g.*, $7\times7\times7$) with spatial-wise groups	None	[3]
This Thesis						
Cluster3D (*this thesis*)	Dataset-level	CE Loss + PPC + PCC + Lovász Loss [191]	–	–	–	[16]
LSK3DNet (*this thesis*)	Scene-wise	CE Loss + Lovász loss [191]	–	Large kernel with SDS and CWS	–	[34]
Interpretable3D (*this thesis*)	Shape-wise	CE Loss	–	–	Ad-hoc prototype-based reasoning (built-in)	[33]
Shape2Scene (*this thesis*)	Dataset-level	CE Loss + PPC	Shape $\rightarrow$ Pseudo-Scene	–	–	[35]

Compared to conventional *scene-level* training paradigms, this thesis advances four complementary research directions: (i) dataset-level structural alignment through Cluster3D [16], (ii) scalable large-kernel backbones with SDS and CWS (*i.e.*, LSK3DNet [34]), (iii) native, ad-hoc interpretability via prototype-based reasoning (*i.e.*, Interpretable3D [33]), and (iv) scalable pre-training from shapes to scenes (*i.e.*, Shape2Scene [35]) capturing multi-scale and high-resolution representations. Together, these components form a comprehensive and interpretable 3D understanding framework, as elaborated in Chapters 3–6.

Chapter 3

Clustering based Point Cloud Representation Learning for 3D Analysis

3.1 Introduction

During the last few years, point cloud segmentation has attracted increasing research effort, due to its wide applications in autonomous driving, intelligent robotics, airborne laser scanning, and virtual reality. In particular, the advances in deep learning significantly pushed forward the state-of-the-art in this field. Applying standard neural networks which are specialized for grid-like data, such as natural images, to point clouds is nontrivial, as point data are unorganized and irregular. To adapt neural networks to the geometries of point data, considerable effort has been made and representative achievements include: i) *projection-/voxel-based networks* [18, 70, 71, 74, 75, 78, 79, 82, 85, 192, 193] that project irregular point clouds to regular representations, so that mature 2D/3D convolution can be applied for segmentation; and ii) *point-based networks* [6, 43, 44, 59, 194] that ingest raw point clouds directly, by using permutation-invariant operator [14, 15, 47, 63, 67], graph convolution [52], customized convolution [64, 195, 196], or self-attention (Transformer) based architecture [7, 25, 68].

Nevertheless, the challenges in point cloud segmentation stem not only from the intrinsic non-Euclidean nature of point data, but also from the large intra-class variations caused by depth, occlusion, viewpoint, shape, *etc*. Despite various fancy point structure-aware network designs and their encouraging results, a fundamental issue was long ignored: *how to learn a good point embedding space that is discriminative for semantic categorization yet robust for point data variations?*

Mitigating this issue demands a powerful learning regime that is aware of latent variation modes (or representative fine-grained patterns) – comprehensively describing the potential structure of point data. However, in practice, it is infeasible to precisely annotate, or even roughly identify, the underlying data patterns in point clouds. This may be the reason behind the common choice that point cloud segmentation is learned as point-wise classification; any fine-grained patterns that the point data may possess are left to be 'mysteriously' learned through the supervision from high-level semantic tags.

These novel insights motivate us to devise a clustering analysis based training scheme for point cloud segmentation. It complements the standard supervised learning of point-wise classification with unsupervised clustering and regularization of the feature space. Specifically, clustering is conducted inside each labeled semantic class to automatically discover informative yet hidden subclass patterns without explicit annotation. The discovered subclass patterns essentially capture the underlying fine-grained distribution of the whole training dataset. They are then used to reshape the point embedding space, achieved by explicitly inspiring inter-subclass/-cluster discriminativeness, and reducing intra-subclass/-cluster variation. Such regularized representation space in turn facilitates the discovery of typical within-class variation modes, and benefits point recognition eventually.

Our learning algorithm enjoys several appealing advantanges: **First**, it raises a *dataset-level context-aware* training strategy. Unlike the current *de-facto*, scene-wise training paradigm, our algorithm groups point features across training scenes, and conducts clustering based representation learning. By probing the global data distribution, our algorithm encourages the highly flexible feature space to be discretized into a few distinct subcluster centers, easing the difficulty of the final semantic classification. **Second**, it is *efficient* for large-scale point cloud training. To avoid time-consuming clustering of massive point data, we opt the Sinkhorn-Knopp algorithm [197, 198] that solves cluster assignment using fast matrix-vector algebra [144]. Moreover, to follow closely the drifting representation during network training, a momentum update strategy is adopted for online approximation of the subcluster centers. **Third**, it is *principled* enough to be seamlessly incorporated into the training process of any modern point cloud segmentation networks, without bringing extra computation burden or model parameters during inference.

For thorough evaluation, we approach our training algorithm on four remarkable point cloud segmentation models, *i.e.*, Cylinder3D [6] (*voxel*-based), KPConv [64] (*point*-based), Point Transformer v1 (PTV1) [7] (*Transformer*-based), SPVNAS [87] (*neural architecture search* (NAS) based), and conduct experiments on 3D point cloud segmentation for urban scenes (*i.e.*, SemanticKITTI [1] single-scan) and indoor environments (*i.e.*, S3DIS [2]) as well as 4D segmentation of point cloud sequences (*i.e.*, SemanticKITTI [1] multi-scan). Results show that our algorithm owns **2.2-2.6**%, **1.9-2.2**%, **1.8**%, and **2.0**% mIoU gains over Cylinder3D, KPConv, PTV1, and SPVNAS, respectively. Our algorithm even promotes Second [89] and PointPillar [102] by **2.7-3.4**% and **2.0-2.2**% mAP on KITTI [8], verifying its high generality.

3.2 Methodology

3.2.1 Problem Statement and Algorithm Overview

In the context of *fully supervised* learning of point cloud segmentation, current common practice is to learn a point recognition network from a training dataset $\{\mathcal{P}^k, \mathcal{L}^k\}_k$. Here $\mathcal{P}^k = \{p_n^k \in \mathbb{R}^{3+x}\}_{n=1}^N$ is the k-*th* point cloud containing N points with 3D position and other auxiliary information (*e.g.*, color, intensity); $\mathcal{L}^k = \{l_n^k \in \mathcal{C}\}_{n=1}^N$ contains semantic labels for the points in $\mathcal{P}^k$, where $\mathcal{C}$ is the label list, *e.g.*, $\mathcal{C} = \{car, road, \cdots\}$. The segmentation network is achieved as $h \circ \varphi : \mathcal{P} \mapsto \mathcal{L}$, where $\varphi : \mathbb{R}^{N\times(3+x)} \mapsto \mathbb{R}^{N\times d}$ is a *feature extractor* (▥ in Figure 3.1) that embeds points in $\mathcal{P}$ into a d-dimensional feature space, and $h : \mathbb{R}^{N\times d} \mapsto \mathbb{R}^{N\times|\mathcal{C}|}$ is a *segmentation head* (▯) usually consisting of a small MLP, mapping point features into the discriminative semantic space for point-wise, $|\mathcal{C}|$-way classification. Thus the whole network is typically learned by minimizing the point-wise cross-entropy loss[1]:

$$\mathcal{J}_{\mathrm{CE}}(p_n) = -\log P(l_n|p_n) = -\log \frac{\exp(y_{n,l_n})}{\sum_{c\in\mathcal{C}} \exp(y_{n,c})}, \tag{3.1}$$

where $\boldsymbol{y}_n = [y_{n,c}]_c \in \mathbb{R}^{|\mathcal{C}|}$ is the vector of categorical scores (*logits*) for point p_n, *i.e.*, $\boldsymbol{y}_n = h(\boldsymbol{p}_n)$, and $\boldsymbol{p}_n \in \mathbb{R}^d$ is the feature of p_n obtained from φ. For the feature extractor φ, there already have many candidates (*e.g.*, voxel-/point-based 3D networks) elaborately designed to capture the specific geometries of point data. However, point clouds yield rich and diverse patterns,

[1]In practice, some other losses (*e.g.*, lovász loss [191]) can be used as complementary, but this does not affect our conclusion.

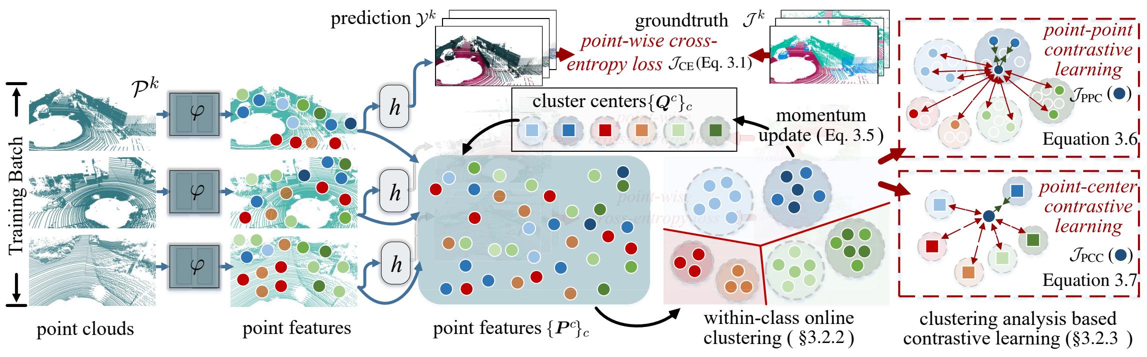

prediction $\mathcal{Y}^k$
groundtruth $\mathcal{J}^k$
point-wise cross-entropy loss $\mathcal{J}_{CE}$ (Eq. 3.1)
$\mathcal{P}^k$
Training Batch
φ
h
cluster centers $\{Q^c\}_c$
momentum update (Eq. 3.5)
point-point contrastive learning
$\mathcal{J}_{PPC}$ (●)
Equation 3.6
point-center contrastive learning
$\mathcal{J}_{PCC}$ (●)
Equation 3.7
point clouds
point features
point features $\{P^c\}_c$
within-class online clustering (§3.2.2)
clustering analysis based contrastive learning (§3.2.3)

FIGURE 3.1: **Overview** of our clustering based supervised learning algorithm for point cloud segmentation (Section 3.2).

e.g., fine-grained semantics, intra-class variations, *etc*. These patterns reflect underlying data structures; they are informative yet challenging for semantic understanding, and even hard to be identified. Thus it is usually the case that simply learning the segmentation network $h \circ \varphi$ from the supervision of easily-acquired high-level semantic tags (*i.e.*, Equation 3.1), without considering the underlying data structures.

We instead devise a *clustering analysis* based supervised learning framework (Figure 3.1). Our algorithm not only learns point recognition with pre-given semantic tags, but more essentially, it automatically discovers and encodes latent structures of point data into the feature space φ. Features learned in such strategy are expected to be more discriminative for (fine-grained) semantics and robust for intra-class variations, hence facilitating final dense recognition of point clouds.

At each training iteration, our algorithm has two phases. In **phase 1**, we perform online clustering over a massive quantity of points inside of each labeled classes. The purpose is to search for subclass patterns which are hard to be labeled yet significant across scenes. In **phase 2**, in addition to optimizing the whole segmentation network $h \circ \varphi$ with the point-wise classification loss $\mathcal{L}_{\text{CE}}$ as usual, we leverage deterministic cluster assignments as an auxiliary constraint to shape the feature space φ. The improved features, in turn, enable more reliable within-class clustering, and eventually boost point recognition. Independent of a certain point segmentation network, our training scheme is powerful and general.

3.2.2 Online Clustering based Subclass Pattern Mining

Our algorithm is built upon an intuitive insight: capturing underlying data structures can facilitate point representation learning and semantic recognition. Thus the first major question arises: *how to automatically and efficiently discover underlying data structures, which cannot be explicitly labeled, from massive training points?* This motivates us to conduct unsupervised clustering inside each labeled class $c \in \mathcal{C}$ so as to automatically mine representative yet latent subclass patterns. To scale our algorithm to millions of point data, we formulate such within-class clustering as optimal transport, which can be efficiently solved using Sinkhorn Iteration [198]. In addition, to overcome the computational expensive process of cluster center computation, which requires a full epoch over the entire dataset after every update of the representation, we adopt a momentum update strategy for proceeding online clustering simultaneously with network batch training.

For each class $c \in \mathcal{C}$, we assume it contains M latent, fine-grained patterns. Hence there are a total of $M \times |\mathcal{C}|$ unobservable patterns are desired to be discovered from the training dataset $\{\mathcal{P}^k, \mathcal{L}^k\}_k$. To do so, we perform within-class clustering on the point embedding space φ. As a result, the training points belonging to class c, *i.e.*, $\mathcal{P}^c = \{p_n | l_n = c\}$, are partitioned into M subclasses, and the M patterns of class c can be intuitively represented as the corresponding cluster centers. Let $\boldsymbol{Q}^c = [\boldsymbol{q}_1^c, \cdots, \boldsymbol{q}_M^c] \in \mathbb{R}^{d \times M}$ denote the M cluster centers of class c (*e.g.*, ▣▣ in Figure 3.1), and $\boldsymbol{P}^c = [\boldsymbol{p}_1^c, \cdots, \boldsymbol{p}_{N^c}^c] \in \mathbb{R}^{d \times N^c}$ all the features[2] of points belonging to class c (*e.g.*, ▦), where $p^c \in \mathcal{P}^c$ and $N^c = |\mathcal{P}^c|$. The cluster assignment can be represented as a binary matrix, $\boldsymbol{A}^c \in \{0,1\}^{M \times N^c}$, where the (m,i)-*th* element of $\boldsymbol{A}^c$ indicates whether assigning the i-*th* point of $\mathcal{P}^c$ to the m-*th* cluster center, *i.e.*, the m-*th* subclass, of c. The clustering inside class c can be achieved as the optimization of the assignment matrix $\boldsymbol{A}^c$, *i.e.*, maximizing the similarity between the point features and cluster centers:

$$\begin{gathered}\min_{\boldsymbol{A}^c \in \mathcal{A}^c} \langle \boldsymbol{A}^{c\top}, -\log \boldsymbol{S}^c \rangle_F, \\ \mathcal{A}^c = \{\boldsymbol{A}^c \in \{0,1\}^{M \times N^c} | \boldsymbol{A}^{c\top}\mathbf{1}_M = \mathbf{1}_{N^c}, \boldsymbol{A}^c \mathbf{1}_{N^c} = \frac{N^c}{M}\mathbf{1}_M\}\end{gathered} \tag{3.2}$$

where $\boldsymbol{S}^c = \text{softmax}(\boldsymbol{Q}^{c\top}\boldsymbol{P}^c)$ refers to the similarity matrix between cluster centers and points, $\langle\cdot\rangle_F$ is the Frobenius dot-product, log is applied element-wise, and $\mathbf{1}_M$ denotes the vector of ones in dimension M. For the solution space $\mathcal{A}^c$, the former constraint enforces that each point is assigned to exactly one subclass, and the later imposes an equipartition constraint [120, 144] to inspire the N^c

[2]Point feature has been projected to the unit sphere: $\boldsymbol{p} = \boldsymbol{p}/||\boldsymbol{p}||_2$; $\boldsymbol{p}$ is reused without causing ambiguity.

points to be grouped into M subclasses of equal size. The equipartition constraint helps avoid the degenerate solution where all the point samples are partitioned to a single cluster [142]. By relaxing $\boldsymbol{A}^c$ to be an element of *transportation polytope* [198], *i.e.*, $\mathcal{A}'^c = \{\boldsymbol{A}^c \in \mathbb{R}_+^{M \times N^c} | \boldsymbol{A}^{c\top}\mathbf{1}_M = \frac{1}{N^c}\mathbf{1}_{N^c}, \boldsymbol{A}^c\mathbf{1}_{N^c} = \frac{1}{M}\mathbf{1}_M\}$, the label assignment task can be viewed as an instance of the *optimal transport* problem, which can be efficiently solved by a fast version of the Sinkhorn-Knopp algorithm [198]:

$$\min_{\boldsymbol{A}^c \in \mathcal{A}'^c} \langle \boldsymbol{A}^{c\top}, -\log \boldsymbol{S}^c \rangle + \frac{1}{\lambda}\mathrm{KL}(\boldsymbol{A}^c || \frac{1}{MN^c}\mathbf{1}_M \mathbf{1}_{N^c}^\top), \tag{3.3}$$

where KL is the Kullback-Leibler divergence, and λ is the strength of the regularization. The solution of Prob. (3.3) over the set $\mathcal{A}'^c$ can be written as:

$$\boldsymbol{A}^{c*} = \mathrm{diag}(\boldsymbol{u})(\boldsymbol{S}^c)^\lambda \mathrm{diag}(\boldsymbol{v}), \tag{3.4}$$

where exponentiation is meant element-wise. $\boldsymbol{u} \in \mathbb{R}^M$ and $v \in \mathbb{R}^{N^c}$ are two vectors of scaling coefficients, obtained using a small number of matrix-vector multiplications via iterative Sinkhorn-Knopp algorithm [198]. Due to the drift of the point representation caused by iterative network training, after each training batch of point clouds, re-computing the cluster assignment would cost a pass over the full data. To circumvent such computationally expensive procedure of offline cluster assignment, we restrict the transportation polytope to the minibatch, through approximating the cluster centers $\boldsymbol{Q}^c$ with momentum. Specifically, at each training iteration, each cluster center $\boldsymbol{q}_m^c$ of class c is updated as:

$$\boldsymbol{q}_m^c \leftarrow \mu \boldsymbol{q}_m^c + (1-\mu)\bar{\boldsymbol{p}}_m^c, \tag{3.5}$$

where $\mu \in [0, 1]$ is the momentum coefficient, and $\bar{\boldsymbol{p}}_m^c$ indicates mean feature vector of the points that are assigned to cluster center $\boldsymbol{q}_m^c$ in the current batch. The cluster centers are initialized randomly and gradually updated every batch, following smoothly the changing of the representation φ. These designs lead to scalable and online clustering, allowing to automatically mine latent subclass patterns from massive point data. The clustering is very efficient on GPU; in practice, assigning 50K points into 40 clusters takes only 60 ms. We visualize clustering results ($M = 2$) of five classes in Figure 3.2, where subclasses under the same class are associated with similar colors. As seen, points with similar patterns are grouped together, thus the underlying data distribution of each class can be comprehensively captured. In Figure 3.2a, we visualize some clustering results of ‘building’, ‘vegetation’, ‘car’, ‘sidewalk’, and ‘road’ classes. For better visualization, different clusters/patterns are shown in different colors and points of other classes are plotted in dark green. Each subfigure only shows the clusters belonging to one single class. For example, for the ‘road’

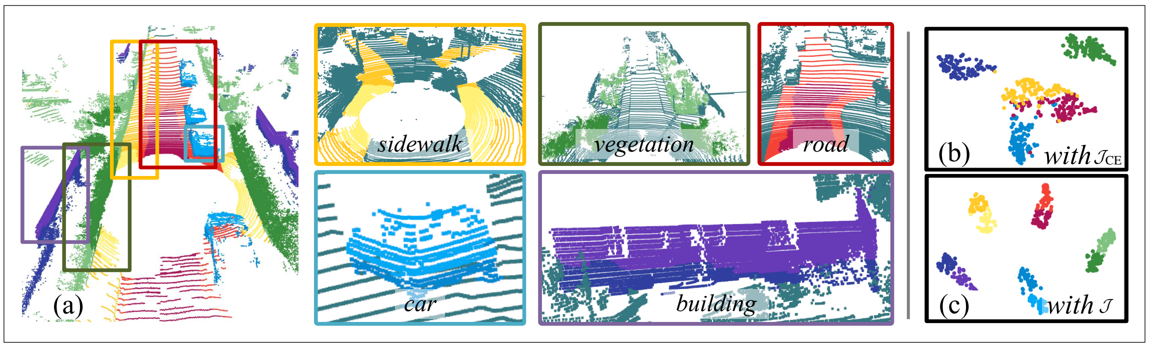


FIGURE 3.2: (a) Our **clustering results** for five classes, *i.e.*, *sidewalk*, *vegetation*, *road*, *car*, and *building*. (b-c) t-SNE **visualization** of point features $\{\boldsymbol{P}^c\}_c$ learned with $\mathcal{J}_{\text{CE}}$ (Equation 3.1) and $\mathcal{J}$ (Equation 3.8). We set $M=2$ here.

subfigure, only clusters for 'road' class are colored by red; points of other classes are plotted in dark green. In general, for car and building, the objects are roughly divided into upper and lower parts. Meanwhile, road, sidewalk, and vegetation are split on the horizontal planes.

3.2.3 Clustering Analysis based Point Cloud Representation Learning

Through within-class clustering, we search for latent structures in point clouds, and detect locally discriminative patterns, *i.e.*, the cluster centers $\{\boldsymbol{q}^c_m\}_{m,c}$. The next question is: *how to leverage these fine-grained patterns to aid point cloud representation learning?* To answer this, we complement the supervised point-wise classification loss $\mathcal{J}_{\text{CE}}$ (Equation 3.1) with a clustering analysis based contrastive learning strategy, which poses structured and direct supervision for point representation. In particular, with the deterministic cluster assignments in Section 3.2.2, we conduct contrastive representation learning over both point-point and point-center pairs. This allows us to fully exploit relations between any two points and local data structures, and directly optimize the point feature space φ.

Point-Point Contrastive Learning. Our point-point contrastive learning is achieved by comparing pairs of points to push away point representations from different subclasses while pulling together those from the same subclass. The corresponding training objective for each point p_n is defined as:

$$\mathcal{J}_{\text{PPC}}(p_n)=\frac{1}{|\mathcal{O}_{p_n}|}\sum_{p^+\in\mathcal{O}_{p_n}}-\log\frac{\exp(\boldsymbol{p}_n\cdot\boldsymbol{p}^+/\tau)}{\exp(\boldsymbol{p}_n\cdot\boldsymbol{p}^+/\tau)+\sum_{p^-\in\mathcal{N}_{p_n}}\exp(\boldsymbol{p}_n\cdot\boldsymbol{p}^-/\tau)}, \tag{3.6}$$

where $\tau>0$ is a scalar temperature parameter, $\mathcal{O}_{p_n}$ and $\mathcal{N}_{p_n}$ denote collections of *positive* and *negative* samples, respectively, for p_n. Training points belonging to the same cluster of p_n are positive samples, while being assigned to other clusters are negative. Note that the positive (negative) samples are not limited to a same training point cloud. To further boost our point-point contrastive learning, we follow the common practice in unsupervised representation learning [115, 199, 200] to build a memory bank per cluster, leading to $M \times |\mathcal{C}|$ memory banks totally. The memory banks gather point features of corresponding clusters from previous training batches, hence increasing the quantity and diversity of positive and negative samples. These designs deliver a *cross-scene* training scheme, rather than the current *de facto* scene-wise training paradigm that ignores the rich correspondences among points across different scenes. Minimizing Equation 3.6 leads to a well-structured embedding space φ, where points with similar patterns are grouped close to each other while points with dissimilar patterns are separated.

Point-Center Contrastive Learning. With a similar spirit of point-point contrastive learning, *i.e.*, inspiring intra-cluster compactness and inter-cluster separation, our point-center contrastive learning strategy contrasts the similarities between points and cluster centers on the embedding space φ:

$$\mathcal{J}_{\text{PCC}}(p_n) = -\log \frac{\exp(\boldsymbol{p}_n \cdot \boldsymbol{q}^{+}/\tau)}{\sum_{c,m} \exp(\boldsymbol{p}_n \cdot \boldsymbol{q}^{c}_{m}/\tau)}, \tag{3.7}$$

where $\boldsymbol{q}^{+}$ refers to the cluster center of point p_n. Equation 3.7 lets p_n find out the assigned cluster center $\boldsymbol{q}^{+}$ from all the centers $\{\boldsymbol{q}^{c}_{m}\}_{c,m}$, so as to decrease the distance between $\boldsymbol{p}_n$ and $\boldsymbol{q}^{+}$, while increasing the distance between $\boldsymbol{p}_n$ and other cluster centers. Since cluster centers are representative of the dataset, Equation 3.7 provides a cheaper and more direct way to impose dataset-level context, or underlying data structures, on feature space optimization, compared with the point-point contrastive learning (Equation 3.6). In practice, we find combining the two cluster-analysis based contrastive learning strategies yields the best performance (see detailed experiments in Section 3.3.5). One may also view point-center contrastive learning from an *information bottleneck* perspective [143, 201], wherein the deterministic clustering imposes a natural bottleneck and discretizes the embedding space φ as a finite set of cluster centers, *i.e.*, $\{\boldsymbol{q}^{c}_{m}\}_{c,m}$, through minimizing Equation 3.7, as opposed to learning φ as a continuous vector space.

Overall Training Objective. The standard cross-entropy loss $\mathcal{J}_{\text{CE}}$ in Equation 3.1 is essentially a unary training objective that is only aware of point-wise semantic discrimination, without accounting for any underlying data structure and pairwise relations between training points. The clustering analysis based contrastive losses, *i.e.*, $\mathcal{J}_{\text{PPC}}$ in Equation 3.6 and $\mathcal{J}_{\text{PCC}}$ in Equation 3.7, are pairwise training objectives that exploit locally representative patterns for structure-aware, distance based point representation learning. Thus we assemble these two complementary training targets as our overall learning objective:

$$\mathcal{J} = \mathcal{J}_{\text{CE}} + \alpha(\mathcal{J}_{\text{PPC}} + \mathcal{J}_{\text{PCC}}), \tag{3.8}$$

where $\mathcal{J}_{\text{CE}}$ denotes the cross-entropy objective defined in Equation 3.1, $\alpha \in [0, 1]$ is a balancing coefficient, and $\mathcal{J}_{\text{PPC}}$ and $\mathcal{J}_{\text{PCC}}$ are the proposed clustering-based contrastive learning objectives. Our training algorithm alternately performs within-class clustering over the point embedding space φ, and optimizes the whole segmentation network $h \circ \varphi$ with the semantic labels $\{\mathcal{L}^k\}_k$ and cluster assignments $\{\boldsymbol{A}^c\}_c$. As such, meaningful clusters capture fine-grained data structures and become informative supervisory signals for point representation learning; in turn, discriminative representations help obtain meaningful clusters and eventually ease point recognition. In Figure 3.2 (b-c), we provide visualization of point embeddings learned by $\mathcal{J}_{\text{CE}}$ and $\mathcal{J}$ (sub-class clusters under the same class are associated with similar colors). As can be seen, incorporating training targets based on clustering analysis yields a more structured point embedding space. The resulting clusters capture meaningful patterns, with representations from different classes well separated and subpatterns within the same class both closely grouped and clearly disentangled.

3.2.4 Algorithm Details

Online Clustering (Section 3.2.2). We group point samples of each class into M subclasses for exploiting latent structures of the entire dataset. We empirically set $M = 40$ and the momentum coefficient in Equation 3.5 $\mu = 0.9999$ (related experiments can be found in Section 3.3.5). Following [144], we set $\lambda = 25$ in Equation 3.3.

Clustering Analysis based Training (Section 3.2.3). Our clustering analysis based training strategy enforces the point feature space to better respect the discovered data structures. Following the common practice in contrastive learning [116, 145], we set the scalar temperature τ in Eqs. (3.6-3.7) as 0.1. For the cluster-wise memory bank, we sample 10 point features per-cluster from each scene and store all the sampled features of all the training point clouds $\{\mathcal{P}^k\}_k$. For the training loss $\mathcal{J}$ (Equation 3.8), the coefficient is set as $\alpha = 1$ (we empirically find our algorithm is insensitive to α when $\alpha \in [0, 1]$).

Point Cloud Segmentation Network $h \circ \varphi$. Our algorithm is a general supervised learning scheme for point cloud segmentation. In principle, it can be applied to any segmentation networks that can learn point-wise features. In our experiments, we approach our algorithm on four typical segmentation networks, including voxel-based [6], point-based [64], Transformer-based [7], and NAS-based [87].

Inference. Our training algorithm does not cause extra inference cost or network architectural modification during model deployment. The $M \times |\mathcal{C}|$ cluster centers and $M \times |\mathcal{C}|$ memory banks are directly discarded after network training.

3.3 Experiment

We first report our 3D segmentation results on static point clouds of urban scenes and indoor environments in Section 3.3.1 and Section 3.3.2, respectively. Then we assess our performance on 4D segmentation of outdoor point cloud sequences in Section 3.3.3. For thorough evaluation, in Section 3.3.4, we extend our algorithm to 3D object detection setting and conduct experiments. The hyperparameters mentioned in Section 3.2.4 are used for all the above experiments. Finally, in Section 3.3.5, we provide ablative analyses on the core components of our training algorithm.

Base Segmentation Networks. For thorough examination, we apply our training algorithm to Cylinder3D [6] (*voxel*-based), KPConv [64] (*point*-based), PTV1 [7] (*Transformer*-based), and SPVNAS [87] (*NAS*-based), which are representative for current mainstream network architectures in point cloud segmentation and with publicly accessible implementations. For fair comparison, we adopt their default implementation settings, including hyper-parameters and augmentation recipes.

3.3.1 3D Segmentation on Static Urban Point Clouds

Dataset and Metric. SemanticKITTI [1] is a large-scale driving-scene dataset for point cloud segmentation. It has 43,000 scans with point-wise annotation, collected from 22 sequences. According to the official setting, we use sequences 00 to 10 for `train` (but 08 is left for `val`), and 11 to 21 for `test`. In single-scan challenge for static segmentation, 19 classes are used; Mean Intersection over Union (mIoU) and Intersection over Union (IoU) of each class are reported.

Quantitative Result. Table 3.1 reports comparison results on SemanticKITTI single-scan challenge `test`. Moreover, complete quantitative results can be found in Appendix A. As seen, our algorithm

TABLE 3.1: **Quantitative 3D segmentation results** on SemanticKITTI [1] single-scan challenge `test` (Section 3.3.1). For clarity, IoUs on 6 of 19 classes are given (c_1: *sidewalk*, c_2: *parking*, c_3: *building*, c_4: *truck*, c_5: *bicycle*, c_6: *motorcyclist*).

Method	mIoU(%)	c_1(%)	c_2(%)	c_3(%)	c_4(%)	c_5(%)	c_6(%)
PointASNL [CVPR20] [46]	46.8	74.3	24.3	83.1	39.0	0.0	0.0
PolarNet [CVPR20] [71]	54.3	74.4	61.7	90.0	22.9	40.3	5.6
RandLA-Net [CVPR20] [47]	55.9	74.0	61.8	89.7	43.9	29.8	9.4
SqueezeSegV3 [ECCV20] [79]	55.9	74.8	63.4	89.0	29.6	38.7	20.1
SalsaNext [ISVC20] [78]	59.5	75.8	63.7	90.2	38.9	48.3	19.4
FusionNet [ECCV20] [86]	61.3	77.1	68.8	92.5	41.8	47.5	11.9
JS3C-Net [AAAI21] [202]	66.0	72.1	61.9	92.5	54.3	59.3	39.9
AF2S3Net [CVPR21] [98]	69.7	72.5	68.8	87.9	39.2	65.4	74.3
RPVNet [ICCV21] [91]	70.3	80.7	70.3	93.5	44.2	68.4	43.4
PVKD [CVPR22] [203]	71.4	77.5	70.9	92.4	53.5	67.9	50.5
KPConv [ICCV19] [64]	58.8	72.7	61.3	90.5	33.4	30.2	11.8
KPConv + **`Ours`**	**61.0** ↑2.2	75.0	63.4	91.4	49.0	45.0	36.4
SPVNAS$_{10.8M}$ [ECCV20] [87]	62.3	73.8	63.2	90.9	50.9	40.6	21.8
SPVNAS$_{10.8M}$ + **`Ours`**	**64.3** ↑2.0	73.9	64.0	91.4	48.0	48.9	23.2
Cylinder3D [CVPR21] [6]	67.8	75.5	65.1	91.0	50.8	67.6	36.0
Cylinder3D + **`Ours`**	**70.4** ↑2.6	77.2	66.1	92.3	51.9	68.4	54.6

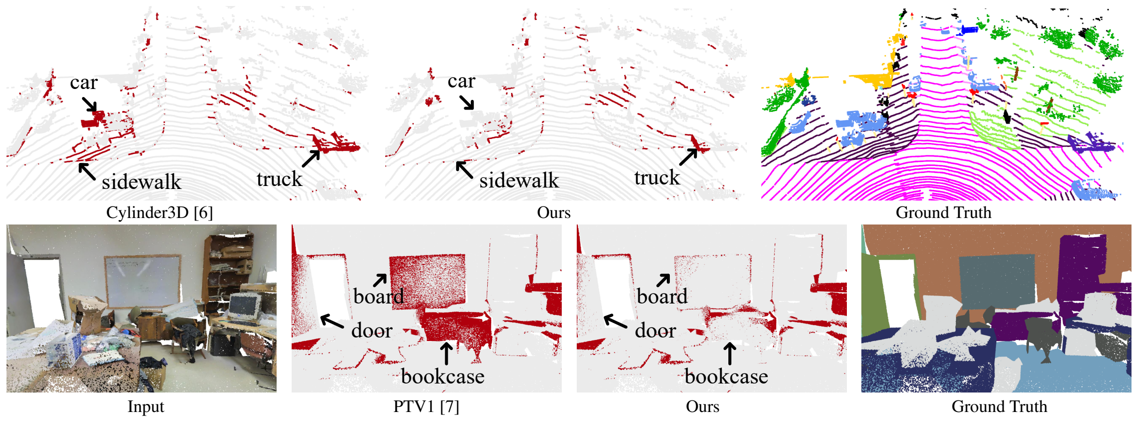


FIGURE 3.3: **Error maps** on SemanticKITTI [1] single-scan challenge `val` (**top**), and S3DIS [2] Area-5 (**bottom**). The differences are illustrated by arrows.

improves the performance of the base segmentation networks by solid margins. Concretely, it yields **2.2**%, **2.6**%, and **2.0**% mIoU gains over point-based KPConv [64], voxel-based Cylinder3D [6], and SPVNAS [87], respectively. Our algorithm also obtains consistent performance improvements across most classes. These results illusrate the wide potential benefit of our algorithm. Moreover, "Cylinder3D + `Ours`" reaches comparable results with published competitors. This is particularly impressive considering the fact that the improvement is solely achieved by our training scheme, without any network architectural modification and inference speed delay.

Qualitative Result. As shown in the top row of Figure 3.3, our method can reduce errors over both

TABLE 3.2: **Quantitative 3D segmentation results** on S3DIS [2] Area-5 (Section 3.3.2). For clarity, IoUs on 5 of 13 classes are given (c_1: *wall*, c_2: *column*, c_3: *window*, c_4: *door*, c_5: *board*).

Method	mIoU(%)	mAcc(%)	c_1(%)	c_2(%)	c_3(%)	c_4(%)	c_5(%)
HPEIN [ICCV19] [45]	61.9	68.3	81.4	23.3	65.3	40.0	65.6
PAT [CVPR19] [67]	60.1	70.8	72.3	41.5	85.1	38.2	61.3
PointWeb [CVPR19] [44]	60.3	66.6	79.4	21.1	59.7	34.8	64.9
MinkowskiNet [CVPR19] [18]	65.4	71.7	86.2	34.1	48.9	62.4	74.4
SCF-Net [CVPR21] [48]	63.8	-	-	-	-	-	-
BAAF-Net [CVPR21] [204]	65.4	73.1	-	-	-	-	-
CGA-Net [CVPR21] [205]	68.6	-	83.0	25.3	59.6	71.0	69.5
PTV1+CBL [CVPR22] [206]	71.6	77.9	-	-	-	-	-
Stratified Trans. [CVPR22] [207]	72.0	78.1	-	-	-	-	-
PTV2 [NeurIPS22] [208]	72.6	78.0	-	-	-	-	-
KPConv [ICCV19] [64]	67.1	72.8	82.4	23.9	58.0	69.0	66.7
KPConv+ `Ours`	**69.0** ↑1.9	**76.2** ↑3.4	84.0	30.7	66.7	77.6	63.0
PTV1 [ICCV21] [7]	70.4	76.5	86.3	38.0	63.4	74.3	76.0
PTV1+ `Ours`	**72.2** ↑1.8	**79.6** ↑3.1	88.1	49.3	65.3	79.4	81.0

TABLE 3.3: **Quantitative 4D segmentation results** on SemanticKITTI [1] multi-scan challenge `test` (Section 3.3.3). IoUs on 6 of 25 classes are reported (c_1: *sidewalk*, c_2: *moving car*, c_3: *moving truck*, c_4: *bicycle*, c_5: *motorcyclist*, c_6: *traffic-sign*).

Method	mIoU(%)	c_1(%)	c_2(%)	c_3(%)	c_4(%)	c_5(%)	c_6(%)
TangentConv [CVPR18] [73]	34.1	64.0	40.3	1.1	2.0	0.0	31.2
DarkNet53 [ICCV19] [1]	41.6	75.3	61.5	14.1	30.4	0.0	31.2
TemporalLidarSeg [3DV20] [22]	47.0	75.8	68.2	2.1	47.7	0.0	60.4
SpSeqnet [CVPR20] [23]	43.1	73.9	53.2	41.2	24.0	0.0	48.7
KPConv [ICCV19] [64]	51.2	70.5	69.4	5.8	44.9	0.0	53.9
KPConv+ `Ours`	**53.2** ↑2.0	75.2	75.2	4.1	67.2	9.9	64.6
Cylinder3D [CVPR21] [6]	52.5	74.5	74.9	0.0	67.6	0.2	61.4
Cylinder3D+ `Ours`	**54.7** ↑2.2	76.9	81.7	11.9	55.9	3.0	68.0

small nature objects (such as *trunk*) and widely distributed classes (like *sidewalk*). More qualitative results can be found in Appendix A.

3.3.2 3D Segmentation on Static Indoor Point Clouds

Dataset and Metric. S3DIS [2] is a famous 3D indoor parsing dataset. It contains 273M points collected from six areas and labeled with 13 classes. Following [64, 192], we use Area-5 as test scene to better test the generalization ability. We report two metrics: mIoU and Mean Accuracy (mAcc).

Quantitative Result. Table 3.2 summarizes the comparison results on S3DIS, showing our training algorithm also works well on large-scale challenging indoor point clouds. In particular, our algorithm brings impressive gains over KPConv, *i.e.*, 67.1%→**69.0**% and 72.8%→**76.2**%, in terms of mIoU and mAcc. Notably, with PTV1 as the backbone, our approach attains mIoU/mAcc of **72.2%**/**79.6%**, outperforming PTV1+CBL (71.6%/77.9%). Moreover, complete quantitative results can be found in Appendix A.

Qualitative Result. As shown in the bottom row of Figure 3.3, our method significantly reduces the errors of PTV1 [7] in an indoor environment of S3DIS [2] Area-5. More qualitative results can be found in Appendix A.

3.3.3 4D Segmentation on Urban Point Sequences

Dataset and Metric. SemanticKITTI [1] multi-scan challenge is devoted to 4D point cloud segmentation. It involves six more classes to distinguish between moving objects and stationary ones for *car*, *trunk*, *bicyclist*, *other-vehicle*, *person*, and *motor-cyclist* categories. mIoU is adopted as the evaluation metric.

Quantitative Result. Table 3.3 reports our comparison results on SemanticKITTI [1] multi-scan challenge `test`. More complete quantitative results can be found in Appendix A. Our algorithm, again, leads to improvements over backbones, *i.e.*, **2.0**% and **2.2**% mIoU gain compared with KPConv [64] and Cylinder3D [6], respectively. This confirms our algorithm is also applicable in point cloud sequences. Our algorithm also obtains superior performance for vehicle categories with moving patterns, such as *moving car*, *moving truck*, *moving other-vehicle*, *etc*. We attribute this to our capacity of capturing complex patterns and variations, which improves the robustness in dynamic scenes.

Qualitative Result. Qualitative results between Cylinder3D [6] and our method can be found in Appendix A.

3.3.4 3D Detection on Static Urban Point Clouds

To fully reveal the power of our idea, we conduct additional experiments on 3D object detection, although we put our main focus on the point cloud segmentation task.

Algorithmic Modification. To apply our algorithm to the 3D object detection task and minimize the modification effort, we view the bounding box annotations as a form of coarse segmentation

TABLE 3.4: **Quantitative 3D detection results** on KITTI [8] challenge `val` (Section 3.3.4).

Difficulty	Method	mAP(%)	Car(%)	Pedestrian(%)	Cyclist(%)
Easy	Second [SENSORS18] [89]	75.25	88.61	56.55	80.59
	Second + **Ours**	**78.60** ↑3.35	89.13	58.50	88.16
	PointPillar [CVPR19] [209]	74.76	86.46	57.75	80.06
	PointPillar + **Ours**	**76.82** ↑2.06	88.34	58.19	83.92
Moderate	Second [SENSORS18] [89]	66.25	78.62	52.98	67.16
	Second + **Ours**	**69.67** ↑3.42	82.97	55.64	70.39
	PointPillar [CVPR19] [209]	64.08	77.28	52.29	62.68
	PointPillar + **Ours**	**66.07** ↑1.99	78.43	53.31	66.47
Hard	Second [SENSORS18] [89]	62.69	77.22	47.73	63.11
	Second + **Ours**	**65.36** ↑2.67	78.55	50.91	66.61
	PointPillar [CVPR19] [209]	60.76	74.65	47.91	59.71
	PointPillar + **Ours**	**62.96** ↑2.20	77.14	49.15	62.61

labels. For each labeled bounding box with semantic class $c \in \mathcal{C}$, we simply treat all the points within the bounding box as data examples of class c, which are used in our clustering analysis based representation learning (*cf.* Equation 3.6 and Equation 3.7). Note that there is no change to the base 3D detection network, including the detection head.

Dataset and Metric. KITTI [8] is a standard benchmark for 3D object detection. We split 3712 scans for `train` and 3769 scans for `val`, with 3D bounding box annotations of vehicles, pedestrians and cyclists. Detection outcomes are evaluated under three regimes: *easy, moderate, hard*, defined according to occlusion and truncation levels of objects. Average precisions are reported with IoU thresholds of 0.7, 0.5, and 0.5, respectively for *car*, *pedestrian*, and *cyclist* classes. We use Mean Average Precision (mAP) as our evaluation measure.

Base Detection Networks. We apply our algorithm to two famous 3D detectors, *i.e.*, Second [89] and PointPillar [102].

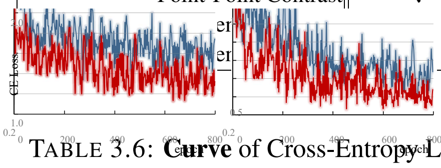
Quantitative Result. Table 3.4 reports the experimental results on KITTI `val`. We can observe that, for both Second and PointPillar, our training algorithm brings notable performance gains, across different classes and under different regimes. This proves the high versatility of our algorithm.

3.3.5 Diagnostic Experiment

To fully test the effectiveness of our core algorithm designs, we conduct a series of ablative studies on S3DIS [2] Area-5 and SemanticKITTI [1] multi-scan challenge `val`. We adopt KPConv [64]

TABLE 3.5: **Study of clustering based training** on S3DIS [2] Area-5 and SemanticKITTI [1] multi-scan `val` set (Section 3.3.5).

	$\mathcal{J}_{\text{PPC}}$ (Eq. (3.6))	$\mathcal{J}_{\text{PCC}}$ (Eq. (3.7))	S3DIS mIoU(%)	S-KITTI mIoU(%)	Training Speed (sec/epoch)
Baseline (*w/o* clustering analysis)			67.1	53.3	281.46
Point-Point Contrast	✓		68.0	54.4	310.20
[illegible]er		✓	68.4	54.7	310.28
[illegible]er		✓	**69.0**	**55.7**	311.71



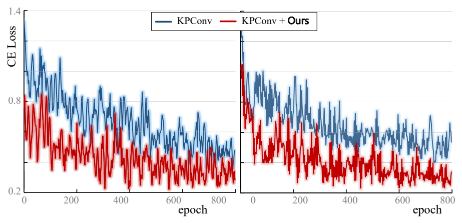


TABLE 3.6: **Curve** of Cross-Entropy Loss on SemanticKITTI [1] single-scan challenge `train` (**left**) and `val` (**right**).

TABLE 3.7: **Parameter studies** on S3DIS [2] Area-5 and SemanticKITTI [1] (S-KITTI) multi-scan `val` set (Section 3.3.5). (mIoU(%) is reported.)

# Cluster	S3DIS	S-KITTI
$M=1$	67.5	53.7
$M=10$	68.0	54.5
$M=20$	68.5	55.2
$M=$ **40**	**69.0**	**55.7**
$M=60$	68.9	55.2
$M=80$	68.7	55.5

(a) Per-class cluster Num

Memory Capacity	S3DIS	S-KITTI
Mini-Batch (*w/o* memory)	68.0	54.4
5 × #scene	68.6	55.0
10 × #scene	**69.0**	**55.7**
15 × #scene	68.8	55.7
20 × #scene	68.7	55.6

(b) Per-cluster memory

Coefficient μ	S3DIS	S-KITTI
$\mu=0$	67.7	53.6
$\mu=0.9$	68.0	54.0
$\mu=0.99$	68.5	54.7
$\mu=0.999$	68.6	55.3
$\mu=$ **0.9999**	**69.0**	**55.7**
$\mu=0.99999$	68.8	55.5

(c) Momentum coefficient

as our base segmentation network. The results are reported without post-processing or test-time augmentation.

Clustering Analysis based Network Training. We first test the efficacy of our core idea of clustering analysis based point representation learning. As shown in Table 3.5, the baseline model, trained in the standard strategy, gains 67.1% and 53.3% mIoU, on S3DIS and SemanticKITTI, respectively. Additionally considering point-point contrast $\mathcal{J}_{\text{PPC}}$ (Equation 3.6) or point-center contrast $\mathcal{J}_{\text{PCC}}$ (Equation 3.7) can lead to better performance. However, combining these two training objectives yields the best results, *i.e.*, 69.0% and 55.7%. These results verify that mining latent data structures can benefit detailed analysis of point cloud. Table 3.5 also gives comparisons for training speed. Our algorithm only brings negligible delay (∼30 s for each epoch), confirming its high efficiency.

Per-Class Cluster Number M. We next investigate the impact of the cluster number M of each class. The results are summarized in Table 3.7a. $M=1$ means that directly treating each class as a single cluster. This baseline obtains 67.5% and 53.7% mIoU, on S3DIS and SemanticKITTI, respectively. After clustering based fine-grained pattern mining, we observe consistent improvements, *e.g.*, 67.5%→69.0% on S3DIS when $M=40$. This verifies that **i)** there indeed exist some latent patterns in point clouds, and **ii)** these latent patterns are valuable for point cloud parsing. When $M>40$, further increasing M gives marginal performance gains even worse results. We speculate this is because the model is distracted by some trivial patterns due to over-clustering.

Memory Bank. Then we study the influence of our memory bank in Table 3.7b. "Mini-Batch (*w/o* memory)" means that only computing contrast within each mini-batch, without the memory; it earns 68.0% and 54.4% mIoU, on S3DIS and SemanticKITTI, respectively. We then provision this baseline with class-wise memory bank with different capacities. When storing 10 point features per scene for each cluster, the best performance is achieved, *i.e.*, 69.0% and 55.7%.

Momentum Coefficient μ. Table 3.7c gives the performance with regard to the momentum coefficient μ (*cf.* Equation 3.5), which controls the evolution speed of cluster centers. The model performs better with a relatively large coefficient (*i.e.*, $\mu = 0.9999$), showing that slow update is more favored. Moreover, at the extreme case of $\mu = 0$, the performance drops considerably, evidencing that simply approximating the cluster centers with per-batch cluster means is not a sound solution.

3.4 Summary

We devise a clustering based supervised training scheme for point cloud segmentation, which discovers and respects latent data structures during point representation learning. Rather than simply minimizing the point recognition error, we iteratively perform 1) unsupervised, within-class clustering based subclass pattern mining, and 2) clustering assignment based point embedding space optimization. Our algorithm is general and shows outstanding performance over various tasks and datasets. It also brings some new challenges, including the extension in instance-aware segmentation setting, and automatic estimation of the cluster number.

Chapter 4

LSK3DNet: Towards Effective and Efficient 3D Perception with Large Sparse Kernels

4.1 Introduction

Autonomous systems are equipped with 3D LiDAR sensors and data processing platforms. The LiDAR sensor generates point clouds, which serve as the input for the processing platform. The platform performs perception tasks such as semantic segmentation [16, 92, 210] and object detection [108, 211, 212], providing a foundation for motion planning and decision-making [33] in autonomous systems. On the one hand, the processing platform is required to handle large-scale, sparse, and irregular point clouds. On the other hand, the platform's computing resources are limited. Therefore, it is crucial to develop effective and efficient LiDAR perception methods. Point-based methods [7, 207, 210] rely on computationally expensive, memory inefficient, and time-consuming point sampling strategy [213]. In contrast, sparse convolution [18, 82, 110, 127] has been widely adopted for processing large-scale point clouds up to $160\times160\times20$ meters [6, 18, 82, 87, 108, 214]. Nevertheless, the application of such a technique in autonomous systems has also been hindered by limited computational resources [87]. PV-KD [203] attempts to reduce the model size with knowledge distillation, but its performance is bounded by its teacher model [6]. In opposition, large kernels

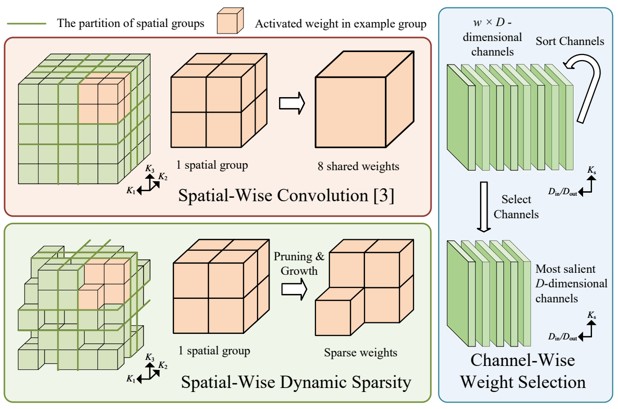

The partition of spatial groups
Activated weight in example group
1 spatial group
8 shared weights
Spatial-Wise Convolution [3]
Pruning & Growth
1 spatial group
Sparse weights
Spatial-Wise Dynamic Sparsity
w × D-dimensional channels
Sort Channels
Select Channels
Most salient D-dimensional channels
Channel-Wise Weight Selection

FIGURE 4.1: **Illustrations** on *Spatial-wise Group Convolution* [3], SDS, and CWS. The spatial dimensions K_s (*i.e.*, K_1, K_2, K_3) and channel dimensions (D_{in}/D_{out}) are shown. *Spatial-wise Group Convolution* shares the weights within each spatial group during training, leading to redundant model weights. In contrast, SDS removes non-salient weights and redundancies that are not sensitive to the input in each group, while CWS eliminates redundant weights in a channel-wise manner (Section 4.1).

can enhance performance through the advantages of large receptive fields [3, 159, 161]. However, naive 3D large kernels will significantly increase computational burdens. LargeKernel3D [3] explores large 3D kernels with *Spatial-wise Group Convolution* to circumvent optimization and efficiency problems caused by naive 3D large kernels. However, LargeKernel3D has a lot of redundant model weights during training as it shares the weights in each spatial group. Moreover, the performance of LargeKernel3D [3] drops when scaling up the kernel size over $7\times7\times7$.

In this study, we propose a Large Sparse Kernel 3D Neural Network (LSK3DNet), which is an efficient and effective 3D perception model. LSK3DNet incorporates two key components, namely *Spatial-wise Dynamic Sparsity* (SDS) and *Channel-wise Weight Selection* (CWS). These components enhance performance on perception tasks and overcome the challenges of high computational costs. SDS enlarges 3D kernels and reduces model parameters by learning sparse 3D kernels from scratch. CWS increases the model width and learns channel-wise importance during training, resulting in accelerated inference by pruning redundant channels [215]. As shown in Figure 4.1, SDS can prune 3D kernels to remove redundant weights, while CWS can eliminate redundant channels in an online manner. Our SDS and CWS complement each other following a guideline of *"using spatial sparse groups, expanding width without more parameters"*, in contrast to *"using sparse groups, expanding more"* [161].

Specifically, SDS implements a remove-and-regrow process within each spatial group and preserves the sparsity in each group during dynamic training. This stands in stark contrast to the 2D

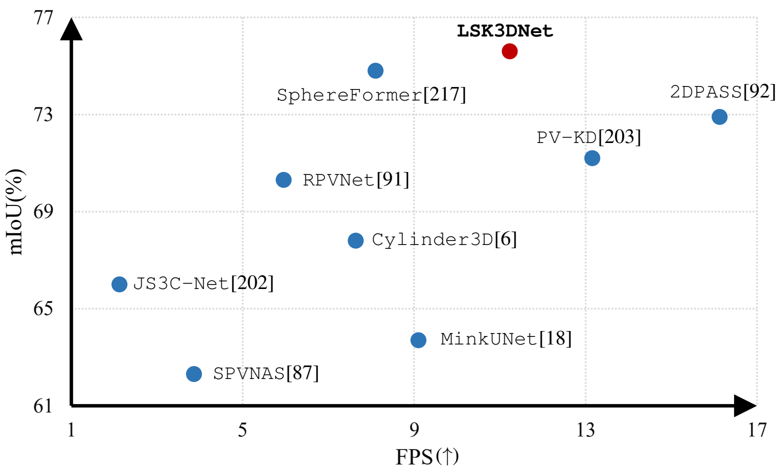


FIGURE 4.2: **Comparisons of classical methods.** Performance (mIoU) *vs.* Inference Speed (Frames Per Second (FPS)) on SemanticKITTI [1] single-scan challenge (Section 4.1).

approach [161], as it achieves dynamic sparsity in depth-wise groups. SDS allows to scale up the receptive fields with large kernel sizes, easily reaching or surpassing $9 \times 9 \times 9$, thereby achieving higher performance. In addition, CWS selectively identifies salient channels during training. It speeds up inference on the 3D vision tasks by pruning non-salient channel parameters. In this way, LSK3DNet can benefit from expanded width to achieve enhanced performance, but maintain the original model size during inference. Reducing the models' complexity (*e.g.*, model size) is a big benefit and a straightforward way to make them deployable on resource-constrained devices [216]. We conduct experiments on three benchmark datasets and five tracks. Our method achieves better performance compared to state-of-the-art methods [6, 91, 217] on SemanticKITTI [1] but with faster inference speed (Figure 4.2). Moreover, LSK3DNet outperforms the prior 3D large kernel method of LargeKernel3D [3] on ScanNet v2 [9] and KITTI [8].

In a nutshell, our **contributions** are as follows:

- We propose LSK3DNet to enhance performance in perception tasks and mitigate high computational costs, surpassing state-of-the-art models while reducing model size by 40% and computing operations by 60%.
- We propose SDS to scale up 3D kernels. It learns large sparse kernels by dynamically pruning and regrowing weights from scratch, thereby reducing model size and computational operations.

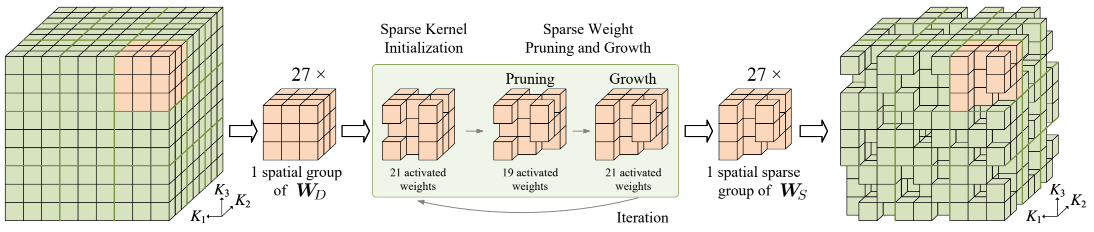


FIGURE 4.3: **Spatial-wise Dynamic Sparsity.** The utilization of SDS enables us to create and train sparse kernel 3D neural networks from the beginning. The sparse weights in each spatial group are firstly initialized by *Sparse Kernel Initialization* (Equation 4.2), and then regularly altered by discarding the least significant connections and introducing new ones (Equation 4.3). The sparse kernels are steadily improved by this dynamic process, which leads to a more thorough collection of local features. Note that different spatial sparse group has different sparse distribution. Here we take a sparsity of 22% for example (Section 4.2.3 and Algorithm 4.1).

- We develop CWS to improve performance by expanding the width. CWS assesses the importance of channels during training and speeds up inference by pruning redundant channel-wise parameters.

4.2 Methodology

In this section, we first formulate the problem in Section 4.2.1, then move on to a concise overview of submanifold sparse convolution in Section 4.2.2. Afterwards, the details of *Spatial-wise Dynamic Sparsity* (SDS) and *Channel-wise Weight Selection* (CWS) are described in Section 4.2.3 and 4.2.4. Lastly, network architecture is outlined in Section 4.2.5.

4.2.1 Problem Formulation

Suppose that the input is a set of unordered point $\mathcal{X} = \left\{x_i \in \mathbb{R}^{D_{in}} : i = 1, 2, \dots, N\right\}$ for 3D perception tasks, and D_{in} is the dimensionality of the input features. $\mathcal{L}$ is a set of object/point labels, which varies according to different datasets. For segmentation task, LSK3DNet assigns each point x_i with a predicted label y_i, resulting in a point-wise prediction set $\mathcal{Y} = \{y_i : i = 1, 2, \dots, N\}$. In voxel-based methods, point clouds are converted to voxel representations. We denote the input sparse tensor with $X_s^{in} = [C, F]$. Coordinate matrix $C \in \mathbb{N}_+^{N\times 3}$ includes 3-dimensional discrete coordinates, and feature matrix $F \in \mathbb{R}^{N\times D_{in}}$ has D_{in}-dimensional features.

4.2.2 Review of Submanifold Sparse Convolution

Regular sparse convolution has widespread use in U-Net type 3D networks [18, 82]. However, it significantly reduces the sparsity level of point data [88], obfuscates feature distinctions [88], and leads to low-resolution and indistinguishable small objects [87]. In our LSK3DNet, we discard regular sparse convolution and exclusively employ submanifold sparse convolution for feature extraction.

In a 3-dimensional space $C \in \mathbb{N}_{+}^{N\times 3}$, the D_{in}-dimensional input feature at $c \in C$ can be denoted as $x_c^{in} \in \mathbb{R}^{D_{in}}$, and the 3D kernel is represented by $\boldsymbol{W} \in \mathbb{R}^{K^3 \times D_{\text{out}} \times D_{\text{in}}}$. We divide the kernel into K^3 spatial weights, each with a size of $D_{out} \times D_{in}$, and denote the divided weights as $\boldsymbol{W}_i$. The submanifold sparse convolution is formulated as follows:

$$x_c^{out} = \sum_{i \in \mathcal{V}^D(c,C)} \boldsymbol{W}_i \, x_{c+i}^{in}, \quad c \in C, \tag{4.1}$$

where x_c^{out} represents the current voxel on which the submanifold sparse convolution is applied; $\boldsymbol{W}_i$ corresponds to x_{c+i}^{in}, which represents the i-th adjacent input voxel ($[C_{c+i}, F_{c+i}]$); $\mathcal{V}^D$ is a set of offsets that define the shape of a kernel, and $\mathcal{V}^D(c, C) = \{i | c + i \in C, i \in \mathcal{V}^D\}$ is a set of offsets from the current center c that exist in C [18]. It should be noted that the input coordinates and output coordinates are equivalent; in other words, they share the same C. Due to this restriction, insufficient information flow exists while differentiating the distinct spatial characteristics [88]. Large receptive fields could be a potential solution to this problem.

4.2.3 Spatial-wise Dynamic Sparsity (SDS)

SDS incorporates two essential components: *Sparse Kernel Initialization* and *Sparse Weight Pruning and Growth*. SDS is applied to the 3D large kernel layers, while other layers are kept dense. When initialized with *Sparse Kernel Initialization*, all spatial groups of sparse tensors have the same level of sparsity s, where sparsity refers to the fraction of zeros in sparse kernels. We adjust the sparse position within spatial groups of 3D large kernel layers with an adaptation frequency f_a. During training, at regular intervals determined by f_a, the adjustable weights in the kernels are modified through a two-phase procedure, which includes *sparse weight pruning* and *sparse weight growth*. We maintain a constant number of non-zero parameters (*i.e.*, sparsity s) throughout the training process.

Sparse Kernel Initialization. Prior research [218] has explored the distribution of sparsity parameters across multiple layers. The proportion and adjustment of sparse weights are regulated by the layer-wise sparsity ratio, which determines how the weights are modified to seek more efficient sparse structures during training. However, this method is not suitable for *Spatial-wise Group Convolution*, since 3D large kernels are divided into multiple spatial groups. To ensure that each group has non-zero weights, we execute Erdős-Rényi (ER) based sparsity ratios [163, 166, 219] in each spatial group. Dense weights $\boldsymbol{W}_D$ are firstly initialized in a standard way (*i.e.*, kaiming uniform initialization [220]). The binary Mask $\boldsymbol{M}$ is then applied to these dense weights; $\boldsymbol{M}$ is generated using an ER-based method with a sparsity of s. The zero-weight number in $\boldsymbol{M}$ is scaled by $1-\frac{D_{in}+D_{out}+K_1^g+K_2^g+K_3^g}{D_{in}\times D_{out}\times K_1^g\times K_2^g\times K_3^g}$, where K_1^g, K_2^g, and K_3^g denote the spatial group sizes. Therefore, the sparse weights $\boldsymbol{W}_S$ are initialized as follows:

$$\boldsymbol{W}_S \leftarrow \boldsymbol{W}_D \odot \boldsymbol{M}, \tag{4.2}$$

where $\odot$ represents the Hadamard product.

To design 3D networks with sparse groups, following the recipe of *"using spatial sparse groups, expanding width without more parameters"*, we have chosen *Spatial-wise Group Convolution* rather than *Depth-wise Group Convolution*. Our experiments have also shown that *Depth-wise Group Convolution* is unsuitable for large 3D kernels. This aligns with the findings in LargeKernel3D [3].

Sparse Weight Pruning and Growth. Previous research [161] in 2D convolution has illustrated that dynamic sparsity can effectively enlarge kernel sizes and enhance scalability. But *Depth-wise Group Convolution* is unsuitable for large 3D kernels [3]. Therefore, we adapt sparse weights in each spatial group dynamically to accommodate the data. Specifically, a certain rate of connections (*i.e.*, prune rate p) with the lowest magnitude are eliminated, and then we generate an equal number of new random connections. The regrow weights are randomly placed at zero positions within each spatial group. This is achieved by modifying the mask $\boldsymbol{M}$ (see Algorithm 4.1). The positions of the eliminated connections and the newly generated connections are denoted as $\boldsymbol{E}$ and $\boldsymbol{G}$, respectively. The formula for pruning and regrowth can be expressed as follows:

$$\boldsymbol{W}_S \leftarrow \boldsymbol{W}_S \odot (\boldsymbol{M}-\boldsymbol{E}),\ \boldsymbol{W}_S \leftarrow \boldsymbol{W}_S \odot (\boldsymbol{M}+\boldsymbol{G}). \tag{4.3}$$

By doing so, the sparsity of each spatial group is kept steady. Moreover, the weights can be adjusted dynamically, allowing for improved local characteristics. Please refer to Figure 4.3 for details. Once training is complete, mask M in spatial groups is also recorded. Unlike previous methods [161], we employ the *Spatial-wise Group Convolution* and partition dynamic sparse processes into separate spatial groups. Pruning and growth are conducted independently within each group without interfering with each other.

We can assume that weights are changing factors over time. Then, removing the least important weights is akin to the selection phase in natural evolution. Alternatively, the random addition of new weights is analogous to the alteration stage of evolutionary selection [163]. This phenomenon is also analogous to a biological process in the brain during sleep, known as synaptic shrinking. Researchers found that the weakest neural link in the brain weakens during slumber, while the vital neural connections remain how they are before. This shows that one of the main roles of sleeping is to reset the overall synaptic strength [221, 222].

4.2.4 Channel-wise Weight Selection (CWS)

Despite SLaK [161] employs DST to enlarge the 2D kernel size, it mitigates the performance degradation caused by sparsity in the way of expanding model width. It strives to strike a balance between sparsity and width. However, expanding width leads to the issue of increased computational burden. Therefore, SLaK [161] faces the dilemma that larger sparsity and smaller model size cannot be achieved simultaneously. Instead of naively increasing network width, we propose CWS to decouple sparsity and width. It selectively chooses the most salient channels and obtains improved performance while keeping the original model size during inference [215].

CWS operates in an online mode with a sorting frequency of f_s. A single weight sorting cycle involves multiple iterations of *Sparse Weight Pruning and Growth*, *i.e.*, each cycle of weight sorting is a multiple of the adaptation rate f_a. The width factor w determines the extent to which the number of channels is augmented, *i.e.*, extending D-dimension to $w \times D$-dimension. Once the expanded model has been created, we consider the D-dimensional small model to be embedded within the $w \times D$-dimensional augmented model [215]. When the number of training iterations reaches the sorting condition, the channels are sorted based on their L1 value (*i.e.*, SortChannels), prompting the model to focus on the most relevant channels. During validation, we select the most important D-dimensional channels from $w \times D$-dimensional channels [215] (*i.e.*, SelectChannels). The pseudo-code for the entire algorithm is provided in Algorithm 4.1. This operation effectively

Algorithm 4.1 Pseudo-code of SDS and CWS

```
Require: set adaptation frequency $f_a$, sparsity $s$; set sorting frequency $f_s$ and width factor $w$;
         expand model width form $D$ dimension to $w \times D$ dimension;
         initialize dense model $\boldsymbol{W}_D^{w\times D}$; initialize sparse layers $\boldsymbol{W}_S^{w\times D}$;
Ensure: a network learned with 3D sparse convolution kernels.
  for each training iteration $i$ do
      $\boldsymbol{W}_S^{w\times D} \leftarrow$NormalTraining($\boldsymbol{W}_S^{w\times D}$);
      if ($i\%f_a$) equals to 0 then
          $\boldsymbol{W}_S^{w\times D} \leftarrow \boldsymbol{W}_S^{w\times D} \odot (\boldsymbol{M}-\boldsymbol{E})$; $\boldsymbol{W}_S^{w\times D} \leftarrow \boldsymbol{W}_S^{w\times D} \odot (\boldsymbol{M}+\boldsymbol{G})$;
      end if
      if ($i\%f_s$) equals to 0 then
          $\boldsymbol{W}_S^{w\times D} \leftarrow$SortChannels($\boldsymbol{W}_S^{w\times D}$);
      end if
      if validation then
          $\boldsymbol{W}_S^{w\times D} \leftarrow$SortChannels($\boldsymbol{W}_S^{w\times D}$); $\boldsymbol{W}_S^{D} \leftarrow$SelectChannels($\boldsymbol{W}_S^{w\times D}$);
          NormalValidation($\boldsymbol{W}_S^{D}$);
      end if
  end for
```

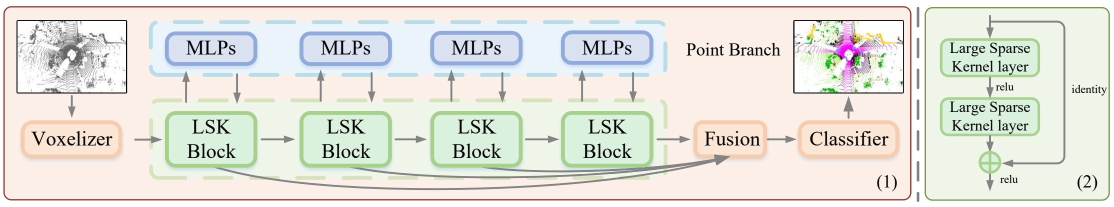


FIGURE 4.4: **LSK3DNet and LSK Block.** (1) LSK3DNet: Point clouds are fed into voxelizer for voxel-wise features. Then we extract features with LSK Block and Point Branch. The final prediction is a point-wise output. Here MLPs is the standard Multi-Layer Perceptrons. (2) Large Sparse Kernel Block (LSK Block) (Section 4.2.5).

achieves higher performance while maintaining the parameters within the expected size. This is especially important in deployments where memory usage and computational efficiency are critical factors.

4.2.5 Network Architecture

Segmentation Baseline. U-Net type 3D networks (such as SparseConvNet [82] and MinkUNet [18]) employ aggressive downsampling (*i.e.*, regular sparse convolution) to increase the receptive field but at the cost of reduced resolution. However, with a low resolution, several points or tiny objects could be combined into one single grid and become indistinguishable [87]. So SPVCNN [87] is equipped with a high-resolution point-based branch. Subsequently, 2DPASS [92] further modifies multi-representation branches by omitting the regular sparse convolution. This is because it dilates all sparse features and blurs valuable information, increasing the burden for following layers [88]. Only submanifold sparse convolution is employed for feature extraction in Modified SPVCNN [92]. By limiting the output feature positions to the input positions, submanifold sparse convolution is able to circumvent the computation burden. We use Modified SPVCNN as a baseline for our segmentation network. This 3D network is compact yet powerful and can generate high-resolution representations from sparse point clouds in large-scale scenes. However, one challenge we face is that the restricted area of submanifold sparse convolution limits the information flow and makes it difficult to distinguish different spatial characteristics of the scene. To address this challenge, we introduce LSK Block, which stands for Large Sparse Kernel Block. This block increases the kernel sizes of submanifold sparse convolution and expands the receptive field to facilitate information flow. LSK Block has a standard residual structure that adds the output of identity mapping to that of two stacked large kernel convolutions (Figure 4.4 (2)). Our network does not need the parallel convolutional branch with dilatation convolution [3]. The details of our segmentation network architecture are listed below:

- **Segmentation Network on SemanticKITTI [1].** LSK3DNet (see Figure 4.4 (1)) divides the entire scene into voxels, each with a size of 5 cm. It has four scales of $\{2, 4, 8, 16\}$. The hiden size D of the entire network is 64. We deploy LSK Blocks in *SparseBasicBlock*[1]. Following [6, 47, 92], we employ the weighted cross-entropy loss to optimize point accuracy and utilize the lovasz-softmax [191] loss to maximize the intersection-over-union.
- **Segmentation Network on ScanNet v2 [9].** This segmentation network has the same architecture as that on SemanticKITTI [1]. The entire scene is split with a voxel size of 2 cm, with scales of $\{2, 4, 8, 16, 16\}$. The hidden size D is 128. Following [99], we take the weighted cross-entropy loss as the objective function.

Detection Network on KITTI. We take Voxel R-CNN [108] as a baseline. Specifically, we retain the backbone of Voxel R-CNN and substitute plain sparse convolutional block with LSK Block in the first three stages. Other settings remain the same as [3]. The input spatial shape is $\{1600, 1408, 41\}$, and the channels of stages are $\{16, 16, 32, 64, 64\}$.

[1]`https://github.com/yanx27/2DPASS`

4.3 Experiment

4.3.1 Setups and Implementations

Dataset. To evaluate our method, we perform experiments on three benchmark datasets and five tracks:

- **SemanticKITTI** [1] has 43,551 traffic point cloud scenes with fine annotations, split into 19,130/4,071/20,350 scenes for `train`/`val`/`test`. The dataset has 28 semantic classes, but only 19 classes are used for single-scan track and 25 classes for multi-scan track.
- **ScanNet v2** [9] contains 1,201/312/100 indoor scenes for `train`, `val`, and `test` splits, respectively. There are 20 semantic categories for ScanNet20 track and 200 categories for ScanNet200 track.
- **KITTI** [8] has 7,481/7,518 samples for `train` and `test`. We follow the frequently used `train`/`val` split [223] to divide the training samples into *train* split (3,712 samples) and *val* split (3,769 samples).

Training and Testing Details. We utilize AdamW optimizer with OneCycleLR scheduler starting with a learning rate of 5e-3. Data augmentation is also employed, such as random flipping, scaling, rotation around the gravity axis, spatial translation. We apply instance CutMix [91] and Test Time Augmentation (TTA) [92] to the SemanticKITTI *test* benchmark, and enhance the model with extra training epochs.

Metrics. We employ mean class intersection over union (mIoU) and overall accuracy (Acc) metrics as the evaluation criterion for 3D semantic segmentation tasks, as outlined in [1]. What is more, we calculate Average Precision (AP) by recalling 11 positions for 3D object detection.

4.3.2 3D Semantic Segmentation

Quantitative Results on SemanticKITTI. We test LSK3DNet on both single-scan and multi-scan tracks [1]. Table 4.1 presents the quantitative results of SemanticKITTI single-scan track. LSK3DNet outperforms 2DPASS [92] in terms of both mIoU and IoU scores for most categories. Moreover, SDS reduces the model size of naive 3D kernels and CWS keeps the model size within the

TABLE 4.1: **Results** of LSK3DNet on SemanticKITTI [1] single-scan *test* (Section 4.3.2). Regarding input data format, P denotes points, V represents voxelizations, R signifies range images, and 2DPASS incorporates additional 2D data.

Methods	Input	**mIoU**	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign
PointNet++ [15]	P	20.1	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9
TangentConv [73]	R	40.9	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1	79.5	49.3	58.1	23.0	28.4	8.1	49.0	35.8	28.5
PolarNet [71]	R	54.3	90.8	74.4	61.7	21.7	90.0	93.8	22.9	40.3	30.1	28.5	84.0	65.5	67.8	43.2	40.2	5.6	61.3	51.8	57.5
RandLA-Net [213]	P	55.9	90.5	74.0	61.8	24.5	89.7	94.2	43.9	29.8	32.2	39.1	83.8	63.6	68.6	48.4	47.4	9.4	60.4	51.0	50.7
SqueezeSegV3 [79]	R	55.9	91.7	74.8	63.4	26.4	89.0	92.5	29.6	38.7	36.5	33.0	82.0	58.7	65.4	45.6	46.2	20.1	59.4	49.6	58.9
KPConv [64]	P	58.8	90.3	72.7	61.3	31.5	90.5	95.0	33.4	30.2	42.5	44.3	84.8	69.2	69.1	61.5	61.6	11.8	64.2	56.4	47.4
JS3C-Net [202]	V	66.0	88.9	72.1	61.9	31.9	92.5	95.8	54.3	59.3	52.9	46.0	84.5	69.8	67.9	69.5	65.4	39.9	70.8	60.7	68.7
SPVNAS [87]	PV	67.0	90.2	75.4	67.6	21.8	91.6	97.2	56.6	50.6	50.4	58.0	86.1	73.4	71.0	67.4	67.1	50.3	66.9	64.3	67.3
Cylinder3D [6]	V	68.9	92.2	77.0	65.0	32.3	90.7	97.1	50.8	67.6	63.8	58.5	85.6	72.5	69.8	73.7	69.2	48.0	66.5	62.4	66.2
RPVNet [91]	RPV	70.3	93.4	80.7	70.3	33.3	93.5	97.6	44.2	68.4	68.7	61.1	86.5	75.1	71.7	75.9	74.4	43.4	72.1	64.8	61.4
(AF)^2-S3Net [98]	V	70.8	92.0	76.2	66.8	45.8	92.5	94.3	40.2	63.0	81.4	40.0	78.6	68.0	63.1	76.4	81.7	77.7	69.6	64.0	73.3
PV-KD [203]	V	71.2	91.8	77.5	70.9	41.0	92.4	97.0	53.5	67.9	69.3	60.2	86.5	73.8	71.9	75.1	73.5	50.5	69.4	64.9	65.8
2DPASS [92]	PV	72.9	89.7	74.7	67.4	40.0	93.5	97.0	61.1	63.6	63.4	61.5	86.2	73.9	71.0	77.9	81.3	74.1	72.9	65.0	70.4
SphereFormer [217]	V	74.8	91.8	78.2	69.7	41.3	93.8	97.5	59.6	70.1	70.5	67.7	86.7	75.1	72.4	79.0	80.4	75.3	72.8	66.8	72.9
LSK3DNet	PV	**75.6**	92.2	78.9	70.2	41.8	92.7	97.3	61.0	71.4	75.6	64.2	86.4	72.7	71.9	81.2	80.6	85.2	72.0	67.0	74.6

TABLE 4.2: **Results** of LSK3DNet on SemanticKITTI [1] multi-scan *test* set. The arrow below classes indicate moving classes (Section 4.3.2).

Method	Input	**mIoU**	**Acc**	car $\uparrow$	truck $\uparrow$	other-vehicle $\uparrow$	person $\uparrow$	bicyclist $\uparrow$	motorcyclist $\uparrow$
LatticeNet [224]	P	45.2	89.3	54.8	3.5	0.6	49.9	44.6	64.3
TLSeg [22]	R	47.0	89.6	68.2	2.1	12.4	40.4	42.8	12.9
KPConv [64]	P	51.2	89.3	69.4	5.8	4.7	67.5	67.4	47.2
Cylinder3D [6]	V	52.5	91.0	74.9	0.0	0.1	65.7	68.3	11.9
(AF)^2-S3Net [98]	V	56.9	88.1	65.3	5.6	3.9	67.6	66.4	59.6
2DPASS [92]	PV	62.4	91.4	82.1	16.1	3.8	80.3	71.2	73.1
LSK3DNet	PV	**63.4**	**92.2**	84.4	7.2	40.9	77.4	69.9	72.1

desired size. So, LSK3DNet has a faster running speed than most prior methods (89 milliseconds per scene, see Figure 4.2.), but it's performance significantly benefits from the large kernels. Table 4.2 shows the results of SemanticKITTI multi-scan track. The mIoU and Acc are computed over all 25 classes. Due to page limitations, we only report the per-class IoUs for moving categories. Under this challenging setting, LSK3DNet surpasses 2DPASS [92] in both mIoU and Acc. Notably, LSK3DNet achieves state-of-the-art performance on both single-scan and multi-scan tracks in SemanticKITTI. Surpassing over the state of the art (SOTA) 2D-3D method 2DPASS is more valuable, which utilizes both 2D and 3D data while our LSK3DNet takes as input only the 3D data. Qualitative results can be found in Appendix B.

TABLE 4.3: **Results** of LSK3DNet and other state-of-the-art methods on ScanNet v2 [9] (Section 4.3.2).

Method	Input	ScanNet20	ScanNet20	ScanNet200
		Val	*Test*	*Val*
PointNet++ [15]	P	53.5	55.7	-
3DMV [225]	P	-	48.4	-
PanopticFusion [226]	P	-	52.9	-
PointCNN [39]	P	-	45.8	-
PointConv [227]	P	61.0	66.6	-
JointPointBased [228]	P	69.2	63.4	-
PointASNL [229]	P	63.5	66.6	-
SegGCN [230]	P	-	58.9	-
RandLA-Net [47]	P	-	64.5	-
KPConv [64]	P	69.2	68.6	-
JSENet [231]	P	-	69.9	-
FusionNet [86]	P	-	68.8	-
Point Transformer [7]	P	70.6	-	-
Fast Point Transformer [100]	P	72.1	-	-
Stratified Transformer [207]	P	74.3	73.7	-
Point Transformer v2 [99]	P	75.4	75.2	31.9
SparseConvNet [82, 99]	V	69.3	72.5	28.8
MinkUNet [18]	V	72.2	73.6	-
LargeKernel3D [3]	V	73.5	73.9	-
LSK3DNet	PV	**75.7**	**75.5**	**33.1**

TABLE 4.4: **Semantic segmentation results** on NuScenes `validation` set [10]. Regarding input data format, P denotes points, V represents voxelizations, R signifies range images, 2D3DNet and 2DPASS incorporate additional 2D data. mIoU (%) and IoUs (%) are reported.

Method	Input	**mIoU**	barrier	bicycle	bus	car	construction	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other flat	sidewalk	terrain	manmade	vegetation
(AF)^2-S3Net [98]	V	62.2	60.3	12.6	82.3	80.0	20.1	62.0	59.0	49.0	42.2	67.4	94.2	68.0	64.1	68.6	82.9	82.4
RangeNet++ [75]	R	65.5	66.0	21.3	77.2	80.9	30.2	66.8	69.6	52.1	54.2	72.3	94.1	66.6	63.5	70.1	83.1	79.8
PolarNet [71]	R	71.0	74.7	28.2	85.3	90.9	35.1	77.5	71.3	58.8	57.4	76.1	96.5	71.1	74.7	74.0	87.3	85.7
Salsanext [78]	R	72.2	74.8	34.1	85.9	88.4	42.2	72.4	72.2	63.1	61.3	76.5	96.0	70.8	71.2	71.5	86.7	84.4
AMVNet [232]	P	76.1	79.8	32.4	82.2	86.4	62.5	81.9	75.3	72.3	83.5	65.1	97.4	67.0	78.8	74.6	90.8	87.9
Cylinder3D [6]	V	76.1	76.4	40.3	91.2	93.8	51.3	78.0	78.9	64.9	62.1	84.4	96.8	71.6	76.4	75.4	90.5	87.4
RPVNet [91]	RPV	77.6	78.2	43.4	92.7	93.2	49.0	85.7	80.5	66.0	66.9	84.0	96.9	73.5	75.9	76.0	90.6	88.9
PVKD [203]	V	76.0	76.2	40.0	90.2	94.0	50.9	77.4	78.8	64.7	62.0	84.1	96.6	71.4	76.4	76.3	90.3	86.9
2D3DNet [233]	V	79.0	78.3	55.1	95.4	87.7	59.4	79.3	80.7	70.2	68.2	86.6	96.1	74.9	75.7	75.1	91.4	89.9
2DPASS [92]	PV	79.4	78.8	49.6	95.6	93.6	60.0	84.1	82.2	67.5	72.6	88.1	96.8	72.8	76.2	76.5	89.4	87.2
SphereFormer [217]	V	79.5	78.7	46.7	95.2	93.7	54.0	88.9	81.1	68.0	74.2	86.2	97.2	74.3	76.3	75.8	91.4	89.7
LSK3DNet	PV	**80.1**	80.0	52.5	94.5	91.8	58.8	85.8	84.4	71.2	73.8	88.3	96.9	74.8	75.9	75.9	89.3	87.5

Quantitative Results on ScanNet V2. We compare LSK3DNet with previous state-of-the-art methods on ScanNet v2 [9], a large-scale dataset for 3D indoor scene segmentation. ScanNet v2 has two tracks: ScanNet20 and ScanNet200, which use 20 and 200 semantic classes respectively. Table 4.3 presents the quantitative results of our model and other methods on both tracks. LSK3DNet achieves higher performance than previous methods in both tracks. Our performance is superior to transformer-based methods [7, 99, 100, 207], including Point Transformer v2. Moreover,

TABLE 4.5: **Results of 3D object detection methods** on the car class of KITTI *val* set [8] (Section 4.3.3).

Method	Input	3D AP (IoU=0.7)		
		Easy	Moderate	Hard
VoxelNet [234]	V	81.97	65.46	62.85
PointPillars [102]	R	86.62	76.06	68.91
SECOND [89]	V	88.61	78.62	77.22
Point R-CNN [111]	P	88.88	78.63	77.38
Part-A^2 [235]	V	89.47	79.47	78.54
PV-RCNN [211]	PV	89.35	83.69	78.70
Focals Conv [88]	V	89.52	84.93	79.18
Voxel R-CNN [108]	V	89.41	84.52	78.93
LargeKernel3D [3]	V	89.52	85.07	79.32
LSK3DNet	V	**90.16**	**85.61**	**79.53**

LSK3DNet has a clear advantage over LargeKernel3D [3], improving the mIoU by 2.2% and 1.6% on the ScanNet v2 *val* and *test* set, respectively.

Quantitative Results on NuScenes. In Table 4.4, we present the results of semantic segmentation on the nuScenes validation set [10]. Our approach consistently surpasses others by a significant margin, establishing itself as the SOTA performer on this benchmark. What's particularly intriguing is that our method relies solely on LiDAR data, yet it outperforms multi-modal techniques [92, 233], which incorporate supplementary 2D information.

4.3.3 3D Object Detection

We have also evaluated the detection performance of LSK3DNet on the *val* split for car category, as shown in Table 4.5. We report the average precision metric for a 3D bounding box with 11 recall positions (Recall 11). Based on Voxel R-CNN [108], we showcase the effectiveness of the LSK Block by comparing it with LargeKernel3D [3], a previous 3D large-kernel method. LSK3DNet achieves better results in three difficulty levels, compared with [3, 108]. Qualitative results can be found in Appendix B.

4.3.4 Ablation Studies

We first explore the effect of kernel size on performance, and then the overall architecture design. Next, we explain the hyperparameter choices of SDS and CWS. All ablation experiments are performed on the *val* set of SemancKITTI.

TABLE 4.6: **Segmentation performance of Modified SPVCNN with various large kernel settings**. "Dense" refers to directly enlarging the kernel size. "Ours" refers to training with SDS and CWS (Section 4.3.4). The unit for "Param" is in million (M), "FLOPs" is in billion (G), and "Speed" is measured in milliseconds per scene.

Method	Kernel Size	mIoU	Param	FLOPs	Speed ↓
Dense ($D = 64$)	$3, 3, 3$	65.2	1.9	78.4	36
	$5, 5, 5$	65.6	8.4	127.8	57
	$7, 7, 7$	66.2	22.6	381.2	65
	$9, 9, 9$	67.5	47.9	1916.3	93
	$11, 11, 11$	66.6	87.4	3192.0	134
Dense ($1.8 \times D$)	$5, 5, 5$	66.2	27.0	444.0	84
	$7, 7, 7$	66.5	73.1	1551.2	123
	$9, 9, 9$	67.2	154.7	3030.0	177
	$11, 11, 11$	66.3	282.1	7139.2	250
Ours ($1.8 \times D$)	$5, 5, 5$	66.8	5.1	203.8	46
	$7, 7, 7$	68.1	13.6	249.4	63
	$\mathbf{9, 9, 9}$	**70.2**	28.8	763.6	89
	$11, 11, 11$	70.1	52.5	1847.4	116

Kernel Size. We use Modified SPVCNN as the baseline and then explore 3D kernel sizes under two settings: naive dense large kernel and our LSK3DNet. In the former setting, the dense 3D kernel is straightforwardly expanded. In the latter one, LSK3DNet is trained simultaneously with SDS and CWS, meaning that our LSK3DNet can learn a large sparse kernel model from the beginning.

The baseline does not use aggressive downsampling which is common in most U-type 3D networks [18, 82]. However, submanifold sparse convolution with small kernels may lose important information flow, especially for the spatially disconnected features. We solve this problem by enlarging the size of the 3D convolution kernel to obtain a large receptive field. This agrees with the performance improvement in Table 4.6, when the kernel size increases. In contrast to this observation, LargeKernel3D [3] shows the opposite trends; we empirically attribute this to the fact that LargeKernel3D is based on U-type 3D networks (Section 4.2.5).

Compared to the "naive dense large kernel" of each kernel size, LSK3DNet exhibits superior performance in its corresponding size. LSK3DNet not only leverages SDS to expand its receptive field but also starts from scratch to learn 3D sparse kernels. Concurrently, CWS widens the network during training while pruning redundant channels to accelerate inference. Furthermore, SDS and CWS address the issues of overparameterization and overfitting that arise from simply increasing the kernel's size and expanding the model's width. The best performance of LSK3DNet is achieved at the $9 \times 9 \times 9$ size, and the following experiments are conducted with $9 \times 9 \times 9$ size. Moreover, due to the extra dimension in 3D kernels compared to 2D kernels, naively enlarging the 3D kernel

TABLE 4.7: **Overall architecture design with $9 \times 9 \times 9$ size**. Here, we report Param, FLOPs, and Speed for the inference model (Section 4.3.4).

Method	mIoU	Param	FLOPs	Speed $\downarrow$
Dense $(1.0 \times D)$	67.5	47.9	1916.3	93
Dense $(1.8 \times D)$	67.2	154.7	3030.0	177
SDS $(1.8 \times D)$	69.3	93.0	2201.4	173
CWS $(1.8 \times D)$	69.1	47.9	1916.3	93
LSK3DNet (1.8×D)	**70.2**	**28.8**	**763.6**	**89**

TABLE 4.8: **Training and inference speed analysis, speed is measured in milliseconds per scene**. The kernels have $9 \times 9 \times 9$ size, "T" and "I" represent training and inference speed, respectively (Section 4.3.4).

Sparsity	Dense $(1.0 \times D)$			SDS $(1.8 \times D)$			LSK3DNet $(1.8 \times D)$		
s	mIoU	T $\downarrow$	I $\downarrow$	mIoU	T $\downarrow$	I $\downarrow$	mIoU	T $\downarrow$	I $\downarrow$
0	67.5	451	93	-	-	-	-	-	-
0.1	-	-	-	68.5	489	176	69.1	491	92
0.2	-	-	-	68.9	478	176	69.7	482	91
0.4	-	-	-	69.3	463	173	**70.2**	467	89
0.6	-	-	-	69.0	450	171	68.8	454	87
0.8	-	-	-	68.7	439	171	67.2	443	87

size will lead to a cubically-increasing overhead. Our LSK3DNet can effectively reduce 19.1M parameters and approximately 60% of computing operations when compared to the naive large kernel network. This aspect significantly enhances the value of SDS and CWS for 3D kernels.

Overall Architecture Design. In Table 4.7, Dense $(1.8 \times D)$ does not yield improvement due to overparameterization and overfitting. Although SDS maintains a model width of $1.8 \times D$ during inference, it achieves improved performance due to sparse training and the large receptive field. CWS effectively addresses overfitting by selecting salient channels, thereby enhancing performance. By combining SDS and CWS, LSK3DNet can achieve the best performance while significantly reducing the model size and computational cost.

Spatial-wise Dynamic Sparsity. To control the degree of sparse weight adaptation in LSK3DNet, we have adjusted three hyperparameters: the adaptation frequency f_a, sparsity rate s, and prune rate p (Section 4.2.3). Adaptation frequency f_a indicates how often we update the sparse weights during training. Sparsity rate s specifies how sparse the 3D large kernels are. Prune rate p shows the fraction of updated weights in one adaptation. We have conducted experiments with different values of these hyperparameters and report the results in Table 4.8 and Table 4.9. Based on our empirical findings, we have chosen $f_a = 2000$, $s = 0.4$, and $p = 0.3$ as the optimal settings for LSK3DNet.

Channel-wise Weight Selection. Another aspect of our method that we investigated is the CWS. This technique involves two hyperparameters: the sorting frequency f_s and the width factor w

TABLE 4.9: **Ablation studies** of *Spatial-wise Dynamic Sparsity* on adaptation frequency f_a and prune rate p (Section 4.3.4).

Adaptation f_a	mIoU
100	68.8
1000	69.6
2000	**70.2**
3000	70.0
4000	69.8

Pruning p	mIoU
0.10	69.9
0.20	70.1
0.30	**70.2**
0.50	70.0
0.70	69.7

TABLE 4.10: **Ablation studies** of *Channel-wise Weight Selection* on sorting frequency f_s and width factor w (Section 4.3.4).

Sorting f_s	mIoU
$1 \times f_a$	68.2
$2 \times f_a$	69.1
$4 \times f_a$	69.8
$\mathbf{6 \times f_a}$	**70.2**
$10 \times f_a$	69.9

Width w	mIoU
$1.1\times$	68.4
$1.5\times$	69.6
$\mathbf{1.8\times}$	**70.2**
$2.1\times$	69.9
$2.5\times$	70.1

(Section 4.2.4). The former determines how frequently we sort and select channels for large kernels during training. It is expressed as a multiple of f_a, the adaptation frequency, meaning that we perform channel sorting after a certain number of weight adaptation cycles. When $f_s = 6 \times f_a$, the optimal performance is achieved. The latter controls the times of network width compared to the target model. We find that increasing the width factor up to $1.8\times$ improves the model's performance, but beyond that point, there is no significant gain. Therefore, we opt for a width of $1.8\times$ to avoid more extra compute resources that a wider width would require during training.

Spatial Group Size. We report the results of the ablation study on the kernel size selection in Table 4.11. *kernel size* means the absolute kernel size. *group* means the number of groups to split kernels. We conduct the experiment on LSK3DNet on the SemanticKITTI dataset, with the same training hyper-parameters in the main paper. For 3×3×3 *group* and 9×9×9 *kernel size*, every group has $\{3,3,3\}$ divisions in each dimension. Moreover, $9\times9\times9$ is split into $5\times5\times5$ with $\{2,2,1,2,2\}$ divisions. For 13×13×3 *kernel size*, 3×3×3 *group* has $\{4,5,4\}$ divisions in 13-dimensional axis, and similarly, $\{2,3,3,3,2\}$ divisions for $5\times5\times3$ *group*, $\{2,2,2,1,2,2,2\}$ divisions for $7\times7\times3$ *group*. We finally chose the $9\times9\times9$ *kernel size* and $3\times3\times3$ *group* based on our empirical results. We speculate that this is because each group has a $3\times3\times3$ size, showing more parameter space to explore than other *group* settings. Previous studies have shown that exploring a large parameter space is important for dynamic sparse training [168, 170, 175]. In addition, the kernel size of $13\times13\times3$ does not achieve better performance, being affected by overparameterization and overfitting issues.

TABLE 4.11: **Ablation studies** on spatial group size.

kernel size	*Spatial Group*	mIoU (%)
9×9×9	3×3×3	70.2
	5×5×5	69.1
13 × 13 × 3	3×3×3	69.9
	5×5×3	69.5
	7×7×3	68.9

TABLE 4.12: **Ablation studies** on *Depth-wise Group Convolution* and *Spatial-wise Group Convolution* on ScanNet v2 [9] and SemanticKITTI [1].

Methods	Kernel	mIoU (%)
ScanNet v2		
MinkowskiNet-14 [3]	7×7×7	68.6
+ *depth-wise* conv	7×7×7	56.4
+ *spatial-wise* conv	7×7×7 → 3×3×3	69.7
+ *spatial-wise* conv	9×9×9 → 7×7×7	69.6
MinkowskiNet-34 [3]	7×7×7	68.6
+ *depth-wise* conv	7×7×7	68.7
+ *spatial-wise* conv	7×7×7 → 3×3×3	73.2
Modified SPVCNN	9×9×9	72.4
+ *depth-wise* conv (SDS and CWS)	9×9×9	67.0
+ *spatial-wise* conv (SDS and CWS)	9×9×9	75.7
SemanticKITTI		
Modified SPVCNN	9×9×9	67.5
+ *depth-wise* conv (SDS and CWS)	9×9×9	65.4
+ *spatial-wise* conv (SDS and CWS)	9×9×9	70.2

Convolutional Schemes. The results of MinkowskiNet-14 and MinkowskiNet-34 are directly taken from LargeKernel3D [3]. MinkowskiNet-14 shows stagnation when attempting to expand the convolution kernel to $9\times9\times9$. As shown in Table 4.12, we use Modified SPVCNN as the baseline and conduct experiments on ScanNet v2 [9] and SemanticKITTI [1]. Our findings indicate that *Spatial-wise Group Convolution* is more suitable for 3D large kernel designs, aligning with the results observed in LargeKernel3D [3]. Therefore, we carry out dynamic sparse training within the spatial-wise groups.

Effective Receptive Fields (ERFs) size. The concept of the receptive field is important for deep CNNs work, as anywhere in an input scene outside the receptive field of a unit does not affect the value of that unit. The small nucleus in the baseline has a smaller receptive field, and the restricted areas of submanifold sparse convolutions limit the informationflow and make it difficult to distinguish different spatial characteristics of the scene. To address this challenge, we increase the kernel sizes of submanifold sparse convolutions and expand the receptive field to facilitate information flow. To evaluate this hypothesis, we make a comparison between the ERF captured by

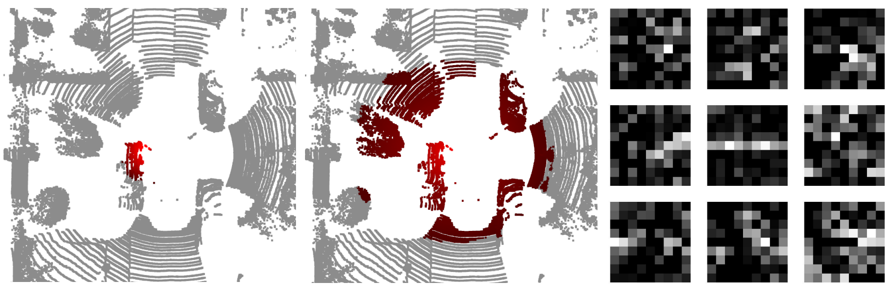


FIGURE 4.5: **ERFs of Baseline and LSK3DNet**. LSK3DNet has a larger ERF size. Additionally, we provide visualization of learned sparse kernels, where all weight values have taken the absolute value and normalization operations. The black areas indicate positions with weight values of zero.

Baseline and LSKNet. We visualize ERF on the SemanticKITTI validation set. Figure 4.5 illustrates the comparison between the Baseline and LSK3DNet. The left one is the ERFs of Baseline, and the right one is the ERF of our LSKNet. We follow the definition of ERF [236], and measure the gradient of input point data to the central point of the features generated in the last layer. We take the point closest to the origin as the central point. We normalize the gradient of each point to [0,1] and the contribution of points is shown in BEV. Compared to the baseline, the high-contribution points of LSK3DNet are distributed over a larger input range, indicating a larger ERF. Additionally, to enhance comprehension of learned kernels, we offer kernel visualization in Figure 4.5, providing insights into sparse training.

Training Stability. Compared with the baseline, LSK3DNet achieves performance improvements. However, due to the adoption of a larger kernel design, dynamic sparse training, and the operation of channel sorting in LSK3DNet, it converges $\sim$5 epochs later than the baseline. Nevertheless, it can still be ensured that convergence is achieved before the end of 64 epochs.

4.4 Summary

We propose *Spatial-wise Dynamic Sparsity* to scale up 3D kernels beyond $9 \times 9 \times 9$, which prunes the volumetric weight and reduces the parameter size of large kernel layers. Our LSK3DNet can benefit from a large receptive field without increasing the computational cost compared to a naive 3D large kernel. *Channel-wise Weight Selection* expands the model width during training, and then sorts and selects important channels during validation to get a model of the expected size. In this way, we achieve *"using spatial sparse groups, expanding width without more parameters"*. We evaluate our method on the SemanticKITTI and achieve state-of-the-art performance. Our LSK3DNet also surpasses previous 3D large kernel methods on ScanNet v2 and KITTI.

Chapter 5

Interpretable3D: An Ad-Hoc Interpretable Classifier for 3D Point Clouds

5.1 Introduction

Over the past decade, significant progress has been made in Deep Neural Networks (DNNs). However, serious concerns have been raised over DNNs' safety and trustworthiness [176, 179], when applied to 3D real-world decision-critical scenarios, such as self-driving cars [127, 237, 238] and medical diagnosis systems. Interpretability has emerged as a critical concern regarding the trustworthiness of predictions generated by DNNs. However, understanding DNNs proves to be particularly challenging due to their complex, non-linear, and high-dimensional nature. They lack natural explanations that humans can easily comprehend. Similarly, the parametric softmax classifier learns highly abstract parameters and lacks a direct and intuitive interpretation [181, 239, 240]. Incorrect data fed into black-box models may lead to harmful consequences. As a result, the demand for interpretable 3D models has become increasingly urgent.

Extensive research has been conducted on the interpretability/explainability of 2D images [239, 241–243], but there is a lack of interpretable 3D point cloud research. Not to mention the few existing explanation studies on 3D models have been conducted with *post-hoc* explanations and only play

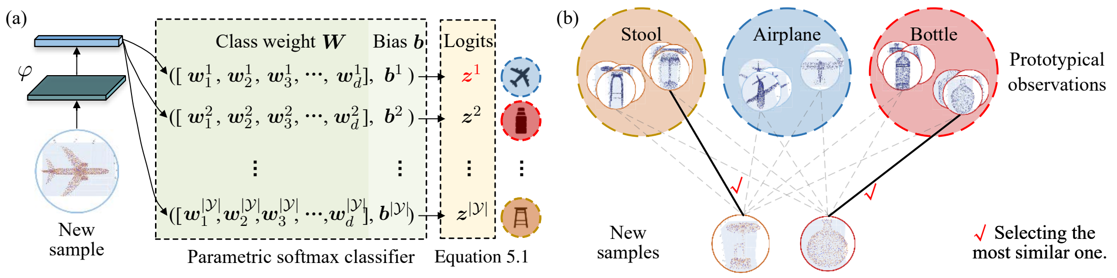

(a)
φ
New sample
Class weight $\boldsymbol{W}$
Bias $\boldsymbol{b}$
Logits
$([\boldsymbol{w}_1^1, \boldsymbol{w}_2^1, \boldsymbol{w}_3^1, \cdots, \boldsymbol{w}_d^1], \boldsymbol{b}^1)$
$([\boldsymbol{w}_1^2, \boldsymbol{w}_2^2, \boldsymbol{w}_3^2, \cdots, \boldsymbol{w}_d^2], \boldsymbol{b}^2)$
$([\boldsymbol{w}_1^{|\mathcal{Y}|}, \boldsymbol{w}_2^{|\mathcal{Y}|}, \boldsymbol{w}_3^{|\mathcal{Y}|}, \cdots, \boldsymbol{w}_d^{|\mathcal{Y}|}], \boldsymbol{b}^{|\mathcal{Y}|})$
z^1
z^2
$z^{|\mathcal{Y}|}$
Parametric softmax classifier
Equation 5.1
(b)
Stool
Airplane
Bottle
Prototypical observations
New samples
√ Selecting the most similar one.

FIGURE 5.1: (a) **Softmax-based DNN models** employ parametric classifiers. However, they lack a direct and intuitive interpretation of the decision-making processes. (b) **Interpretable3D** selects the most similar prototype (*i.e.*, the one with the maximum cosine similarity) for new samples.

an auxiliary role in 3D systems. For instance, saliency/attention maps [31, 32] have been applied to existing voxel-based 3D object detection networks. However, the *post-hoc* saliency maps cannot explain the roles of highlighted parts in the decision-making process [179]. The various saliency methods even produce conflicting saliency maps, making it difficult for researchers to determine which one actually reflects attention [176]. Moreover, *post-hoc* explanations are derived from a separate modeling process with strong priors. These priors, however, are not part of training [181]. The nuances (minor differences) between *post-hoc* modeling process and DNNs are also unknown [179]. So *post-hoc* analysis may result in problematic and misleading explanations [176, 178]. In contrast, *ad-hoc* interpretability methods can provide reliable explanations without any unknown nuances [179].

In this work, we design an inherently interpretable classifier for 3D point clouds (*i.e.*, Interpretable3D). It is inherently an intuitive case- or instance-based classifier, as it reveals what the representation means and how the embedding queries typical past observations from the prototype sets. As shown in Figure 5.1, new observations are classified based on their proximity to a prototype observation within the dataset. Therefore, Interpretable3D can explain its own reasoning process in a human-understandable way. The learned models naturally come with explanations for each prediction, and the explanations are loyal to what the network actually computes. Our Interpretable3D provides a level of interpretability that is absent in existing *post-hoc* 3D explanation models.

In each iteration, training sample embeddings are clustered and averaged to estimate prototypes (*i.e.*, Prototype Estimation). Compared to the softmax classifier, the mean of embeddings holds distinct statistical significance, representing class sub-centers. Furthermore, the estimated prototypes are then either penalized or rewarded based on their performance in the prediction (*i.e.*, Prototype

Optimization). By taking the within-class clustering as a dual *optimal transport problem* [16, 239], Prototype Estimation can proficiently uncover diverse intra-class variations in an online mode. Moreover, it can automatically extract discriminative prototypes to handle complex real-world disparities. The following Prototype Optimization can enhance the representativeness of the estimated prototypes by applying penalties or rewards. Furthermore, we enhance the interpretability of prototypes by updating them with their most similar observations in the final few epochs.

We evaluate Interpretable3D with four point cloud models: DGCNN [244] (Graph-based), PointNet2 [245], PointMLP [37], and PointNeXt [38] (MLP-based). Our experiments are conducted on three well-known public benchmarks (*i.e.*, ModelNet40 [4] and ScanObjectNN [5] for shape classification and ShapeNetPart [11] for part segmentation). Interpretable3D achieves comparable or even better performance than softmax-based black-box models. Additionally, our approach provides intrinsic interpretability to the classification and part segmentation results. This is a key advantage of our approach, as it allows for better transparency and comprehensibility of the AI decision-making process. This is our response to the current lack of *ad-hoc* interpretable research within the field of 3D vision.

5.2 Methodology

We develop an interpretable nearest neighbor based prototype classifier, the overview of which is illustrated in Figure 5.2. Suppose that the input is $\mathcal{X}=\{x_k : k = 1,2,\ldots,K\}$, *i.e.*, a set with K training samples $\{x_k : x_k \in \mathbb{R}^{N\times 3}\}$. Each sample consists of N points. $\mathcal{L}=\{l_k \in \mathcal{Y} : k = 1,2,\ldots,K\}$ is a set of object labels, where $\mathcal{Y}$ is the label set. The point-based methods $\phi \circ \varphi \colon \mathcal{X} \mapsto \mathcal{L}$ take $\mathcal{X}$ as input. $\varphi \colon \mathcal{X} \mapsto \mathcal{F}$ learns the underlying representations $\mathcal{F} = \{f_k : k = 1, 2, \ldots, K\}$, and then $\phi \colon \mathcal{F} \mapsto \mathcal{L}$ predicts the labels.

5.2.1 Preliminary: Softmax Classifier

Currently, the prevailing approach for ϕ is the parametric softmax classifier, it can be formulated as:

$$\hat{l} = \arg\max_{l\in\mathcal{Y}} z^l, \tag{5.1}$$

where $\hat{l}$ denotes class prediction; z^l is an unnormalized l class output of the last fully-connected (FC) layer, composing the logits vector $\boldsymbol{z} = [z^l]_l \in \mathbb{R}^{|\mathcal{Y}|}$. A softmax function is then applied to $\boldsymbol{z}$. While training, the learnable parameters in ϕ and φ are updated by minimizing the cross-entropy loss:

$$\arg\min_{w^*, b^*, \theta} \sum_{k=1}^{K} -\log \frac{\exp\left(\left(\boldsymbol{w}^{l_k}\right)^\top \varphi_\theta(x_k) + b^{l_k}\right)}{\sum_{l \in \mathcal{Y}} \exp\left(\left(\boldsymbol{w}^{l}\right)^\top \varphi_\theta(x_k) + b^{l}\right)}, \tag{5.2}$$

where the parameters θ are in φ, the weight matrix $\boldsymbol{W} = [\boldsymbol{w}^l]_l \in \mathbb{R}^{d \times |\mathcal{Y}|}$ and bias vector $\boldsymbol{b} = [b^l]_l \in \mathbb{R}^{|\mathcal{Y}|}$ are learnable parameters. The softmax classifier ϕ has been trained purely to optimize the accuracy. However, these learned highly abstract parameters are disconnected from the physical characteristics of the problem being modeled [240] and lack a direct or intuitive interpretation [181, 239]. From a human perspective, they do not provide any clues about what drives the classifier to reach its decisions.

5.2.2 Interpretable3D

Previous point cloud methods either utilize a non-transparent parametric softmax classifier or attempt to understand the network in *post-hoc* paradigms. However, these paradigms can be problematic and misleading. To offer a human-readable explanation for point clouds, we propose Interpretable3D. In contrast to the softmax classifier, Interpretable3D is built upon the intuitive concept of selecting the most similar prototype for new samples. Note that this contrast between parametric and non-parametric classifiers is important. It enables researchers to comprehend that ϕ can be learned without relying on non-transparent weight vectors.

Here we utilize S within-class prototypes for $\mathcal{Y}$ to represent past observations, *i.e.*, $|\mathcal{Y}| \times S$ prototypes $\boldsymbol{M} = [\boldsymbol{m}_s^l]_{l,s} \in \mathbb{R}^{d \times |\mathcal{Y}| \times S}$ in total. This is due to the presence of intra-class differences; each class cannot be accurately represented by just one prototype. In most cases, the number of prototypes surpasses that of categories [246]. Therefore, the prediction $\hat{l}$ is made by assigning the labels of the most similar prototypes to the samples [239]:

$$\hat{l} = l^*, \ \ (l^*, s^*) = \arg\max\nolimits_{l \in \mathcal{Y}, s \in \{1, \cdots, S\}} \langle f, \boldsymbol{m}_s^l \rangle, \tag{5.3}$$

where $\langle \cdot, \cdot \rangle$ is cosine similarity. Our model is fully *online* and *end-to-end* trained. We use the most standard and simple initialization strategy [247]: randomly selecting S data samples per class as the initial prototypes. During each training iteration, the prototypes are first updated and then

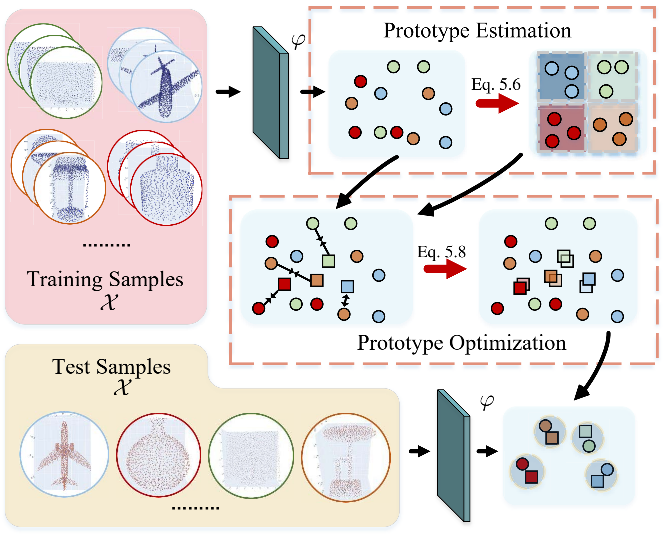


FIGURE 5.2: **The overview of our Interpretable3D**. At each training iteration, Interpretable3D first estimates the prototypes and then optimize them. Sample features are represented by points, and prototypes are represented by squares. In Prototype Estimation, sample features are regarded as an *optimal transport problem* (*i.e.*, Equation 5.6). Intra-class clustering is then conducted to estimate the mean of embeddings that are within the same subclass. In Prototype Optimization, prototypes that have a higher cosine similarity to the correct prediction receive a reward (→←), while those that are misclassified receive a punishment (↔).

optimized. Firstly, training samples within the same category are clustered and assigned sub-class labels. The $|\mathcal{Y}| \times S$ prototypes are updated with the mean of embeddings from the same subclass in a momentum manner. We then assign the labels of the most similar prototypes to the samples (Equation 5.3), and $\hat{l}$ is obtained. Next, Prototype Optimization involves penalizing or rewarding $\boldsymbol{m}_s^l$ according to $\hat{l}$ and l (Equation 5.7 and Equation 5.8). Moreover, in the final few epochs, the prototypes are updated with the features of the most representative training samples, rather than the mean of embeddings. As a result, the prototypes are connected to typical examples and can be seen as typical observations for classification. Finally, all the prototypes are stored as classification evidence for inference.

Interpretable3D predicts by evaluating the similarity between samples and prototypes, offering a clear depiction of the decision-making process. This aligns with the *prototype theory* in psychological cognitive sciences [248, 249]. In *prototype theory*, an object can be classified based on a single representative example or a set of items that are typical for the category [250].

5.2.3 Prototype Estimation

We cluster intra-class representations in the latent space $\mathcal{F}$ to mine typical prototypes. Suppose there are N^l samples for class l, underlying representation $\boldsymbol{F}^l \in \mathbb{R}^{d\times N^l}$ are mapped to prototypes $\boldsymbol{M}^l \in \mathbb{R}^{d\times S}$. We denote $\boldsymbol{A}^l \in \{0,1\}^{S\times N^l}$ as the assignment matrix of $\boldsymbol{F}^l$ among S subclasses. The (s,n)-*th* element of $\boldsymbol{A}^l$ indicates whether to assign the n-*th* feature of $\boldsymbol{F}^l$ to the s-*th* sub-cluster center. The optimization of $\boldsymbol{A}^l$ can be accomplished by taking the similarity between the representations and cluster centers as a cost matrix, *i.e.*, $\boldsymbol{Q}^l = \text{softmax}(-\boldsymbol{M}^{l\top}\boldsymbol{F}^l)$. Thus, the label assignment task can be viewed as an instance of the *optimal transport problem* [144, 239]:

$$\begin{aligned} \boldsymbol{A}^{l*} &= \underset{\boldsymbol{A}^l\geq 0}{\arg\min}\langle \boldsymbol{Q}^l, \boldsymbol{A}^l\rangle_F, \\ \text{s.t. } &\boldsymbol{A}^l\mathbf{1}_{N^l} = \boldsymbol{r}, \boldsymbol{A}^{l\top}\mathbf{1}_S = \boldsymbol{c}, \end{aligned} \tag{5.4}$$

where $\boldsymbol{A}^{l*}$ is the *optimal transportation plan* between the representations and cluster centers, which can be also viewed as the transportation plan of the sample and prototype. $\langle\cdot\rangle_F$ is the Frobenius dot-product. The vectors $\boldsymbol{r}$ and $\boldsymbol{c}$ are marginal projections of $\boldsymbol{A}^l$. They define constraints for $\boldsymbol{A}^l$, *i.e.*, $\boldsymbol{r} = \frac{1}{S}\mathbf{1}_S, \boldsymbol{c} = \frac{1}{N^l}\mathbf{1}_{N^l}$. The former equipartition constraint guarantees that, on average, each prototype is selected at least $1/S$ times. The latter unique assignment constraint ensures that every point is exclusively assigned to one prototype. The entropic regularization [198] of problem 5.4 is formulated as:

$$\begin{aligned} &\min_{\boldsymbol{A}^l\geq 0}\langle \boldsymbol{Q}^l, \boldsymbol{A}^l\rangle_F - \zeta H(\boldsymbol{A}^l), \\ \text{s.t. } &\boldsymbol{A}^l\mathbf{1}_{N^l} = \boldsymbol{r}, \boldsymbol{A}^{l\top}\mathbf{1}_S = \boldsymbol{c}, \zeta > 0, \end{aligned} \tag{5.5}$$

where ζ is the regularization parameter, and $H(\boldsymbol{A}^l)$ is the entropic regularization. Problem 5.5 is constrained by affine constraints, while the dual problem is unconstrained, making it simpler to design algorithms and analyze complexity [251]. With the Lagrangian function, the dual form [251] of problem 5.5 can be simplified as follows:

$$\min_{\boldsymbol{u}\in\mathbb{R}^S, \boldsymbol{v}\in\mathbb{R}^{N^l}} \left\{\mathbf{1}_S^\top \boldsymbol{A}^{l*}\mathbf{1}_{N^l} - \langle \boldsymbol{u}, \boldsymbol{r}\rangle_F - \langle \boldsymbol{v}, \boldsymbol{c}\rangle_F\right\}, \tag{5.6}$$

where $\boldsymbol{A}^{l*} = \text{diag}\left(e^{\boldsymbol{u}}\right) e^{-\boldsymbol{Q}^l/\gamma}\,\text{diag}\left(e^{\boldsymbol{v}}\right)$. $\boldsymbol{u}$ and $\boldsymbol{v}$ are two vectors of scaling coefficients, resulting from a small number of matrix multiplications. γ trades off convergence speed with closeness to the original transport problem. This task can be solved by Sinkhorn-Knopp algorithm [197] and APDAGD [251]. During training, the latent representation space $\mathcal{F}$ undergoes changes. This attribute means that the prototypes need be recalculated after each batch using the entire dataset.

However, this process can be costly and time-consuming. To address this, we employ a momentum update strategy [200]. It updates each prototype with the average of embeddings assigned to each sub-cluster of the training samples. However, in the last few epochs, we substitute the average of embeddings with the features of the closest training sample.

5.2.4 Prototype Optimization

Among prototype-based classification methods, a particularly attractive approach is Learning Vector Quantization (LVQ) [246, 252, 253]. The LVQ family employs the Winner-Takes-All (WTA) principle, where prototypes compete for updates according to labels and predictions [254, 255]. One fundamental design within LVQ family is the LVQ1 algorithm. This algorithm aims to rectify the decision boundary by adjusting the prototypes. To avoid suboptimal local minima or so-called "dead unit" in the WTA training process [247], we take the estimated prototypes $\boldsymbol{M}^l$ in Prototype Estimation as the initial set. Therefore, each class is also represented by S prototypes. Similar to LVQ1, we crisply assign each representation in $\boldsymbol{F}^l = [f^l \in \mathbb{R}^d]_l$ to the closest prototype $\boldsymbol{M}^w$, the so-called *winner*. Only the winning prototypes $\boldsymbol{M}^w$ are altered according to the *competitive learning* update equation:

$$\begin{aligned}\boldsymbol{M}^w &\leftarrow \boldsymbol{M}^w + \eta\,\psi(l,\hat{l}_w)\,(\boldsymbol{F}^l - \boldsymbol{M}^w),\\ \psi(l,\hat{l}_w) &= \begin{cases} +1 & \text{if } l = \hat{l}_w \\ -1 & \text{else} \end{cases},\end{aligned} \tag{5.7}$$

where η is the update rate, it controls the magnitude of updates; $\hat{l}_w$ denotes class prediction of the winning prototypes. This update strategy implements a rewarding mechanism for the *winner*, which is based on its ability to correctly classify the input, by migrating the *winner* towards $\boldsymbol{F}^l$. Conversely, it applies a punishment to the *winner* when it fails to correctly label the input, *i.e.*, moving the *winner* away from $\boldsymbol{F}^l$. Upon repeated presentation of each training sample, the *winner* is moved in the direction of the example feature if they share the same label and in the opposite direction if they don't. An enhanced LVQ algorithm, known as LVQ2.1 [252, 253], is often favored because it is effective in *Bayesian decision theory*. Starting with properly estimated initial prototypes, we can update $\boldsymbol{M}^w$ as follows:

$$\begin{cases} \boldsymbol{M}_p^w \leftarrow \boldsymbol{M}_p^w - \eta \left(\boldsymbol{F}^l - \boldsymbol{M}_p^w\right), \\ \boldsymbol{M}_q^w \leftarrow \boldsymbol{M}_q^w + \eta \left(\boldsymbol{F}^l - \boldsymbol{M}_q^w\right), \end{cases}$$
$$\text{if } \min\big(\frac{\langle \boldsymbol{F}^l, \boldsymbol{M}_p^w\rangle}{\langle \boldsymbol{F}^l, \boldsymbol{M}_q^w\rangle}, \frac{\langle \boldsymbol{F}^l, \boldsymbol{M}_q^w\rangle}{\langle \boldsymbol{F}^l, \boldsymbol{M}_p^w\rangle}\big) > \frac{1-\mu}{1+\mu}, \tag{5.8}$$

where $\langle \cdot, \cdot \rangle$ is cosine similarity. $\boldsymbol{M}_p^w$ and $\boldsymbol{M}_q^w$ are the nearest prototypes to $\boldsymbol{F}^l$; $\boldsymbol{F}^l$ and $\boldsymbol{M}_p^w$ belong to the same class, while $\boldsymbol{M}_q^w$ doesn't. Moreover, μ refers to relative window width, defined around the midplane of the two nearest prototypes. Decision boundaries (*i.e.*, the midplane) are directly shifted toward the Bayes limits with attractive and repulsive forces from $\boldsymbol{F}^l$. Finally, prototypes can represent their respective class by assuming class-typical positions in $\mathcal{F}$.

5.2.5 Algorithm Details

Training Objective. In Interpretable3D, we utilize $|\mathcal{Y}| \times S$ prototypes $\left[\boldsymbol{m}_s^l\right]_{l,s} \in \mathbb{R}^{d \times |\mathcal{Y}| \times S}$ (*i.e.*, mean feature vectors of the training data) to involve in training objective:

$$\arg\min_{\theta} \sum_{k=1}^{K} -\log \frac{\exp\left(\max\left(\{\langle \varphi_\theta(x_k), \boldsymbol{m}_s^{l_k}\rangle\}_{s=1}^{S}\right)\right)}{\sum_{l \in \mathcal{Y}} \exp\left(\max\left(\{\langle \varphi_\theta(x_k), \boldsymbol{m}_s^{l}\rangle\}_{s=1}^{S}\right)\right)}. \tag{5.9}$$

Comparing Equation 5.2 and Equation 5.9, $\left[\boldsymbol{m}_s^l\right]_{l,s}$ are derived solely from data features, and the model can minimize training objective only by optimizing the vector $\varphi_\theta(x_k)$ instead of the parametric softmax classifier. Moreover, with such a nonparametric, distance-based scheme, Interpretable3D builds a closer link to metric learning in the adaptive latent space.

Typical Past Observations. While updating with the average of embeddings, the working mechanism remains intuitive — classify data to the class of closest sub-center, and the prototypes have a clear statistical meaning — class sub-centers. In contrast, the class weights of softmax classifier are learnable parameters that are difficult to understand. During the last 20 epochs, we find the typical past observation that has maximum similarity for each estimated prototype, and the prototypes are updated with the typical samples' features instead of average embeddings. This implements the classic *prototype theory* in cognition: human refer to past exemplar observations for classification decision-making [248–250]. Once trained, these prototypes are stored (like the learnable weights of the softmax classifier) as classification evidence. By showing these prototypical training samples, human can understand how our model actually works — it performs a direct comparison between test data and these prototypical samples for classification. Thus our model is *ad-hoc* interpretable.

Online Clustering. Intra-class data samples are grouped into S subclasses for exploiting latent structures of the entire dataset. We empirically set $S = 15$. The momentum coefficient and μ in Equation 5.8 are set as 0.999 and 0.4, following [200, 252, 253].

Backbones $\phi \circ \varphi$. We evaluate our algorithm on point-based and graph-based networks, including DGCNN [244], PointNet++ [15, 245], PointMLP [37], PointNeXt [38]. The training and testing configurations follow the default settings of the respective methods mentioned above. Our algorithm implements an interpretable prototype-based learning scheme for 3D shape classification and part segmentation. For the part segmentation task, we deploy Deep Hough Voting [234] to extract instance-level features. Therefore, Interpretable3D can be applied to any object recognition and part segmentation networks that can learn instance-wise features.

5.3 Experiment

In this section, we integrate point cloud networks with Interpretable3D to demonstrate its capabilities. We begin by presenting the results of 3D shape classification on the ModelNet40 dataset [4]. Subsequently, we delve into the performance evaluation on the ScanObjectNN dataset [5]. Moving forward, we evaluate our algorithm on a part segmentation benchmark, *i.e.*, the ShapeNetPart dataset [11]. Additionally, we utilize qualitative results to examine the decision-making process, and then use MMD2 and Maximum witness value to evaluate the prototypes and data distribution. Finally, we provide the analysis of the core components.

Datasets and Metrics. We have utilized three well-known public benchmarks, namely ModelNet40 [4] and ScanObjectNN [5] for shape classification, and ShapeNetPart [11] for part segmentation. These three datasets are widely adopted for these tasks. For shape classification, the model takes 1,024 points as input, and for part segmentation, it takes 2,048 points as input. As for the metrics, we followe the common settings of DGCNN [244], PointNet2 [245], PointMLP [37], and PointNeXt [38]. We report the class-average accuracy (mAcc) and overall accuracy (OA) for shape classification, along with the class mean intersection over union (mIoU) and instance mIoU for part segmentation.

ModelNet40. The ModelNet40 [4] benchmark contains manmade meshed CAD models of 40 categories. The data is partitioned into 9,843 training samples and 2,468 testing samples. For each

TABLE 5.1: **Classification results** on ModelNet40 [4] (Section 5.3.1).

Method	OA(%)	mAcc(%)
PointNet [14]	89.2	86.0
PointNet++ [15]	90.7	-
PointNet2 [245]	92.2	-
PointNet2 + `Ours`	93.2	89.3
DGCNN [244]	92.9	90.2
DGCNN + `Ours`	93.5	90.3
PointMLP [37]	94.1	91.3
PointMLP + `Ours`	94.1	92.0
PointNeXt [38]	94.0	91.1
PointNeXt + `Ours`	94.3	91.8

model, 1,024 points are taken randomly from the mesh faces. These points are then rescaled to fit onto a unit sphere. Similar to PointNet++ [15], the class-average accuracy (mAcc) and overall accuracy (OA) are reported on the testing set.

ScanObjectNN. We further test Interpretable3D on the ScanObjectNN benchmark [5]. This dataset, which is built upon SceneNN [256] and ScanNet [9], contains 15,000 objects from 15 different classes, and presents significant challenges. Challenges faced by existing shape classification methods stem from various factors such as occlusions, background noise, and diverse deformations. We have evaluated Interpretable3D on the hardest variant of ScanObjectNN, namely, the PB T50 RS variant [37, 38]. The class-average accuracy (mAcc) and overall accuracy (OA) are reported on the testing set.

ShapeNetPart. ShapeNetPart [11] benchmark has been developed for the 3D part segmentation task. It consists of over 16,000 models from 16 shape categories. Each category contains 2 to 6 parts, resulting in a total of 50 part labels. The dataset is divided into two subsets: 14,006 models for training and 2,874 for testing. Following the settings from [15], 2,048 points are randomly selected as inputs. We report the class mean intersection over union (class mIoU) and the instance mean intersection over union (instance mIoU) as evaluation metrics.

5.3.1 Shape Classification on ModelNet40

We compare the classification results of various classic works in Table 5.1. The results achieved by Interpretable3D are just as good as those of competitors in terms of OA and mAcc. More

TABLE 5.2: **Classification results** on ScanObjectNN [5] (Section 5.3.2).

Method	OA(%)	mAcc(%)
PointNet [14]	68.2	63.4
PointNet++ [15]	77.9	75.4
DGCNN [244]	78.1	73.6
DGCNN + **Ours**	78.0	74.3
PointNet2 [245]	79.1	77.6
PointNet2 + **Ours**	79.3	78.4
PointMLP [37]	85.4	83.9
PointMLP + **Ours**	85.6	84.5
PointNeXt [38]	87.7	85.8
PointNeXt + **Ours**	88.0	86.5

TABLE 5.3: **Segmentation results** on ShapeNetPart [11] (Section 5.3.3).

Method	Class mIoU	Instance mIoU
PointNet [14]	80.4	83.7
PointNet++ [15]	81.9	85.1
DGCNN [244]	82.3	85.2
DGCNN + **Ours**	82.9	85.5
PointNet2 [245]	82.5	85.4
PointNet2 + **Ours**	82.9	85.7
PointMLP [37]	84.6	86.1
PointMLP + **Ours**	84.9	86.2
PointNeXt [38]	85.2	87.0
PointNeXt + **Ours**	85.6	87.2

specifically, PointNet2 + **Ours** achieves 1.0% higher OA than PointNet2. PointNeXt + **Ours** and PointMLP + **Ours** demonstrate comparable performance to their counterparts on OA. These results are promising as the ModelNet40 dataset is extensively studied and the results have been long-standing at around 94%. Moreover, PointNet2 + **Ours** outperforms the advanced variant of PointNet++ [15] (91.9% OA) that incorporates normal vectors and highly dense points (5k). This verifies the effectiveness of PointNet2 + **Ours**.

5.3.2 Shape Classification on ScanObjectNN

Except for the OA metric of DGCNN + **Ours**, our approach, as detailed in Table 5.2, outperforms point-based methods by a margin of 0.2%-0.3% and 0.6%-0.8% in terms of OA and mAcc, respectively. Moreover, with Interpretable3D, all the methods narrow the gap between mAcc and OA, indicating a more decent level of robustness than their counterparts. Actually, ScanObjectNN is highly challenging due to occlusions, noise, *etc*. Through Prototype Estimation, Interpretable3D can automatically uncover real-world variations and extract distinctive prototypes. The consistent improvements of PointNet2 + `Ours`, PointMLP + `Ours`, and PointNeXt + `Ours` strongly confirm the effectiveness of Interpretable3D.

5.3.3 Part Segmentation on ShapeNetPart

We deploy Deep Hough Voting [234] to extract instance-level features. With Interpretable3D, each instance is assigned a part label, and all points belonging to the same instance are assigned the same category label. The outcomes are presented in Table 5.3. Our approach has achieved comparable results among all the tested methods, demonstrating its suitability for the part segmentation task.

5.3.4 Interpretability

As mentioned above, we have demonstrated the effectiveness of Interpretable3D. Here, we showcase its capabilities in transparency and interpretability. Following prior literature [181, 183] in other domains, we examine *ad-hoc* interpretability by presenting prototypes and assessing the similarity between prototypes and samples.

Interpretability for Predictions and Prototypes. Following prior literature [181, 183] in 2D domains, we examine ad-hoc interpretability by showing prototypes, and giving the similarity between prototypes and samples. However, none of the above methods gives quantitative metrics for evaluating model interpretability. We have exhaustively investigated past research on the prototype learning, and found that MMD-metric [257, 258] is highly suitable for evaluating the distribution of prototypes and data samples. [257] proposes a novel method called MMD-critic, which is based on the maximum mean discrepancy (MMD) criterion that measures the distance between two probability distributions. MMD-critic efficiently learns two types of features from highly complex and high-dimensional data: prototypes and criticisms. Prototypes are representative data points that capture the main characteristics and patterns of the data distribution, while criticisms are data points that highlight the differences and discrepancies between the data distribution and the prototypes. These features are designed to aid human interpretability and understanding of complex data. MMD-critic can select prototypes and criticisms that are useful for human analysis and reasoning tasks, such as classification. Therefore, MMD-critic is a reliable and effective method

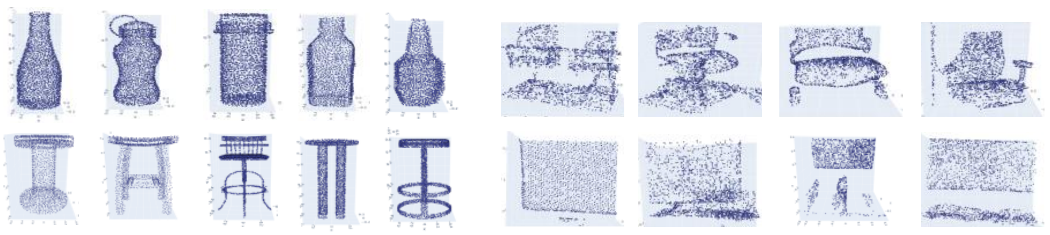


FIGURE 5.3: **Prototypes** on ModelNet40 [4] and ScanObjectNN [5] (Section 5.3.4). The right part shows the prototypes for 'bottle' and 'stool' on ModelNet40, and the left part displays the prototypes for 'chair' and 'display' on ScanObjectNN.

TABLE 5.4: **Quantitative results** on data distribution (Section 5.3.4).

Model	OA(%)↑	mAcc(%)↑	MMD2↓	Witness$_{max}$↓
ProtoPNet	92.83	89.70	0.243	0.577
PointNeXt$_{\texttt{PE}}$	94.21	91.27	0.101	0.328
PointNeXt$_{\texttt{PO}}$	94.29	91.77	0.085	0.225

for evaluating prototype and data distributions. Thus, we adopt MMD-metric, including MMD2 metric and the witness function to quantitatively measure prototype distribution and test sample distribution.

To make it easier to understand, we visualize some prototypical observations of Interpretable3D trained on the ModelNet40 dataset [4] and ScanObjectNN [5]. In Figure 5.3, the right part shows the prototypes for 'bottle' and 'stool', and the left part displays the prototypes for 'chair' and 'display'. In addition, Interpretable3D allows users to see how the model comes to its predictions by visualizing the prototypes based on the similarity scores between test sample representatives and prototypes. In Figure 5.4, we can understand how Interpretable3D makes decisions on the ModelNet40 dataset and ScanObjectNN. Taking the results on ModelNet40 for example, a bookshelf (test sample in the first row) is correctly classified, it also looks close to the prototype of 'bookshelf' (Prototype 1). However, in the failure case, Interpretable3D has difficulty in accurately determining whether the observation represents a 'flower pot' or a 'bottle'. It eventually makes the wrong decision. Even though users are unsure how Interpretable3D maps point clouds to features, the decision-making mode [176] is straightforward for users. Because it provides explanations for its decisions, humans can trust it and make decisions in high-stakes situations.

Understanding Prototypes and Data Distribution. Next, we use the MMD2 metric and the maximum witness value [257, 258] to quantitatively measure prototype distribution and test sample

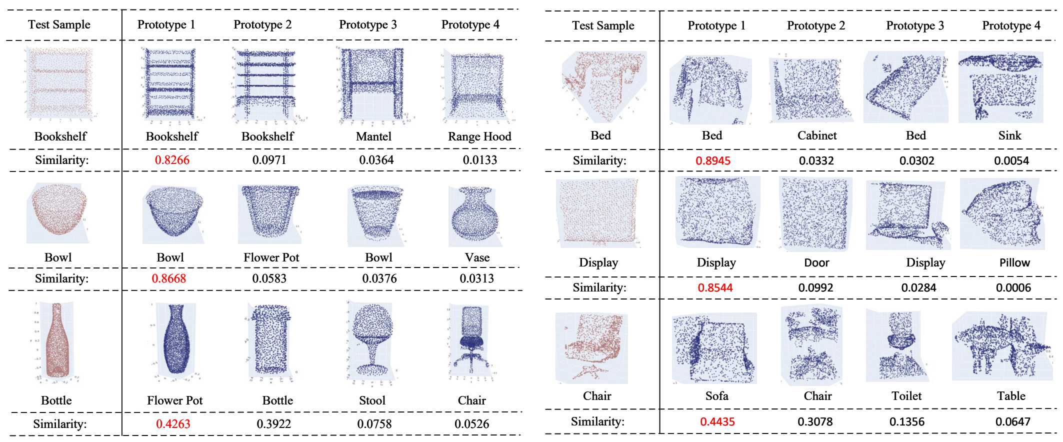


FIGURE 5.4: **The decision-making process of Interpretable3D**. The right part presents the results of Interpretable3D on ModelNet40, while the left part displays the results on ScanObjectNN. Here, we interpret predictions with normalized similarity and visualized prototypes, including success and failure cases. For each part, the first two rows show the success cases that are correctly classified, while the last row shows the failure case. Test sample is in the left column, with the four prototypes sorted by similarity scores in the right columns. Test samples are categorized based on prototypes that have maximum similarity.

distribution. We use the cosine similarity as a kernel function to approximate the densities of prototypes and test samples. This is different from the implementation in [257]. Based on the kernel function, Maximum Mean Discrepancy (MMD) is a measure of the difference between two probability distributions. When the MMD2 value is close to zero, the prototype distribution is well-suited for the dataset. This is because the prototypes are evenly distributed among the test samples within the latent space. Additionally, the witness function quantitatively represents the relationship between each test sample and all the prototypes. We treat the test sample with the maximum witness value (*i.e.*, Witness$_{max}$) as a criticism. Specifically, criticisms are data points where the distribution of prototypes and data diverges.

On the top of PointNeXt, we adapt ProtoPNet [183] to 3D vision for comparison. Following [183], ProtoPNet employs three combined models and relies on voting for results (6000 prototypes in total). Moreover, two Interpretable3D models, PointNeXt$_{\mathtt{PE}}$ and PointNeXt$_{\mathtt{PO}}$ (600 prototypes), are trained on ModelNet40. PointNeXt$_{\mathtt{PE}}$ is trained with Prototype Estimation, and PointNeXt$_{\mathtt{PO}}$ is trained with Prototype Estimation and Optimization. As shown in Table 5.4, PointNeXt$_{\mathtt{PO}}$ exhibits better performance in both MMD2 and OA compared to PointNeXt$_{\mathtt{PE}}$ and ProtoPNet. PointNeXt$_{\mathtt{PO}}$ and PointNeXt$_{\mathtt{PE}}$ surpass ProtoPNet with significantly fewer prototypes. This shows that our method learns more representative prototypes; we don't need to assemble multiple models

TABLE 5.5: **Study of Prototype Estimation/Optimization process** on ScanObjectNN [5] (Section 5.3.5). PE1, PE2, PO1, PO2 refer to Sinkhorn-Knopp algorithm, APDAGD, Equation 5.7, Equation 5.8, respectively. We report the speed by throughput (instances per second).

	PE1	PE2	PO1	PO2	PointMLP OA(%)	PointNeXt OA(%)	Train Speed ↑	Test speed ↑
Random Init					68.0±1.5	70.4±1.2	422.8	1441.0
PE1	✓				80.3±0.5	84.2±0.4	97.6	1441.0
PE2		✓			81.2±0.3	84.8±0.4	228.3	
PE1 + PO1	✓		✓		83.6±0.2	85.5±0.2	92.8	
PE1 + PO2	✓			✓	84.1±0.1	86.4±0.3	91.3	
PE2 + PO1		✓	✓		85.2±0.2	87.0±0.1	223.8	
PE2 + PO2		✓		✓	**85.6±0.1**	**88.0±0.2**	219.5	

to enlarge the representative capability. Comparing PointNeXt$_{\texttt{PO}}$ and PointNeXt$_{\texttt{PE}}$, the better performance of PointNeXt$_{\texttt{PO}}$ can be attributed to the fact that the LVQ-type algorithm optimizes the distribution of prototypes, aligning them more closely with the original distribution of the training data. In other words, it is also understandable for humans that Prototype Optimization gives a "penalty" or "reward" signal to prototypes based on object labels, making the prototypes closer to the density distribution of the training data. A test sample can be identified as a criticism if it has the largest witness value, indicating that it deviates the most from the prototype distribution. The smaller Witness$_{max}$ value of PointNeXt$_{\texttt{PO}}$ reflects that all test samples fit the prototype distribution better. The performance of PointNeXt$_{\texttt{PE}}$ and PointNeXt$_{\texttt{PO}}$ is also reported based on typical samples rather than average embeddings on ModelNet40. What's even more remarkable is that PointNeXt$_{\texttt{PO}}$ outperforms the softmax-based black-box PointNeXt (94.0% OA and 91.1% mAcc) while enhancing interpretability.

5.3.5 Ablation and Analysis

To further analyze the core components in Interpretable3D, we conduct ablation studies on the ScanObjectNN [5] test set. We set two baselines: PointMLP and PointNeXt. In Table 5.5, the mean results from three random runs are reported. All the results are obtained without voting strategies. The OA metric for the baselines (random initialization for prototypes) are 68.0% and 70.4%, respectively. Performance improvement is observed when both Prototype Estimation and Optimization are employed. However, the best results come from combining both processes, yielding 85.6% and 88.0%. This indicates that Prototype Estimation and Optimization have the potential to achieve superior performance by modifying the prototype feature space and preserving

TABLE 5.6: **Ablation studies** of PointMLP + **`Ours`**/PointNeXt + **`Ours`** on ScanObjectNN [5] `test` set.

S	PointMLP	PointNeXt
1	80.6±0.4	83.5±0.5
5	83.5±0.2	85.0±0.1
10	84.7±0.3	86.9±0.2
15	**85.6±0.1**	**88.0±0.2**
20	85.0±0.2	87.2±0.3

η	PointMLP	PointNeXt
0	81.2±0.3	84.8±0.4
$1e^{-4}$	83.9±0.3	86.5±0.2
$1e^{-5}$	84.8±0.1	87.0±0.3
$\mathbf{1e^{-6}}$	**85.6±0.1**	**88.0±0.2**
$1e^{-7}$	85.2±0.2	87.5±0.1

the most prototypical examples. Furthermore, the reported training and testing speeds confirm the high efficiency of our approach.

Per-Class Cluster Number S. The number of prototypes S for each class is shown in Table 5.6, along with its corresponding impact. Typically, there are more prototypes than categories, aligning with the findings of [246]. When each class has only one prototype ($S = 1$), we obtain 80.6% and 83.5% OA for PointMLP + **`Ours`** and PointNeXt + **`Ours`**, respectively. However, due to variations within each class, accurately representing each category with a single prototype becomes challenging. Therefore, we enhance the performance by employing more prototypes. When $S = 15$, PointMLP + **`Ours`** shows a 5.0% OA improvement, specifically from 80.6% to 85.6%. This demonstrates the effectiveness of considering intra-class variations. Beyond $S = 15$, there are diminishing returns, and eventually, the results worsen. This suggests that over-clustering may cause the model to disregard significant patterns.

Update Rate η. We next proceed to analyze the impact of the update rate η in Table 5.6. When η is set to 0, Prototype Optimization is not performed. The PointMLP + **`Ours`** model attains 81.2% OA, while PointNeXt + **`Ours`** achieves 84.8% OA. Subsequently, we optimize the prototypes with different η values. The best performance is observed when $\eta = 1e^{-6}$, yielding 85.6% and 88.0% OA, respectively.

5.4 Summary

We develop an *ad-hoc* interpretable classifier, *i.e.*, Interpretable3D, specifically designed for point cloud classification and part segmentation tasks. By performing Prototype Estimation and Optimization, Interpretable3D benefits from a reshaped prototype feature space and the preserved typical prototypes. The revealed decision-making mode enables users to understand how the system works and how decisions are made. Prototype Optimization further enhances the interpretability by illustrating how prototypes can be modified in a manner easily comprehensible for humans. Our algorithm consistently produces promising results across three datasets. In the future, we will explore the application of this *ad-hoc* style Interpretable3D to other 3D tasks.

Chapter 6

Shape2Scene: 3D Scene Representation Learning Through Pre-training on Shape Data

6.1 Introduction

Self-supervised learning (SSL), a technique for deriving representations from unannotated data, has showcased remarkable achievements across a spectrum of domains, including natural language processing [136, 259–262], computer vision [137, 200], and multi-modal learning [263–265]. The efficacy of these techniques often hinges on extensive training with sizable datasets. However, in comparison to images and text, the *data desert* issue [266] in 3D data has constrained the development of 3D SSL. Amassing extensive scene-level 3D data on a large scale demands dedicated 3D scanning equipment and platforms [2, 8, 9], resulting in financially burdensome and time-intensive data acquisition. Fortunately, shape-level datasets [4, 5] that encompass geometric information of individual objects are more readily accessible. Furthermore, due to the availability of open-source resources [267–271] and the advancement of Image-to-3D technology [272, 273], obtaining such 3D data has become easier. The number of 3D shapes available online is increasingly approaching that of 2D images in the large-scale 2D dataset. This shows potential for forthcoming large-scale shape-level datasets.

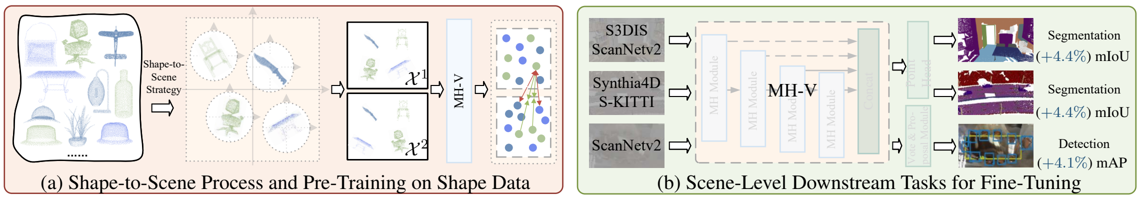

Shape-to-Scene Strategy
$\mathcal{X}^1$
$\mathcal{X}^2$
MH-V
(a) Shape-to-Scene Process and Pre-Training on Shape Data
S3DIS ScanNetv2
Synthia4D S-KITTI
ScanNetv2
MH-V
MH Module
Segmentation (+4.4%) mIoU
Segmentation (+4.4%) mIoU
Detection (+4.1%) mAP
(b) Scene-Level Downstream Tasks for Fine-Tuning

FIGURE 6.1: **Illustration** for transferring from shape data to scene-level downstream tasks, *i.e.*, Shape2Scene. The Shape-to-Scene strategy aggregates 4 (=M) shapes to one pseudo scene. Each shape is resampled and rescaled to fit onto a unit sphere. Blue scores show maximum improvements relative to training-from-scratch models. (S-KITTI stands for SemanticKITTI.)

However, existing shape-level 3D SSL methods [133–135, 139, 274] demonstrate limited potential for 3D scene understanding [13] due to significant disparities in point quantities between shape and scene data. These methods prioritize the pre-training of an encoder structure to preserve global semantic representations but overlook finer high-resolution features. The decoder structure is discarded for downstream tasks. In contrast, 3D SSL methods for 3D scene understanding [13, 118] utilize an encoder-decoder structure for high-resolution capacity. Nonetheless, these methods heavily rely on pre-training with large-scale scene-level 3D datasets, scene of which is composed of hundreds of thousands of points. Considering that 3D shape data is more readily available, the pretext task based on the shape data becomes more meaningful. The challenge lies in adapting the pre-trained model in the aforementioned setting to scene-level downstream tasks.

To tackle these challenges, we have introduced a novel 3D scene representation learning method called Shape2Scene (S2S). This method bridges the gap between shape-level and scene-level datasets from three aspects: **architecture**, **data**, and **loss**. For 3D scene understanding, including semantic segmentation and object detection [127, 223, 237, 275], high resolution capacity is an important influencing factor. However, existing methods either neglect the learning of high-resolution features or overlook the features of multi-scale structures. To tackle this, multi-scale and high-resolution **architecture** is proposed. It is principled enough to be incorporated into both point-based and voxel-based backbones, namely, MH-P and MH-V. Specifically, they map multi-scale deep semantic information to high-resolution representations, ensuring the model's transferability and generalizability for 3D scene understanding. MH-P/V incorporates a series of Multi-scale High-resolution (MH) Modules. By stacking them, MH-P/V generates high-resolution semantic representations across multiple scales, providing an advantage for high-resolution downstream tasks. In terms of **data**, we adopt a straightforward approach called the Shape-to-Scene strategy

(S2SS), which involves combining points from various shapes to generate a pseudo scene with multiple objects. This strategy reduces disparities in point quantities, simulating the placement of multiple instances within a scene, and alleviates the gap between shape and scene. Regarding the **loss** function, we introduce a point-point contrastive loss (PPC) for pre-training, ensuring that MH-P/V model learns to capture intricate details and features within the scene (see Figure 6.1). The inherent point pairs are naturally obtained in S2SS, making them suitable for PPC without the need for time-consuming point-level prepairing [13, 276].

To the best of our knowledge, S2S represents a novel method for learning 3D scene representations. Initially pre-trained on 3D shape data, it is highly adaptable to a range of downstream tasks, including large-scale 3D scene understanding. MH-P/V is a high-resolution architecture specifically designed for this method, offering three key advantages. Firstly, it provides a direct pathway to high-resolution deep semantic information across diverse scales. This fundamental characteristic makes it well-suited for various 3D downstream tasks. Secondly, it seamlessly integrates features from different scales into the point head, maintaining abstraction at varying levels and ensuring consistent semantic retention throughout the network. Features from diverse scales collaborate to collectively enhance the model's performance. Thirdly, it is versatile, as its architecture accommodates both point-based and voxel-based backbones across various 3D downstream tasks.

For a comprehensive evaluation, we examined the transferability of MH-P/V across multiple 3D tasks, yielding more promising results compared to previous 3D SSL methods. MH-P achieves an accuracy of 94.6% on ModelNet40 [4] and 93.8% on ScanObjectNN [5]. Additionally, for part segmentation, MH-P achieves an instance mIoU of 87.6% on ShapeNetPart [11]. In terms of indoor semantic segmentation, MH-V achieves a mIoU of 74.1% on S3DIS [2] and 75.8% on ScanNet v2 [9]. For outdoor semantic segmentation, MH-V achieves a mIoU of 71.5% on SemanticKITTI [1] and 84.2% on Synthia4D [12]. Furthermore, MH-V demonstrates a 43.9% mAP@0.5 in 3D object detection.

6.2 Methodology

This section introduces the novel 3D scene representation learning method, S2S, learning representations of large-scale 3D scenes from 3D shape data. In Section 6.2.1 and Section 6.2.2, we

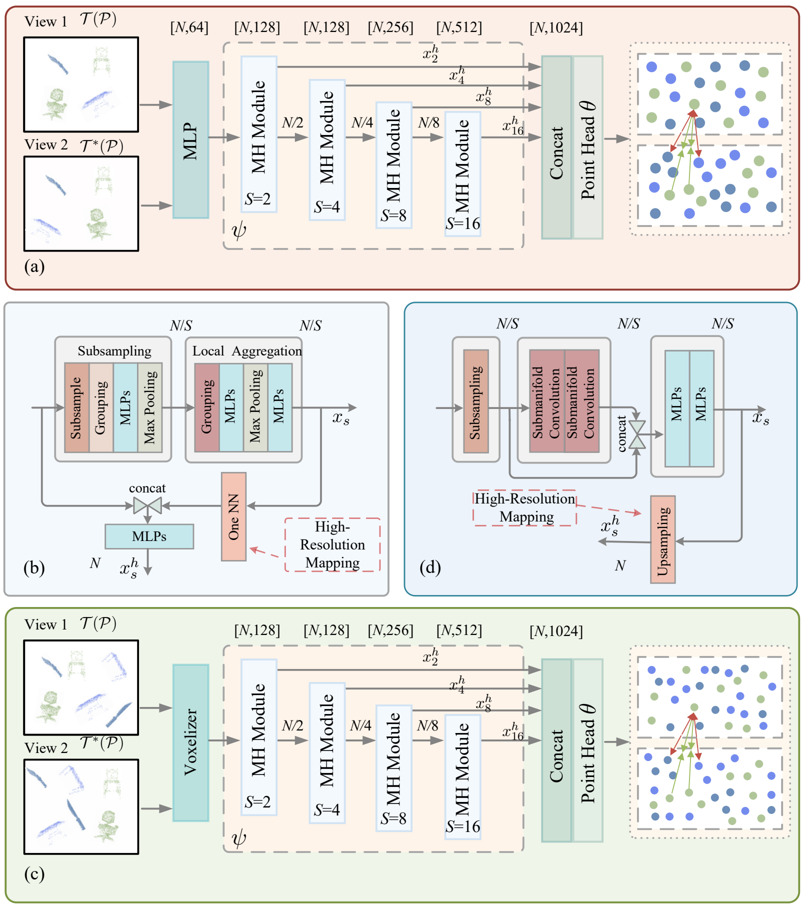


FIGURE 6.2: **Overview of our shape2scene**. (a) The overview of the MH-P backbone during pre-training, with contrastive loss (Equation 6.1). (b) The Multi-scale High-resolution (MH) Module of MH-P (see Section 6.2.1 for details.) (c) The overview of the MH-V backbone during pre-training, with contrastive loss (Equation 6.1). (d) The Multi-scale High-resolution (MH) Module of MH-V (see Section 6.2.2 for details).

commence with the introduction of MH-P/V and MH module. MH-P/V is a versatile point cloud pre-training backbone, providing a flexible foundation for SSL of 3D point clouds. It is applicable to a range of 3D downstream tasks, spanning from shape-level tasks to scene-level tasks, especially for tasks that focus on learning high-resolution features. Subsequently, we delve into the specifics of S2SS in Section 6.2.3, along with a detailed description of PPC in Section 6.2.4.

Suppose that the input is $\mathcal{I}=\{\mathcal{P}^k\}_k^K$, *i.e.*, a set with K training samples. Here $\mathcal{P}^k=\{p_n^k\in\mathbb{R}^{3+a}\}_n^N$ is the k-*th* sample containing N points with 3D position and other auxiliary information (*e.g.*, color). We represent the MH-P/V backbone and point head as ψ and θ, respectively.

6.2.1 Point-based MH Backbone

Figure 6.2 (a) provides an overview of MH-P designed for the shape-level tasks. MH-P comprises MH modules. High-resolution features are learned by point head θ.

MH Module of MH-P. As shown in Figure 6.2 (b), each MH module is divided into three parts: subsampling, local aggregation, and high-resolution mapping. In each MH module, we adopt a set abstraction (SA) block [15] for subsampling. One SA block comprises several key components: a subsampling layer that reduces the resolution of incoming points, a grouping layer responsible for identifying neighbors for each point, a series of shared multi-layer perceptrons (MLPs) designed for feature extraction, and a max pooling layer that combines features from neighboring points. Local aggregation consists of a grouping layer, MLPs, and a max pooling layer. High-resolution mapping employs nearest neighbor interpolation to assign low-resolution high-dimensional features to the nearest high-resolution points.

MH-P. Assuming one MH module with the scale of S, both spatial geometry and semantic information are learned through subsampling and local aggregation. We denote the learned deep semantic representation as the semantic feature x_s for S. After undergoing high-resolution mapping, x_s is ultimately transformed into a high-resolution format, yielding a high-resolution semantic representation x_s^h. It will be used as the input for the next scale. As the scale increases, it progressively gathers more sophisticated semantic information from lower-resolution data. Stacked modules generate point-level semantic representations at multiple scales, which are advantageous for high-resolution tasks. The overall network structure is shown in Figure 6.2 (a).

For shape-level high-resolution tasks, *e.g.*, part segmentation, $\{x_s^h\}_s$ are integrated as the input of θ. This ensures the consistent retention of semantics throughout the network, and the collective synergy of features from various scales enhances the model's performance. MH-P establishes a direct pathway to high-resolution features that capture rich semantic information across multiple scales. By translating multi-scale deep semantic information into high-resolution/point-level representations, MH-P ensures that features originally trained on 3D shape data can seamlessly adapt to a variety of 3D downstream tasks, especially for tasks that necessitate high-resolution representations. This pivotal characteristic of MH-P makes it exceptionally well-suited for high-resolution representations in part segmentation.

The shape classification task does not emphasize high-resolution representations but instead extracts global features. To adapt to this requirement, we only need to make minor modifications to MH-P. In each MH module, x_s passes through one more max-pooling layer, obtaining x_s^g, which represents the global feature of scale S. Subsequently, we seamlessly integrate x_s^g from different scales into the input of one shape head, preserving abstractions at various levels. The surprising thing is that, during the transition from high-resolution representations to global features, MH-P still achieves better performance on downstream tasks (see Section 6.3.2). We attribute this to the network's capacity to acquire essential abstractions at various scales. This guarantees the preservation of vital information and its effective utilization across different scales, thereby maintaining performance even with the shifts towards global features.

6.2.2 Voxel-based MH Backbone

There are significant differences in dataset statistics between shape-level and scene-level data, with $\sim$1k input points for ModelNet40 [4], $\sim$40k input points for ScanNet v2 [9], and $\sim$956K input points for S3DIS [2]. Following the methods for processing large-scale point clouds [6, 18], we design a voxel-based MH backbone with MH modules. Figure 6.2 (c) provides an overview of MH-V, and Figure 6.2 (d) shows the MH module for MH-V. Different from hourglass-like networks, MH-V follows a similar concept to MH-P and learns high-resolution features at the scene level across multiple scales. For simplicity, we retain the symbols used in Section 6.2.1.

MH Module of MH-V. As depicted in Figure 6.2 (d), this MH module possesses a structure similar to that of the MH module of MH-P (in Section 6.2.1), including featuring subsampling, local aggregator, and upsampling functionalities. Specifically, the voxelizer first defines indices and inverse indices between points and voxels at different scales, as well as indices and inverse indices between voxels at different scales. These indices will be applied for subsampling, *i.e.*, point features with the same voxel index will be averaged and then treated as features of the corresponding voxel. The inverse indices will be used for upsampling or high-resolution mapping, *i.e.*, voxel features will be assigned to corresponding points according to the inverse indices. The local aggregator employs stacked submanifold convolutions to extract local features, combines them with features from skip connections, and then processes them through a series of MLPs to derive a deep semantic

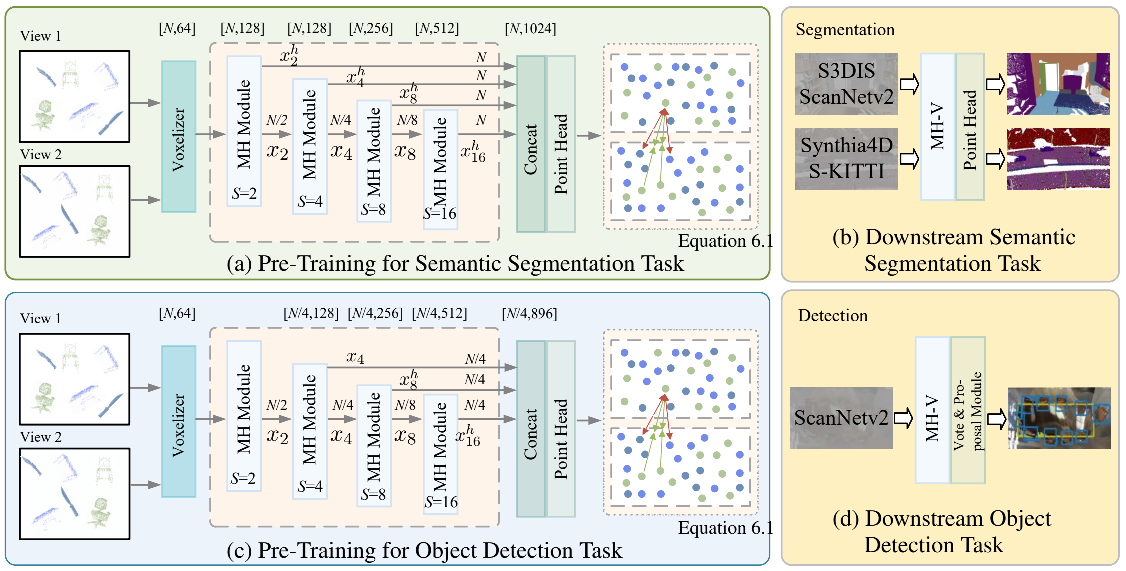

View 1
View 2
Voxelizer
MH Module
[N,64] [N,128] [N,128] [N,256] [N,512] [N,1024]
[N,64] [N/4,128] [N/4,256] [N/4,512] [N/4,896]
S=2
S=4
S=8
S=16
Concat
Point Head
Equation 6.1
(a) Pre-Training for Semantic Segmentation Task
Segmentation
S3DIS ScanNetv2
Synthia4D S-KITTI
MH-V
(b) Downstream Semantic Segmentation Task
(c) Pre-Training for Object Detection Task
Detection
ScanNetv2
Vote & Proposal Module
(d) Downstream Object Detection Task

FIGURE 6.3: **Illustration for transferring from shape data to scene-level downstream tasks**, *i.e.*, Shape2Scene (Section 6.2.2). (a) shows the pre-training of MH-V backbone for semantic segmentation task; (b) shows the downstream semantic segmentation tasks; (c) shows the pre-training of MH-V backbone for object detection task; and (d) shows the downstream object detection tasks. (Best viewed with zoom-in.)

representation x_s. While upsampling, x_s is ultimately transformed into a high-resolution/point-level representation x_s^h.

MH-V. In MH-V, x_s of the MH module for S is taken as input of the MH module for the next scale. Similar to MH-P, we employ high-resolution mappings within each MH module, and all $\{x_s^h\}_s$ are integrated as the input of θ. Our MH-V not only effectively manages the substantial volume of data at the scene level but also benefits from direct access to high-resolution data and integrated features from various scales.

Our MH-V differs fundamentally in structure from other hourglass-like voxel-based 3D networks (*e.g.*, SparseConvNet [82] and MinkUNet [18]). MH-V serves as a backbone network aimed at pursuing multi-scale high-resolution features. The significance of high-resolution 3D network has been validated in [87, 277], thus affirming the effectiveness of our MH-V. However, unlike previous methods, we achieve downsampling and upsampling entirely through indices and inverse indices, eliminating the need for **regular** sparse convolution or trilinear interpolation of nearest neighbor voxels. Our approach is more streamlined, efficient, and not affected by the issue of blurring valuable information caused by regular sparse convolution [34, 88].

MH-V for 3D Semantic Segmentation. Figure 6.3 (a) visually demonstrates the pre-training process of MH-V backbone for the semantic segmentation task. It is constructed atop four MH modules (as referenced in Figure 6.2 (d)). These modules are configured to operate at scales of 2, 4, 8, and 16, respectively. Specifically, within the MH module designated for scale S, its input derives from the output $x_{s'}$ of the former MH module (with a scale of S'). Additionally, x_s transitions into the input for the subsequent MH module. Similar to MH-P backbone, high-resolution mapping is implemented within each MH module, seamlessly integrating all x_s^h as the input for the point head. This innovative architecture empowers MH-V with direct access to high-resolution data and integrated features spanning various scales. Subsequent to the pre-training phase, we apply MH-V backbone to the downstream semantic segmentation task, as illustrated in Figure 6.3 (b).

MH-V for 3D Object Detection Figure 6.3 (c-d) show the MH-V backbone for pre-training and the downstream object detection task, respectively. The backbone is built on the top of four MH modules (see Figure 6.2 (d)). The modules are operated at scales of 2, 4, 8, and 16, respectively. The MH module used here shares the same structure as the one employed in the semantic segmentation task. The difference lies in that the integration of x_4, x_8^h, and x_{16}^h is taken as the input of point head. The high-resolution features, at a scale of 4, are mapped from features on scales of 8 and 16. This is because the 3D object detection task focuses on region-level features. Moreover, 4,096 points at a scale of 4 are randomly selected and then applied to generate point pairs for Equation 6.1. After the pre-training phase, we apply MH-V as the backbone and use the vote & proposal module [234] to detect objects, as illustrated in Figure 6.3 (d).

6.2.3 Shape-to-Scene Strategy

Previous work [13] has demonstrated the significance of point-level representations for 3D scene understanding. Directly training on a single shape and acquiring a global representation might be inadequate for scene-level tasks. To mitigate this concern, it could be advantageous to directly pre-train the network on complex scenes containing multiple objects to more accurately align with the target distributions [13]. Therefore, a series of SSL methods for 3D scene understanding have depended on scene data [13, 118, 148, 149]. However, collecting 3D scene data is financially burdensome and time-intensive. With the development of open-source platforms and Image-to-3D technology, 3D shape data has become more readily accessible. Here, we revisit the challenge of extending 3D SSL from 3D shape data to 3D scene data.

We propose S2SS to aggregate multiple objects (shapes) to generate pseudo scene-level training data. This strategy is entirely different from previous methods [13, 118, 148, 149] in terms of how the pre-training data is created. Moreover, it brings the additional benefit that it even allows the natural derivation of positive and negative pairs by leveraging known shape correspondences, without the need for any extra computation. As shown in Figure 6.1 (a), given M shapes, we undergo random presampling with 2,048 points for each shape. Each shape is then normalized by rescaling it to fit onto a unit sphere. Through translation, we position various shapes within a shared world coordinate. The Euclidean distance between the barycenters of any two shapes is ensured to be greater than 2, preventing any overlap among shapes. The final output yields pseudo scene data containing M shapes: $\mathcal{P}_s = \{p_n\}_{n=1}^{M\times 2048}$. Next, we use two rigid transformations, denoted as $\{\mathcal{T}(\cdot), \mathcal{T}^*(\cdot)\}$, randomly sampled from the transformation set $\mathcal{T}$. The transformations in $\mathcal{T}$ are applied to the pseudo scene data, and $\mathcal{T}$ encompasses rotation, translation, and scaling, *etc*. We produce two views $\mathcal{X}^1{=}\mathcal{T}(\mathcal{P}_s)$ and $\mathcal{X}^2{=}\mathcal{T}^*(\mathcal{P}_s)$, which are aligned within the same world coordinates.

6.2.4 Point-Point Contrastive Loss

The fundamental principle of contrastive learning hinges on the concept of *invariance learning* [200, 278, 279]. This entails that the abstraction of semantics generally remains either invariant or equivariant [280] in the face of various transformed perspectives, such as augmentations [116]. Previous work [13, 281] has demonstrated the significance of point-level representations over global representations. Similarly, our designed MH-P/V aims to acquire point-level/high-resolution features. Therefore, on top of the PointInfoNCE loss [13], we introduce PPC. However, PPC shares a close relationship with S2SS, where the inherent point pairs are naturally obtained without relying on the time-consuming point-level pairing method, FCGF [13, 276]. During pre-training, we utilize θ to acquire high-resolution 3D representations. Given input data p, the model $\mathcal{E}$ is optimized by:

$$\mathcal{L}_{\text{PPC}} = -\sum_{(u,v)\in\mathcal{O}_p} \log \frac{\exp\left(\mathcal{E}(\mathcal{T}(p_u))\cdot\mathcal{E}(\mathcal{T}^*(p_v))/\tau\right)}{\sum_{(\cdot,w)\in\mathcal{O}_p}\exp\left(\mathcal{E}(\mathcal{T}(p_u))\cdot\mathcal{E}(\mathcal{T}^*(p_w))/\tau\right)}, \tag{6.1}$$

where $\mathcal{E}=\theta \circ \psi$ represents the whole model, $\mathcal{E}\big(\mathcal{T}(p_*)\big)$ denotes the representation learned by $\mathcal{E}$; τ is a temperature hyperparameter; and $\mathcal{O}_p=\{(n,n)\,|\,p_n\in\mathcal{P}_s\}\cup\{\,(i,j)\,|\,p_i^k\in\mathcal{P}^k\cap p_j^k\in\mathcal{P}^k\}$ denotes the collection of all positive matches derived from two different views. This approach focuses solely on points that have at least one corresponding match and regards additional non-matched points as negative matches. In a matched pair $(u,v)\in\mathcal{O}_p$, the point feature $\mathcal{E}\big(\mathcal{T}(p_u)\big)$ operates as the query, and $\mathcal{E}\big(\mathcal{T}(p_v)\big)$ functions as the positive key. We employ the point feature $\mathcal{E}\big(\mathcal{T}^*(p_w)\big)$, where $\exists(\cdot,w)\in\mathcal{O}_p$ and $w\neq v$, as the pool of negative keys. Following [13], we choose a subset of 4,096 matched pairs from $\mathcal{O}_p$ for scene-level downstream tasks. However, we empirically select a subset of 2,048 matched pairs from $\mathcal{O}_p$ for shape-level downstream tasks. Additionally, we present a PyTorch-style pseudo-code for PPC in Algorithm 6.1.

Algorithm 6.1 Pseudo-code of Point-Point Contrastive Loss.

```
Require: $Z_1$, $Z_2$: features for matched points between view 1 and view 2: ($M$ × 2048, $C$);
         $t$: temperature; $N_s$: subsampling size for point features.
Ensure: loss: Point-Point Contrastive Loss
  mark = torch.arange($M$).repeat_interleave(2048)                ▷ mark each point
  mark = mark.view(-1,1)
  pairs = torch.eq(mark,mark.T)
  pos_pair = torch.where(pairs==True)                             ▷ get positive pairs
  pos_pair_num = pos_pair[0].shape[0]                             ▷ get subsampling indexes
  inds = random.sample(range(pos_pair_num), $N_s$)
  $Z_1$ = $Z_1$[pos_pair[0][inds],:]                              ▷ get subsample point features
  $Z_2$ = $Z_2$[pos_pair[1][inds],:]
  sim = torch.mm($Z_1$,$Z_2$.T)                                   ▷ $N_s \times N_s$
  labels = torch.arange($N_s$)
  loss = CrossEntropyLoss(sim/$t$,labels)
```

The way we obtain $\mathcal{O}_p$ fundamentally differs from previous methods [13, 118]. On the one hand, rigid transformations maintain the point cloud's order, ensuring a one-to-one correspondence (*i.e.*, $\{(n,n)\,|\,p_n\in\mathcal{P}_s\}$) between $\mathcal{X}^1$ and $\mathcal{X}^2$. On the other hand, points originating from the same shape are regarded as positive pairs (*i.e.*, $\{\,(i,j)\,|\,p_i^k\in\mathcal{P}^k\cap p_j^k\in\mathcal{P}^k\}$) across the two views. In this way, we create the scene-level point pair data that is original from single objects. These designs fully exploit the correspondences between points and the relationships within points in the shape.

6.3 Experiment

In Section 6.3.1, we introduce the self-supervised pre-training settings for MH-P/V. Subsequently, we present the supervised fine-tuning performance on shape-level and scene-level downstream tasks in Section 6.3.2 and Section 6.3.3, respectively. Furthermore, we conduct ablation studies in Section 6.3.4 to validate the effectiveness of each part of our method.

6.3.1 Pre-Training Settings

MH-P and Data Settings. We conduct pre-training of MH-PS on ShapeNet dataset [282] and MH-PH on both the labeled hybrid dataset (LHD) and the unlabeled hybrid dataset (UHD) [139]. MH-PS is a backbone network composed of four MH modules. ShapeNet comprises $\sim$52K synthetic 3D shapes in 55 categories. MH-PS is trained without any post-pre-training, aligning with previous SSL methods [133–135, 274] to enable direct comparison with them. Moreover, MH-PH is pre-trained on LHD and UHD. MH-PH also consists of four MH modules. However, to explore a high capacity model, we scale it by increasing the network width and the number of local aggregation within the MH modules. The UHD, used for self-supervised pre-training, aggregates point clouds from various sources such as ShapeNet [282], S3DIS [2] for indoor scenes, and Semantic3D [283] for outdoor scenes, *etc*. In total, UHD comprises around 300K point clouds. Conversely, LHD, utilized for post-pre-training, aligns the label semantics from diverse datasets, including ShapeNet [282], S3DIS [2], and other sources, encompassing 87 categories and roughly 200K point clouds in total.

MH-V and Data Settings. We employ a similar setup as described above. The difference lies in combining M shapes to form a scene for pre-training. We denote the MH-V pre-trained on ShapeNet as MH-VS, and the MH-V pre-trained on UHD and LHD as MH-VH. We also scale MH-VH by increasing the network width and the number of local aggregators within the MH modules.

Pre-Training Setups. We extract the input shapes by sampling 2,048 points from each raw point cloud, and compose M shapes to form a scene. See Section 6.2.3 and Section 6.3.4. The MH-P model undergoes pre-training for 600 epochs with a batch size of 10. We utilize the AdamW optimizer [284] with an initial learning rate of 0.001, and a weight decay of 0.05. Additionally, we

TABLE 6.1: **Classification results** on ScanObjectNN [5] and ModelNet40 [4] (Section 6.3.2). We report the overall accuracy.

Methods	ScanObjectNN			ModelNet40
	OBJ_BG	OBJ_ONLY	PB_T50_RS	
Supervised Learning Only				
DGCNN [244]	82.8	86.2	78.1	92.9
PointNet++ [15]	-	-	77.9	90.5
PointMLP [37]	-	-	85.4	94.5
PointNeXt [38]	-	-	87.7	94.0
Pre-training on ShapeNet				
Point-BERT [133]	87.4	88.1	83.1	93.2
MaskPoint [274]	89.3	88.1	84.3	93.8
Point-MAE [134]	90.0	88.3	85.2	93.8
Point-M2AE [135]	91.2	88.8	86.4	94.0
PointGPT-S [139]	91.6	90.0	86.9	94.0
MH-PS (Ours)	**92.4**	**90.8**	**87.8**	**94.1**
Pre-training on UHD and LHD				
PointGPT-B [139]	95.8	95.2	91.9	94.4
PointGPT-L [139]	97.2	96.6	93.4	**94.7**
MH-PH (Ours)	**97.4**	**96.8**	**93.8**	94.6

TABLE 6.2: **Segmentation results** on the ShapeNet-Part [11] dataset (Section 6.3.2). We report the Class mIoU and Instance mIoU.

Methods	Backbone	ShapeNetPart	
		Cls. mIoU(%)	Inst. mIoU(%)
Supervised Learning Only			
PointNet++ [15]	PointNet++	81.9	85.1
DGCNN [244]	DGCNN	82.3	85.2
Pre-training on ScanNet			
DGCNN [117]	DGCNN	-	85.0
PointNet++ [148]	PointNet++	-	85.9
PointViT [148]	Transformer	-	86.7
Pre-training on ShapeNet			
Point-BERT [133]	Transformer	84.1	85.6
Point-MAE [134]	Transformer	-	86.1
Point-M2AE [135]	Transformer	84.9	86.5
PointGPT-S [139]	Transformer	84.1	86.2
MH-PS (Ours)	S2S	**85.3**	**87.1**
Pre-training on UHD and LHD			
PointGPT-B [139]	Transformer	84.5	86.5
PointGPT-L [139]	Transformer	84.8	86.6
MH-PH (Ours)	S2S	**85.5**	**87.6**

employ cosine learning rate decay [285] based on our empirical findings. The MH-V model undergoes pre-training for 800 epochs on two V100 GPUs with a batch size of 10. Other configurations align with those used for MH-P.

6.3.2 Shape Level Downstream Tasks

We perform supervised fine-tuning of MH-PS and MH-PH on three classical downstream tasks. Each shape-level downstream task has thousands of input points.

Shape Classification on ScanObjectNN. ScanObjectNN stands out as one of the most formidable 3D datasets, encompassing 15K objects extracted from real-world indoor scans [5]. Within the dataset, there are three commonly utilized data splits, namely OBJ_ONLY (object only), OBJ_BG (with background), and the PB_T50_RS (with background and manual perturbations). Following [5, 133–135, 139, 274], we conduct experiments on the three splits mentioned above and reported overall accuracy (OA) on the respective test sets. As evidenced in Table 6.1, our approach, MH-PS, outperforms PointGPT-S on all three data splits' test sets. Our performance improvements can be attributed to the effective utilization of the multi-scale mechanism. It contributes to the preservation of global features and results in enhanced performance at the shape level tasks,

TABLE 6.3: **Segmentation results** on S3DIS Area5 [2] and ScanNet v2 validation set [9] (Section 6.3.3). We report mIoU (%).

Methods	Backbone	S3DIS Area5	ScanNet v2
Supervised Learning Only			
PointNet++ [15]	PointNet++	55.3	57.9
MinkNet [18]	SR-UNet	68.2	70.3
MH-V (Ours)	MH-V	70.3	71.4
Pre-training on ScanNet			
PointContrast [13]	SR-UNet	70.9	74.1
DepthContrast [118]	SR-UNet	71.5	71.2
OcCo [117]	DGCNN	58.0	-
PointClustering [148]	PointNet++	61.2	62.6
PointClustering [148]	Transformer	65.6	65.8
PointClustering [148]	SR-UNet	73.2	75.5
Pre-training on ShapeNet / UHD and LHD			
MH-VS (Ours)	MH-V	72.7	74.4
MH-VH (Ours)	MH-V	**74.1**	**75.8**

TABLE 6.4: **Segmentation results** on SemanticKITTI val set [1] and Synthia4D test set [12] (Section 6.3.3). We report mIoU (%) here.

Methods	Backbone	SemanticKITTI	Synthia4D
Supervised Learning Only			
PointNet++ [15]	PointNet++	-	79.4
MinkNet [18, 87]	SR-UNet	61.1	79.8
SPVNAS [87]	SPVCNN	64.7	-
MH-V (Ours)	MH-V	67.1	81.4
Pre-training on ScanNet			
PointContrast [13]	SR-UNet	-	83.1
DepthContrast [118]	SR-UNet	-	81.3
Pre-training on ShapeNet			
PointDif [287]	SR-UNet	71.3	-
MH-VS (Ours)	MH-V	70.8	82.4
Pre-training on UHD and LHD			
MH-VH (Ours)	MH-V	**71.5**	**84.2**

yielding an improvement of approximately 0.8% to 0.9%. Furthermore, MH-PH showcases better performance compared to PointGPT-L.

Shape Classification on ModelNet40. ModelNet40 is a classical dataset for synthetic 3D object recognition [4]. It contains $\sim$12K meshed 3D CAD objects of 40 classes. For fair comparisons, the standard voting method [286] is used during testing. The experimental results are presented in Table 6.1, MH-P achieves comparable performance to PointGPT in two pre-training data settings. ModelNet40 has been thoroughly explored (saturated around 94%), and this performance is already convincing enough. Moreover, MH-PS surpasses other MAE-style and BERT-style pre-training frameworks.

Part Segmentation. We evaluate MH-P on ShapeNetPart [11] for part segmentation task (a high-resolution task), which predicts per-point part labels for a known object. The ShapeNetPart dataset consists of 17K objects across 16 categories. The point clouds are sampled into 2,048 points. Table 6.2 provides the instance mean Intersection over Union (Inst. mIoU) and class mean Intersection over Union (Cls. mIoU) results. Despite the strong Transformer architectures (MAE-style and BERT-style backbones), our MH-PS even surpasses the performance of PointGPT-B and PointGPT-L, which are pre-trained on UHD and LHD. This underscores the effectiveness of our method, as MH-P captures high-resolution semantic features from deep semantic information across multiple scales. Additionally, our MH-PH achieves state-of-the-art performance on both Cls. mIoU and Inst. mIoU metrics.

6.3.3 Scene Level Downstream Tasks

We perform fine-tuning for MH-VS and MH-VH on five scene-level downstream tasks. Each scene-level downstream task has hundreds of thousands of input points.

Semantic Segmentation on Indoor Scene. We proceed with the evaluation of MH-V in the indoor semantic segmentation task, involving the classification of points within 3D scenes into distinct categories. This research involves pre-training on pseudo scenes and finetuning on the Stanford Large-Scale 3D Indoor Spaces (S3DIS) dataset [2] and ScanNet v2 validation set [9], respectively. They are based on the standard settings [13, 18, 148]. S3DIS encompasses 3D scans of six expansive indoor areas. Following the prior method [18], Area 5 is designated as the test set. There are approximately 204 samples in the training set. Each training sample contains an average of $\sim$956K points. ScanNet v2 comprises richly annotated 3D indoor scenes, encompassing 1.5K scenes from hundreds of distinct rooms. There are 1,201 scenes for training. The mean Intersection over Union (mIoU) of various approaches is summarized in Table 6.3. Even with S2SS, transferring features from the pseudo scene to S3DIS and ScanNet v2 presents a considerable challenge. While UHD includes specific objects outlined in S3DIS, discrepancies arise in terms of occlusion, object scale, and scene magnitude. However, when compared to pre-training methods on ScanNet, such as OcCo [117] (with DGCNN as the backbone), and PointClustering [148] (with PointNet++, PointViT, and SR-UNet as backbones), MH-V shows significant improvement. It also outperforms pre-training methods [13, 118] that use SR-UNet as the backbone. These results conclusively affirm that MH-V derives substantial advantages from the acquisition of high-resolution features across various scales. The high-resolution representation effectively harnesses the semantics within point cloud data, thereby bolstering the network's proficiency in semantic segmentation.

Semantic Segmentation on Outdoor Scene. We also conduct experiments on SemanticKITTI [1] and Synthia4D dataset [12] for the outdoor semantic segmentation task. SemanticKITTI is a large-scale driving-scene dataset, containing 43K scans with point-wise annotation. We use sequences 00 to 10 for training, and 08 is left for validation. Synthia4D is a large synthetic dataset depicting driving scenarios. We adhere to the train/validation/test split as defined by [18]. Our implementation does not involve temporal learning. mIoU of different methods is summarized in Table 6.4. Fine-tuning on them poses similar challenges as previously mentioned. However, we significantly mitigate the differences between shape data and scene data by employing synthesized pseudo

TABLE 6.5: **3D object detection** on ScanNet v2 [9] (Section 6.3.3). We report mAP (%). SR-UNet stands for Sparse Residual U-Net [13].

Methods	Backbone	mAP@0.25	mAP@0.5
Supervised Learning Only			
VoteNet [234]	PointNet++	58.6	33.5
3DETR [289]	Transformer	62.1	37.9
MH-V (Ours)	MH-V	62.9	39.8
Pre-training on ShapeNet			
TAP [290]	Transformer	63.0	41.4
PointDif [287]	Transformer	-	43.7
MH-VS (Ours)	MH-V	63.4	42.1

Methods	Backbone	mAP@0.25	mAP@0.5
Pre-training on ScanNet			
PointContrast [13]	SR-UNet	59.2	38.0
STRL [288]	PointNet++	59.5	38.4
Point-BERT [133]	Transformer	61.0	38.3
DepthContrast [118]	PointNet++	64.0	42.9
MaskPoint [274]	Transformer	64.2	42.1
Pre-training on UHD and LHD			
MH-VH (Ours)	MH-V	**64.8**	**43.9**

scenes. In comparison to the methods pre-trained on the ScanNet and ShapeNet datasets, our approach has also achieved encouraging results. This indicates that the representations learned in pseudo scenes still enhance the generalization for segmentation in both real-world and synthetic outdoor scenes.

3D Object Detection on ScanNet v2. To further evaluate MH-V on 3D object detection, we utilize MH-V as the backbone on the ScanNet v2 dataset [9]. Our methodology adheres to the training/validation split and 18 classes outlined in VoteNet [234]. We also follow VoteNet [234] and switch the original backbone network with MH-V without other modifications to the detection module. As reported in Table 6.5, MH-VH outperforms MH-V by 4.1% mAP@0.5. Previous methods [13, 118, 133, 274, 288] conduct pre-training on large-scale scene datasets like ScanNet. In contrast, we conduct pre-training on pseudo scenes originally from individual objects. The experiments also show the effectiveness of MH-V over SR-UNet, as MH-V learns multi-scale point cloud encoding by mapping scale features to high-resolution features. This demonstrates that the representations learned from multiple objects can be successfully transferred to large-scale datasets and enhance the performance of scene-level tasks.

6.3.4 Ablation Study

Comparsion to Training-from-scratch Baseline. We first compare the training-from-scratch baseline and the fine-tuned network on ScanObjectNN (PB_T50_RS), ShapeNetPart, and S3DIS Area5, respectively. Specifically, MH-PS and MH-VS are pre-trained on ShapeNet and fine-tuned on the target dataset. The upper part of Table 6.6 details the model configurations. **`Model A`**, **`B`**, **`D`**, and **`E`** are employed as training-from-scratch baselines. The backbone of **`Model A`** is PointNet++ with an encoder on ScanObjectNN, while it uses PointNet++ with both encoder and decoder on

TABLE 6.6: **Ablations for different designs** (Section 6.3.4). We report Overall Accuracy (%) on ScanObjectNN (PB_T50_RS), Instance mIoU (%) on ShapeNetPart, and mIoU (%) on S3DIS Area5.

Models	Pre-training	Backbone	Data Type
`Model A`	Scratch	PointNet++	Shape
`Model B`	Scratch	MH-P	Shape
`Model C`	✓	MH-P	Shape
`Model D`	Scratch	SR-UNet	Scene
`Model E`	Scratch	MH-V	Scene
`Model F`	✓	MH-V	Scene

Models	ScanObjectNN	ShapeNetPart	S3DIS Area5
`Model A`	77.9	85.1	-
`Model B`	85.9	86.0	-
`Model C`	87.8	87.1	-
`Model D`	-	-	68.2
`Model E`	-	-	70.3
`Model F`	-	-	72.7

TABLE 6.7: **Ablations for network design** (Section 6.3.4). We report Overall Accuracy (%) on ScanObjectNN (PB_T50_RS), Instance mIoU (%) on ShapeNetPart, and mIoU (%) on S3DIS Area5.

Models	Backbone	S_1	S_2	S_3	S_4
`Model G`	MH-P	✓/✓	✓/✓		
`Model H`	MH-P	✓/✓	✓/✓	✓/✓	
`Model I`	MH-P	✓/✓	✓/✓	✓/✓	✓/✓
`Model J`	MH-P	✓/	✓/	✓/✓	✓/✓
`Model K`	MH-P	✓/	✓/	✓/	✓/✓
`Model L`	MH-V	✓/✓	✓/✓		
`Model M`	MH-V	✓/✓	✓/✓	✓/✓	
`Model N`	MH-V	✓/✓	✓/✓	✓/✓	✓/✓
`Model O`	MH-V	✓/	✓/	✓/✓	✓/✓
`Model P`	MH-V	✓/	✓/	✓/	✓/✓

Models	ScanObjectNN	ShapeNetPart	S3DIS Area5
`Model G`	82.3	85.4	-
`Model H`	85.6	86.3	-
`Model I`	87.8	87.1	-
`Model J`	87.4	86.8	-
`Model K`	87.1	86.6	-
`Model L`	-	-	62.3
`Model M`	-	-	68.4
`Model N`	-	-	72.7
`Model O`	-	-	72.0
`Model P`	-	-	71.6

ShapeNetPart. **`Model C`**, **`F`** leverages pseudo scenes synthesized through S2SS for pre-training data, with PPC as the loss function. In the lower part of Table 6.6, we present the performance on shape-level tasks (ScanObjectNN, ShapeNetPart) and scene-level tasks (S3DIS Area5). On ScanObjectNN, **`Model B`** aggregates deep semantic features at multiple scales, outperforming **`Model A`**. This proves that MH-P is a strong backbone. For high-resolution tasks, we benefit from the design of multi-scale feature aggregation with high-resolution mapping. Both **`Model B`** and **`E`** exhibit better performance than **`Model A`** and **`D`**. Compared to training-from-scratch baselines, **`Model C`** achieves improvements of 1.9% and 1.1% on ScanObjectNN and ShapeNetPart, respectively. **`Model F`** demonstrated a 2.4% improvement on S3DIS compared to **`Model E`**. This highlights the efficacy of our proposed pre-training strategy, *i.e.*, S2SS and PPC, for high-resolution tasks.

Network Design. Then we present the performance of models that have different network designs. We have investigated four scales, denoted as $S_1 = 2$, $S_2 = 4$, $S_3 = 8$, and $S_4 = 16$. The upper part of Table 6.7 details the configurations of scales. Specifically, whether to use a certain scale and whether to integrate its feature into the final output. **`Model G-K`** are pre-trained on ShapeNet[282] with PPC. **`Model L-P`** synthesize pseudo scene with S2SS on ShapeNet[282] under the supervision of PPC. In the lower part of Table 6.7, we report the performance. MH-P and MH-V achieve optimal performance when using four scales. Comparing **`Model G-K`** and **`Model L-P`**, we find that

TABLE 6.8: **Ablation study** on pre-training data for the scene-level task (Section 6.3.4). We report mIoU (%).

Models	Backbone	Data Type	Source	S3DIS Area5
`Model Q`	MH-V	Shape	ShapeNet[282]	70.5
`Model R`	MH-V	Scene	ShapeNet[282]	72.7
`Model S`	MH-V	Scene	ScanNet[13]	73.5
`Model T`	MH-V	Scene	UHD[139]	74.1

TABLE 6.9: **Ablation study for** M (Section 6.3.4). We report OA (%) on ScanObjectNN, Inst. mIoU (%) on ShapeNetPart, and mIoU (%) on S3DIS.

M	ScanObjectNN	ShapeNetPart	S3DIS Area5
1	86.8	86.2	70.5
2	87.4	86.7	71.1
4	87.8	87.1	71.9
6	87.7	86.9	72.7
8	87.7	86.7	72.5
10	87.6	87.0	72.6

integrating high-resolution features from multiple scales into the final output results in improved performance. This aligns with the key advantages discussed in Section 6.1.

Pre-Training Data for Scene-Level Task. Furthermore, we have conducted research on pre-training data for the scene-level task (S3DIS Area5). **`Model Q`**, **`R`**, **`S`**, and **`T`** have all used MH-V as the backbone and have been pre-trained on ShapeNet (shape data), pseudo scenes made of ShapeNet (scene data), ScanNet from PointContrast[13] (scene data), and pseudo scenes made of UHD (scene data), respectively. **`Model Q`** shows only marginal improvement compared to **`Model E`**. Pre-training on shape-level data exhibit minimal potential in downstream scene-level tasks. However, **`Model R`** demonstrates significantly enhanced performance compared to **`Model Q`**, indicating the effectiveness of S2SS. Additionally, **`Model T`** even outperforms **`Model S`** (pre-training on ScanNet) with larger shape-level datasets (*i.e.*, UHD and LHD[139]). Considering the time-consuming preprocessing required for generating matched pairs in PointContrast[13], our approach can be integrated into the training process to obtain matched pairs in running time, which is highly valuable.

Number of Shapes M. Finally, we conduct ablation experiments to explore M in S2SS on three downstream tasks. The pre-training settings, including pre-training data, S2SS, MH-P/V, loss function, *etc*., remain consistent with **`Model I`** and **`N`**. In downstream shape-level tasks, the pre-trained model demonstrates its peak performance at $M = 4$. In contrast, for scene-level segmentation, the optimal performance of the pre-trained model is observed at $M = 6$. When it comes to larger values of M, neither **`Model I`** nor **`N`** show improved performance. Intuitively, we attribute this to the number of matched pairs utilized for the contrastive loss. However, indiscriminately increasing the number of matched pairs entails in increased resource and time consumption [13], hence we empirically set $M = 4$ for **`Model I`** and $M = 6$ for **`Model N`**.

6.4 Summary

In summary, our research introduces S2S, a novel 3D scene representation learning method. By leveraging readily available 3D shape datasets and introducing innovative architectural designs, data strategies, and loss functions, S2S overcomes the *data desert* issue and bridges the gap between shape-level and scene-level datasets. This approach demonstrates remarkable adaptability and transferability across both shape-level and scene-level datasets, as evidenced by the impressive performance achieved on eight well-established benchmarks. S2S represents a significant step forward in addressing the challenges faced by current 3D SSL for scene understanding. However, despite its notable achievements, challenges still lie ahead, particularly in extending the application of S2S to broader 3D contexts, such as open-world scene understanding. Expanding S2S's capabilities to encompass these more complex scenarios will require further research and innovation. We intend to delve into these challenges in our future research.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis has addressed critical challenges in 3D computer vision, including feature extraction, clustering, scalability, and interpretability, by introducing novel methods and architectures. Key contributions include Cluster3D, a clustering-based representation learning framework for fine-grained pattern discovery in large-scale 3D datasets; LSK3DNet, a 3D backbone network leveraging spatial-wise dynamic sparsity and channel-wise weight selection for efficient and accurate 3D vision tasks; Interpretable3D, a prototype-based interpretable model that bridges the gap between black-box models and transparent decision-making; and Shape2Scene, a scalable pretraining strategy that enables effective shape-to-scene transfer learning. These contributions were validated through extensive experiments on multiple benchmarks, demonstrating significant improvements in point cloud segmentation, object detection, and interpretability. We summarize our main contributions below:

In Chapter 3, we devise a clustering based supervised training scheme for point cloud segmentation, which discovers and respects latent data structures during point representation learning. Rather than simply minimizing the point recognition error, we iteratively perform 1) unsupervised, within-class clustering based subclass pattern mining, and 2) clustering assignment based point embedding space optimization. Our algorithm is general and shows outstanding performance over various tasks and datasets.

In Chapter 4, we propose *Spatial-wise Dynamic Sparsity* to scale up 3D kernels beyond $9 \times 9 \times 9$, which prunes the volumetric weight and reduces the parameter size of large kernel layers. Our LSK3DNet can benefit from a large receptive field without increasing the computational cost compared to a naive 3D large kernel. *Channel-wise Weight Selection* expands the model width during training, and then sorts and selects important channels during validation to get a model of the expected size. In this way, we achieve *"using spatial sparse groups, expanding width without more parameters"*. We evaluate our method on the SemanticKITTI and achieve state-of-the-art performance. Our LSK3DNet also surpasses previous 3D large kernel methods on ScanNet v2 and KITTI.

In Chapter 5, we develop an *ad-hoc* interpretable classifier, *i.e.*, Interpretable3D, specifically designed for point cloud classification and part segmentation tasks. By performing Prototype Estimation and Optimization, Interpretable3D benefits from a reshaped prototype feature space and the preserved typical prototypes. The revealed decision-making mode enables users to understand how the system works and how decisions are made. Prototype Optimization further enhances the interpretability by illustrating how prototypes can be modified in a manner easily comprehensible for humans. Our algorithm consistently produces promising results across three datasets.

In Chapter 6, our research introduces S2S, a novel 3D scene representation learning method. By leveraging readily available 3D shape datasets and introducing innovative architectural designs, data strategies, and loss functions, S2S overcomes the *data desert* issue and bridges the gap between shape-level and scene-level datasets. This approach demonstrates remarkable adaptability and transferability across both shape-level and scene-level datasets, as evidenced by the impressive performance achieved on eight well-established benchmarks. S2S represents a significant step forward in addressing the challenges faced by current 3D SSL for scene understanding.

Collectively, these methodologies provide solutions to long-standing challenges, advancing the field of 3D computer vision and laying the groundwork for future research.

7.2 Future Work

Despite the progress achieved in this thesis, several avenues remain for future exploration.

- Efficiency Optimization: Develop lightweight variants of LSK3DNet using neural architecture search or quantization for edge-device deployment.
- Generalizable Interpretability: Extend Interpretable3D to instance-aware segmentation, tracking, and detection tasks, and automate prototype discovery to reduce manual intervention.
- Dynamic and Multi-modal Fusion: Integrate temporal modeling into LSK3DNet for 4D point cloud processing and explore fusion with LiDAR and RGB data.
- Unified Framework: By integrating Cluster3D's subclass mining, LSK3DNet's large-kernel backbone, Interpretable3D's prototype-level explanations, and Shape2Scene's shape-to-scene pre-training into a single framework, we can form a cohesive, end-to-end pipeline for 3D vision.
- Real-World Impact: Together, these techniques will yield a robust 3D-vision system that is ready for deployment in embodied AI, autonomous-driving, and augmented-reality applications.

These future directions aim to build upon the contributions of this thesis, driving further advancements in 3D vision research and practical applications.

Appendix A

Appendix of Cluster3D

A.1 Quantitative Results of Cluster3D

Complete Quantitative Result on SemanticKITTI Single-Scan Challenge `test`. Table A.1 and Table A.2 report the complete results on SemanticKITTI [1] single-scan challenge `test`. Our method reaches 70.4% mIoU, which yields 2.6% mIoU gains over Cylinder3D [6]. Moreover, it also outperforms many famous segmentation models, such as AF2S3Net [98] and RPVNet [91]. One more thing to point out, spvnas[1] did not provide the source code of 3D-NAS pipeline and the control file for $\text{SPVNAS}_{\text{12.5M}}$. But the control file and pretrained models for $\text{SPVNAS}_{\text{10.8M}}$ are shared[2]. And the difference between $\text{SPVNAS}_{\text{12.5M}}$ and $\text{SPVNAS}_{\text{10.8M}}$ is that $\text{SPVNAS}_{\text{10.8M}}$ is trained except sequence 08. As for our implementation, $\text{SPVNAS}_{\text{10.8M}}$ and $\text{SPVNAS}_{\text{10.8M}}$ + **`Ours`** are trained on sequences 00-10 and evaluated on 11-21.

Complete Quantitative Result on S3DIS Area-5. Table A.3 and Table A.4 present the complete per-class IoU on S3DIS [2] Area-5. Both CBL [206] and our method use contrastive loss on the premise of fully supervised learning. But [206] only samples negative points locally around the boundaries, while we contrast global subclass centers against the points sampled from the ENTIRE training dataset. Our idea is much more powerful and insightful. The fair comparison based

[1]`https://github.com/mit-han-lab/spvnas/`

[2]SPVNAS has cancelled the download link for the Control file and $\text{SPVNAS}_{\text{10.8M}}$ model. Instead, we will release the two previously downloaded files.

on PTV1 [7] shows that our approach attains mIoU/mAcc/OA of 72.2%/79.6%, outperforming PTV1+CBL (71.6%/77.9%).

Complete Quantitative Result on SemanticKITTI multi-scan challenge `test`. Table A.5 and Table A.6 report the complete results on SemanticKITTI [1] multi-scan challenge `test`. With Cylinder3D, our algorithm also attains consistent performance improvements of 2.2% mIoU, just like that in single-scan `test`. Moreover, Cylinder3D+ **`Ours`** surpasses Cylinder3D in 17 classes out of 25 classes.

A.2 Qualitative Results of Cluster3D

Qualitative Results of Cluster3D for Segmetation. We show more qualitative results on SemanticKITTI [1] single-scan challenge `val` (Figure A.1), S3DIS [2] Area-5 (Figure A.2) and SemanticKITTI [1] multi-scan challenge `val` (Figure A.3). As observed, our approach generally gives more accurate predictions compared with vanilla PTV1[7] and Cylinder3D[6]. In Figure A.2, vanilla PTV1 fails to recognize region boundaries and tends to misclassify board-like objects, while our method can significantly reduce these errors. Figure A.3 depicts qualitative comparisons of Cylinder3D and Cylinder3D + **`Ours`** over lidar sequences on SemanticKITTI multi-scan challenge `val`. Note that, the predicted labels of five consecutive frames are displayed in one frame. It can be observed that Cylinder3D + **`Ours`** has smaller errors over the semantic boundaries as well as classes belonging to ground and nature.

TABLE A.1: **Quantitative results** on SemanticKITTI [1] single-scan challenge `test` (§A.1) - Part I. mIoU (%) and IoUs (%) are reported.

Method	mIoU	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle
TangentConv [CVPR18] [73]	40.9	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1
SqueezeSegV2 [ICRA19] [74]	39.7	88.6	67.6	45.8	17.7	73.7	81.8	13.4	18.5	17.9	14.0
DarkNet53 [ICCV19] [1]	49.9	91.8	74.6	64.8	27.9	84.1	86.4	25.5	24.5	32.7	22.6
Rangenet++ [IROS19] [75]	52.2	91.8	75.2	65.0	27.8	87.4	91.4	25.7	25.7	34.4	23.0
3D-MiniNet [IROS20] [291]	55.8	91.6	74.5	64.2	25.4	89.4	90.5	28.5	42.3	42.1	29.4
PointASNL [CVPR20] [46]	46.8	87.4	74.3	24.3	1.8	83.1	87.9	39.0	0.0	25.1	29.2
PolarNet [CVPR20] [71]	54.3	90.8	74.4	61.7	21.7	90.0	93.8	22.9	40.3	30.1	28.5
RandLA-Net [CVPR20] [47]	55.9	90.5	74.0	61.8	24.5	89.7	94.2	43.9	29.8	32.2	39.1
SqueezeSegV3 [ECCV20] [79]	55.9	91.7	74.8	63.4	26.4	89.0	92.5	29.6	38.7	36.5	33.0
SalsaNext [ISVC20] [78]	59.5	91.7	75.8	63.7	29.1	90.2	91.9	38.9	48.3	38.6	31.9
FusionNet [ECCV20] [86]	61.3	91.8	77.1	68.8	30.8	92.5	95.3	41.8	47.5	37.7	34.5
JS3C-Net [AAAI21] [202]	66.0	88.9	72.1	61.9	31.9	92.5	95.8	**54.3**	59.3	52.9	46.0
AF2S3Net [CVPR21] [98]	69.7	91.3	72.5	68.8	**53.5**	87.9	94.5	39.2	65.4	**86.8**	41.1
RPVNet [ICCV21] [91]	70.3	**93.4**	**80.7**	70.3	33.3	**93.5**	**97.6**	44.2	**68.4**	68.7	**61.1**
PVKD [CVPR22] [203]	**71.4**	91.8	77.5	**70.9**	41.0	92.4	97.0	53.5	67.9	69.3	60.2
KPConv [ICCV19] [64]	58.8	88.8	72.7	61.3	31.6	90.5	96.0	33.4	30.2	42.5	44.3
KPConv + `Ours`	61.0	89.9	75.0	63.4	34.3	91.4	88.8	49.0	45.0	46.6	45.5
SPVNAS$_{10.8M}$ [ECCV20] [87]	62.3	89.6	73.8	63.2	29.1	90.9	96.7	50.9	40.6	42.1	51.3
SPVNAS$_{10.8M}$ + `Ours`	64.3	89.6	73.9	64.0	28.8	91.4	96.7	48.0	48.9	50.5	51.0
Cylinder3D [CVPR21] [6]	67.8	91.4	75.5	65.1	32.3	91.0	97.1	50.8	67.6	64.0	58.6
Cylinder3D + `Ours`	70.4	91.7	77.2	66.1	34.1	92.3	97.0	51.9	**68.4**	65.8	58.8

TABLE A.2: **Quantitative results** on SemanticKITTI [1] single-scan challenge `test` (§A.1) - Part II. mIoU (%) and IoUs (%) are reported.

Method	mIoU	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic-sign
TangentConv [CVPR18] [73]	40.9	79.5	49.3	58.1	23.0	28.4	8.1	49.0	35.8	28.5
SqueezeSegV2 [ICRA19] [74]	39.7	71.8	35.8	60.2	20.1	25.1	3.9	41.1	20.2	26.3
DarkNet53 [ICCV19] [1]	49.9	78.3	50.1	64.0	36.2	33.6	4.7	55.0	38.9	52.2
Rangenet++ [IROS19] [75]	52.2	80.5	55.1	64.6	38.3	38.8	4.8	58.6	47.9	55.9
3D-MiniNet [IROS20] [291]	55.8	82.8	60.8	66.7	47.8	44.1	14.5	60.8	48.0	56.6
PointASNL [CVPR20] [46]	46.8	84.1	52.2	70.6	34.2	57.6	0.0	43.9	57.8	36.9
PolarNet [CVPR20] [71]	54.3	84.0	65.5	67.8	43.2	40.2	5.6	61.3	51.8	57.5
RandLA-Net [CVPR20] [47]	55.9	83.8	63.6	68.6	48.4	47.4	9.4	60.4	51.0	50.7
SqueezeSegV3 [ECCV20] [79]	55.9	82.0	58.7	65.4	45.6	46.2	20.1	59.4	49.6	58.9
SalsaNext [ISVC20] [78]	59.5	81.8	63.6	66.5	60.2	59.0	19.4	64.2	54.3	62.1
FusionNet [ECCV20] [86]	61.3	84.5	69.8	68.5	59.5	56.8	11.9	69.4	60.4	66.5
JS3C-Net [AAAI21] [202]	66.0	84.5	69.8	67.9	69.5	65.4	39.9	70.8	60.7	68.7
AF2S3Net [CVPR21] [98]	69.7	70.2	68.5	53.7	**80.7**	**80.4**	**74.3**	63.2	61.5	71.0
RPVNet [ICCV21] [91]	70.3	86.5	**75.1**	71.7	75.9	74.4	43.4	**72.1**	64.8	61.4
PVKD [CVPR22] [203]	**71.4**	86.5	73.8	**71.9**	75.1	73.5	50.5	69.4	64.9	61.4
KPConv [ICCV19] [64]	58.8	84.8	69.2	69.1	61.5	61.6	11.8	64.2	56.4	47.4
KPConv + `Ours`	61.0	72.0	56.5	68.8	59.4	60.1	36.4	66.1	49.5	60.4
SPVNAS$_{10.8M}$ [ECCV20] [87]	62.3	85.5	70.3	69.8	60.4	62.8	21.8	65.3	57.6	62.0
SPVNAS$_{10.8M}$ + `Ours`	64.3	85.3	72.1	69.1	67.1	70.5	23.2	67.0	60.7	64.5
Cylinder3D [CVPR21] [6]	67.8	85.4	71.8	68.5	73.9	67.9	36.0	66.5	62.6	65.6
Cylinder3D + `Ours`	70.4	**86.7**	73.5	71.7	69.6	70.1	54.6	70.8	**65.1**	**71.6**

TABLE A.3: **Quantitative results** on S3DIS [2] Area-5 (§A.1) - Part I. mIoU (%) and IoUs (%) are reported.

Method	mIoU	mAcc	OA	ceiling	floor	wall	beam	column	window	door
PointNet [CVPR17] [14]	41.1	49.0	-	88.8	97.3	69.8	0.1	3.9	46.3	10.8
SegCloud [3DV17] [192]	48.9	57.4	-	90.1	96.1	69.9	0.0	18.4	38.4	23.1
TangentConv [CVPR18] [73]	52.6	62.2	-	90.5	97.7	74.0	0.0	20.7	39.0	31.3
PointCNN [NeurIPS18] [39]	57.3	63.9	85.9	92.3	98.2	79.4	0.0	17.6	22.8	62.1
SPGraph [CVPR18] [52]	58.0	66.5	86.4	89.4	96.9	78.1	0.0	42.8	48.9	61.6
PCCN [CVPR18] [195]	58.3	-	67.0	92.3	96.2	75.9	0.3	6.0	69.5	63.5
HPEIN [ICCV19] [45]	61.9	68.3	87.2	91.5	98.2	81.4	0.0	23.3	65.3	40.0
PAT [CVPR19] [67]	60.1	70.8	-	93.0	98.5	72.3	1.0	41.5	85.1	38.2
PointWeb [CVPR19] [44]	60.3	66.6	87.0	92.0	98.5	79.4	0.0	21.1	59.7	34.8
MinkowskiNet [CVPR19] [18]	65.4	71.7	-	91.8	98.7	86.2	0.0	34.1	48.9	62.4
SCF-Net [CVPR21] [48]	63.8	-	-	-	-	-	-	-	-	-
BAAF-Net [CVPR21] [204]	65.4	73.1	88.9	-	-	-	-	-	-	-
CGA-Net [CVPR21] [205]	68.6	-	-	94.5	98.3	83.0	0.0	25.3	59.6	71.0
Stratified Trans. [CVPR22] [207]	72.0	78.1	91.5	-	-	-	-	-	-	-
PTV2 [NeurIPS22] [208]	**72.6**	78.0	**91.6**	-	-	-	-	-	-	-
KPConv [ICCV19] [64]	67.1	72.8	-	92.8	97.3	82.4	0.0	23.9	58.0	69.0
KPConv+ `Ours`	69.0	76.2	90.5	95.7	98.3	84.0	0.0	30.7	66.7	77.6
PTV1 [ICCV21] [7]	70.4	76.5	90.8	94.0	98.5	86.3	0.0	38.0	63.4	74.3
PTV1+CBL [CVPR22] [206]	71.6	77.9	91.2	-	-	-	-	-	-	-
PTV1+ `Ours`	72.2	**79.6**	91.2	94.2	98.4	88.1	0.0	49.3	65.3	79.4

TABLE A.4: **Quantitative results** on S3DIS [2] Area-5 (§A.1) - Part II. mIoU (%) and IoUs (%) are reported.

Method	mIoU	mAcc	OA	table	chair	sofa	bookcase	board	clutter
PointNet [CVPR17] [14]	41.1	49.0	-	52.6	58.9	40.3	5.9	26.4	33.3
SegCloud [3DV17] [192]	48.9	57.4	-	70.4	75.9	40.9	58.4	13.0	41.6
TangentConv [CVPR18] [73]	52.6	62.2	-	77.5	69.4	57.3	38.5	48.8	39.8
PointCNN [NeurIPS18] [39]	57.3	63.9	85.9	74.4	80.6	31.7	66.7	62.1	56.7
SPGraph [CVPR18] [52]	58.0	66.5	86.4	84.7	75.4	69.8	52.6	2.1	52.2
PCCN [CVPR18] [195]	58.3	-	67.0	66.9	65.6	47.3	68.9	59.1	46.2
HPEIN [ICCV19] [45]	61.9	68.3	87.2	75.5	87.7	58.5	67.8	65.6	49.4
PAT [CVPR19] [67]	60.1	70.8	-	57.7	83.6	48.1	67.0	61.3	33.6
PointWeb [CVPR19] [44]	60.3	66.6	87.0	76.3	88.3	46.9	69.3	64.9	52.5
MinkowskiNet [CVPR19] [18]	65.4	71.7	-	81.6	89.8	47.2	74.9	74.4	58.6
SCF-Net [CVPR21] [48]	63.8	-	-	-	-	-	-	-	-
BAAF-Net [CVPR21] [204]	65.4	73.1	88.9	-	-	-	-	-	-
CGA-Net [CVPR21] [205]	68.6	-	-	82.6	92.2	77.7	76.4	69.5	61.5
Stratified Trans. [CVPR22] [207]	72.0	78.1	91.5	-	-	-	-	-	-
PTV2 [NeurIPS22] [208]	**72.6**	78.0	**91.6**	-	-	-	-	-	-
KPConv [ICCV19] [64]	67.1	72.8	-	81.5	91.0	75.4	75.3	66.7	58.9
KPConv+ `Ours`	69.0	76.2	90.8	79.9	91.0	70.3	76.7	63.0	63.6
PTV1 [ICCV21] [7]	70.4	76.5	90.8	89.1	82.4	74.3	80.2	76.0	59.3
PTV1+CBL [CVPR22] [206]	71.6	77.9	91.2	-	-	-	-	-	-
PTV1+ `Ours`	72.2	**79.6**	91.2	89.4	82.2	74.8	77.6	81.0	58.7

TABLE A.5: **Quantitative results** on SemanticKITTI [1] multi-scan challenge `test` (§A.1) - Part I. mIoU (%) and IoUs (%) are reported.

Method	mIoU	road	sidewalk	parking	other-ground	building	car	moving car	truck	moving truck	bicycle	motorcycle	other-vehicle	moving other-vehicle
TangentConv [CVPR18] [73]	34.1	83.9	64.0	38.3	15.3	85.8	84.9	40.3	21.1	1.1	2.0	18.2	18.5	6.4
DarkNet53 [ICCV19] [1]	41.6	91.6	75.3	64.9	27.5	85.2	84.1	61.5	20.0	14.1	30.4	32.9	20.7	15.2
TemporalLidarSeg [3DV20] [22]	47.0	**91.8**	75.8	59.6	23.2	89.8	92.1	68.2	39.2	2.1	47.7	40.9	35.0	12.4
SpSeqnet [CVPR20] [23]	43.1	90.1	73.9	57.6	27.1	91.2	88.5	53.2	29.2	**41.2**	24.0	26.2	22.7	**26.2**
KPConv [ICCV19] [64]	51.2	86.5	70.5	58.4	26.7	90.8	93.7	69.4	42.5	5.8	44.9	47.2	38.6	4.7
KPConv+ **`Ours`**	53.2	90.4	75.2	62.1	25.1	91.8	**95.8**	75.2	**43.8**	4.1	67.2	63.1	**44.2**	0.7
Cylinder3D [CVPR21] [6]	52.5	90.7	74.5	65.0	**32.3**	**92.6**	94.6	74.9	41.3	0.0	**67.6**	**63.8**	38.8	0.1
Cylinder3D+ **`Ours`**	**54.7**	91.4	**76.9**	**66.1**	27.8	91.4	95.3	**81.7**	42.7	11.9	55.9	52.9	38.7	11.2

TABLE A.6: **Quantitative results** on SemanticKITTI [1] multi-scan challenge `test` (§A.1) - Part II. mIoU (%) and IoUs (%) are reported.

Method	mIoU	vegetation	trunk	terrain	person	moving person	bicyclist	moving bicyclist	motorcyclist	moving motorcyclist	fence	pole	traffic-sign
TangentConv [CVPR18] [73]	34.1	79.5	43.2	56.7	1.6	1.9	0.0	30.1	0.0	42.2	49.1	36.4	31.2
DarkNet53 [ICCV19] [1]	41.6	78.4	50.7	64.8	7.5	0.2	0.0	28.9	0.0	37.8	56.5	38.1	53.3
TemporalLidarSeg [3DV20] [22]	47.0	82.3	62.5	64.7	14.4	40.4	0.0	42.8	0.0	12.9	63.8	52.6	60.4
SpSeqnet [CVPR20] [23]	43.1	84.0	66.0	65.7	6.3	36.2	0.0	2.3	0.0	0.1	66.8	50.8	48.7
KPConv [ICCV19] [64]	51.2	84.6	70.3	66.0	**21.6**	**67.5**	0.0	67.4	0.0	**47.2**	64.5	57.0	53.9
KPConv+ **`Ours`**	53.2	85.4	71.1	69.3	10.7	72.1	0.0	**68.5**	**9.9**	9.9	**67.5**	62.6	64.6
Cylinder3D [CVPR21] [6]	52.5	85.8	72.0	68.9	12.5	65.7	**1.7**	68.3	0.2	11.9	66.0	63.1	61.4
Cylinder3D+ **`Ours`**	**54.7**	**86.5**	**72.7**	**71.6**	15.5	61.8	0.0	68.2	3.0	46.0	66.1	**64.0**	**68.0**

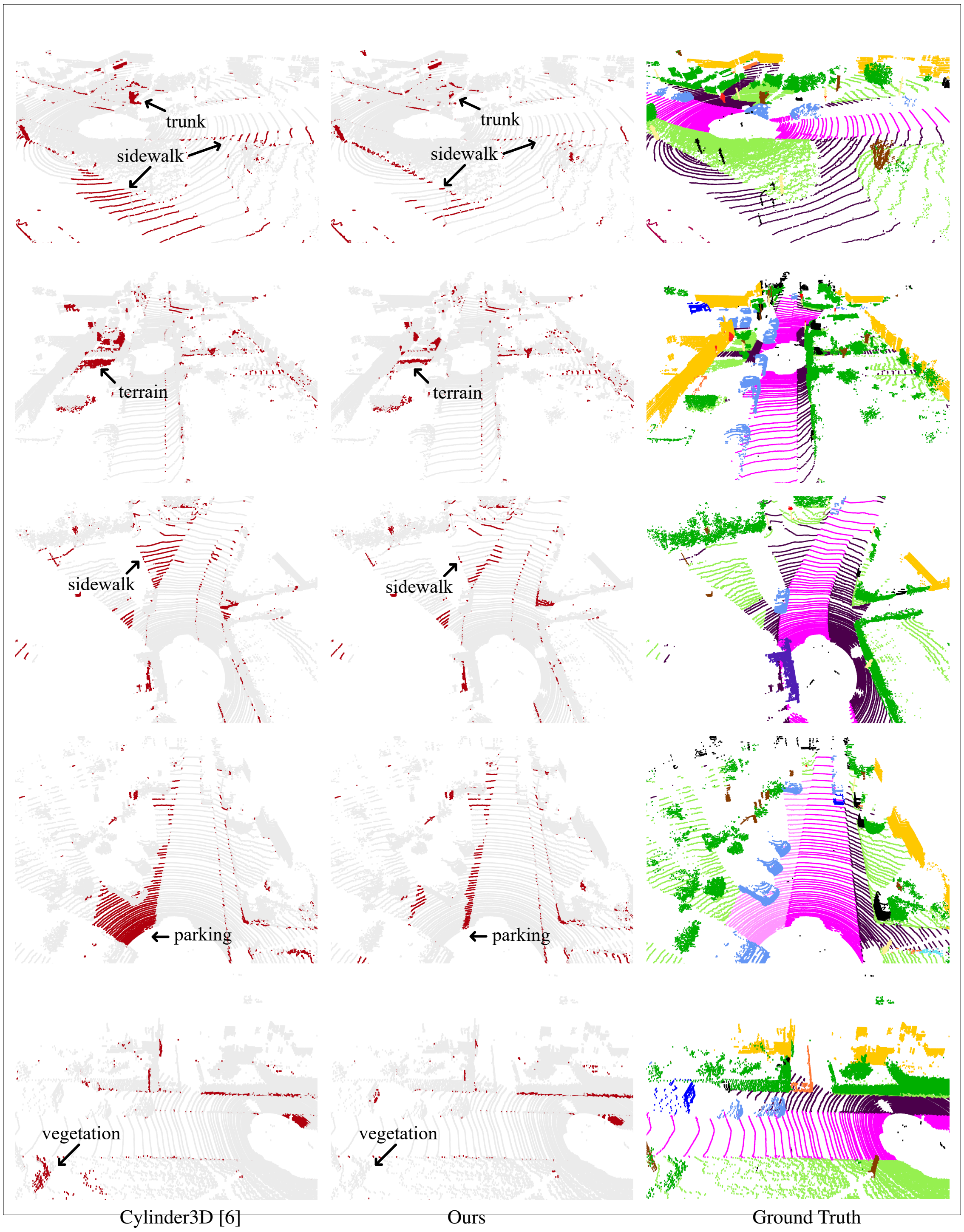


FIGURE A.1: **Error maps** of Cylinder3D [6] and Ours on SemanticKITTI [1] single-scan challenge `val` (§A.2). The differences are as illustrated by arrows.

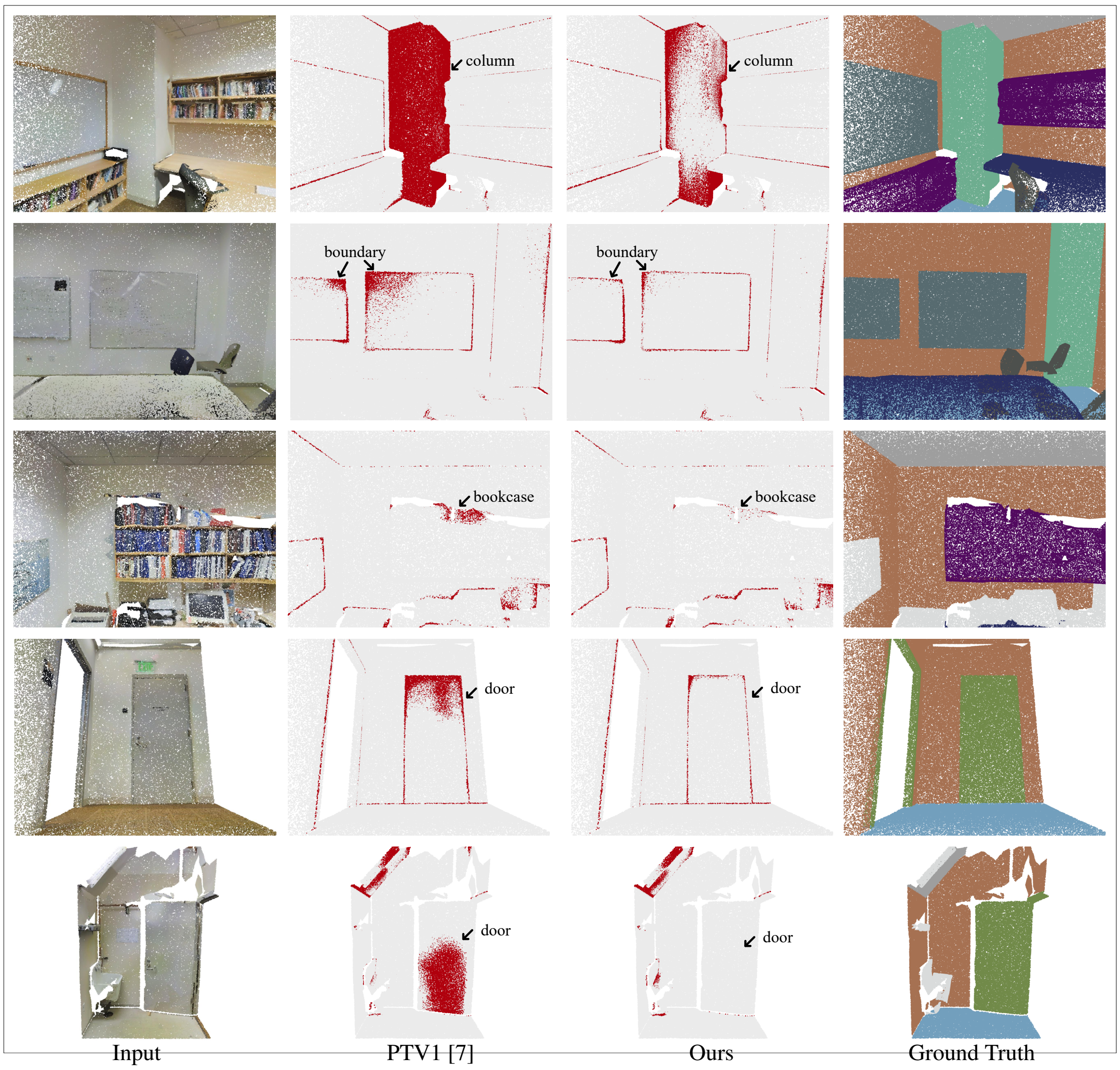


FIGURE A.2: **Error maps** of PTV1 [7] and Ours on S3DIS [2] Area-5 (§A.2). The differences are as illustrated by arrows.

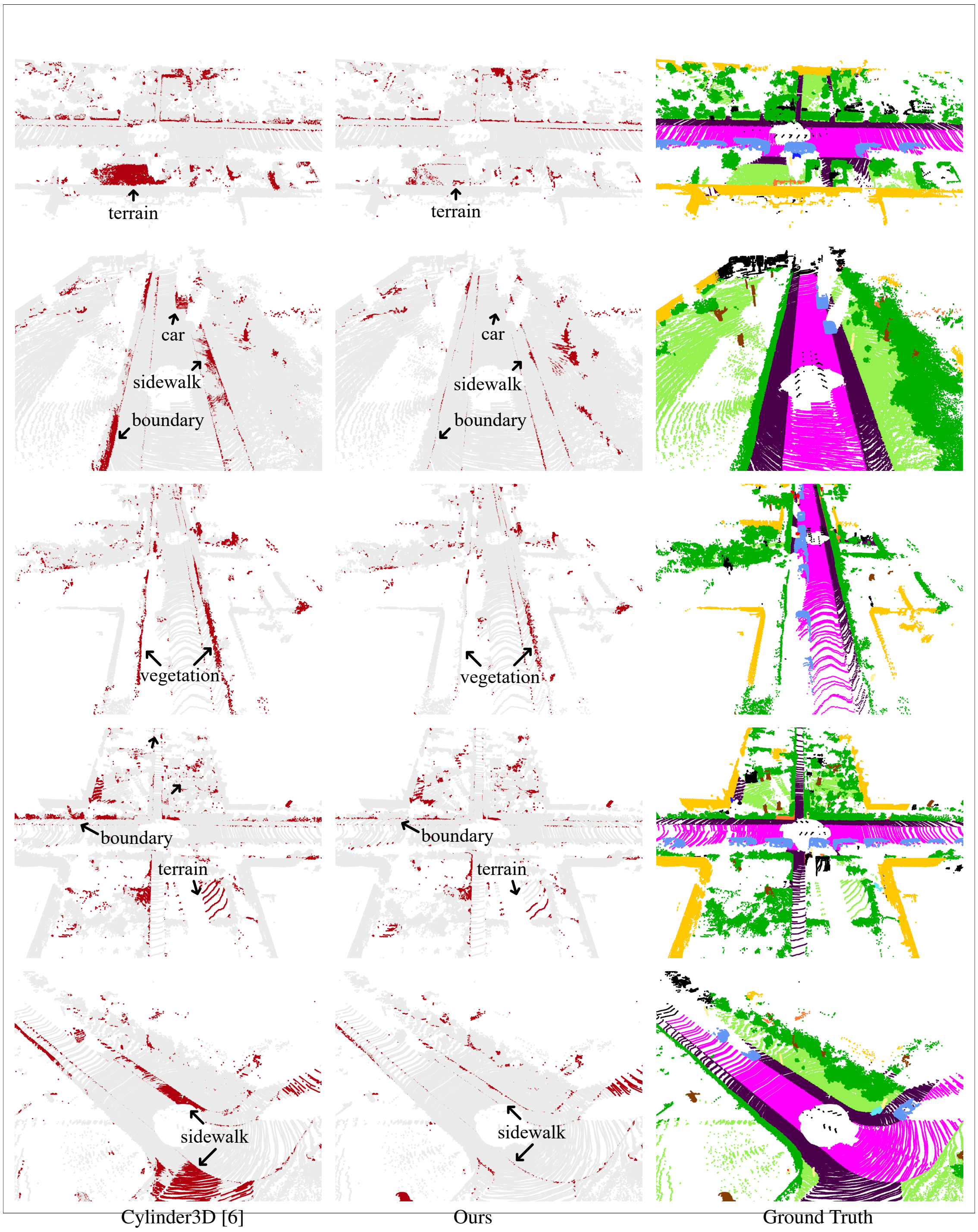


FIGURE A.3: **Error maps** of Cylinder3D [6] and Ours on SemanticKITTI [1] multi-scan challenge `val` (§A.2). The differences are as illustrated by arrows.

Appendix B

Appendix of LSK3DNet

B.1 Qualitative Results of LSK3DNet

Qualitative Detection Results of LSK3DNet for KITTI. We compare the visualization results between LSK3DNet and Voxel R-CNN for the car category on the KITTI *val* split [8], as illustrated in Fig. B.1. The blue-bordered boxes depict ground truth bounding boxes, while the green-bordered boxes represent predicted bounding boxes. In comparison to Voxel R-CNN, LSK3DNet demonstrates more accurate prediction results.

Qualitative Results of LSK3DNet for SemanticKITTI. We provide visual examples of our algorithm on two challenges for 3D semantic segmentation: SemanticKITTI [1] single-scan `val` challenge and SemanticKITTI [1] multi-scan `test` challenge. The corresponding figures are Fig. B.2 and Fig. B.3, respectively. Moreover, we display the predicted labels of three successive frames in one single frame in Fig. B.3. It can be seen that our LSK3DNet produces more accurate and consistent predictions than the baseline (Modified SPVCNN [92]). By a larger kernel size, our LSK3DNet expands the receptive field of submanifold convolution and enhances the flow of discrete spatial information. This results in better capturing the object boundaries and distinguishing between different semantic classes. As shown, our LSK3DNet can segment the ground classes and natural objects more effectively.

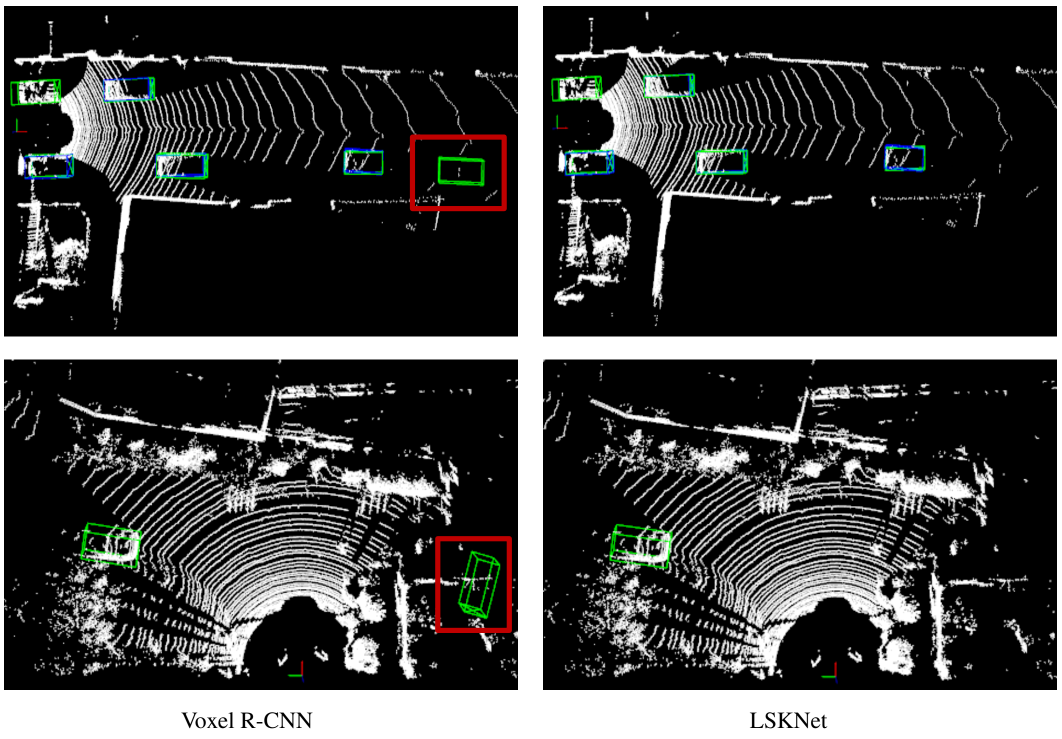


FIGURE B.1: **Qualitative detection results** of LSKNet and Voxel R-CNN on the KITTI *val* split (§B). Results in the red box are false positives.

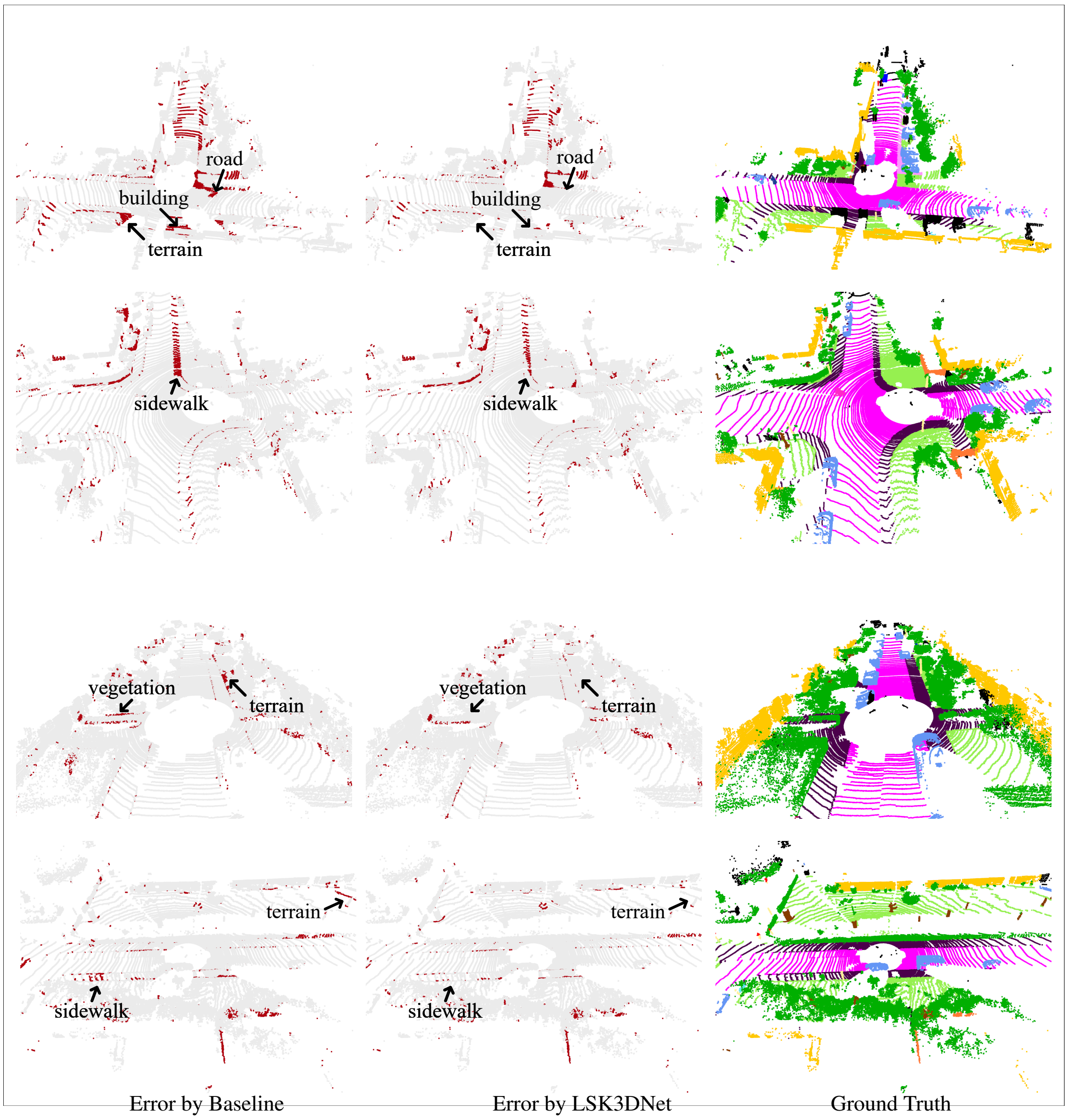


FIGURE B.2: **Error maps** of Baseline and Ours on SemanticKITTI [1] single-scan `val` (§B).

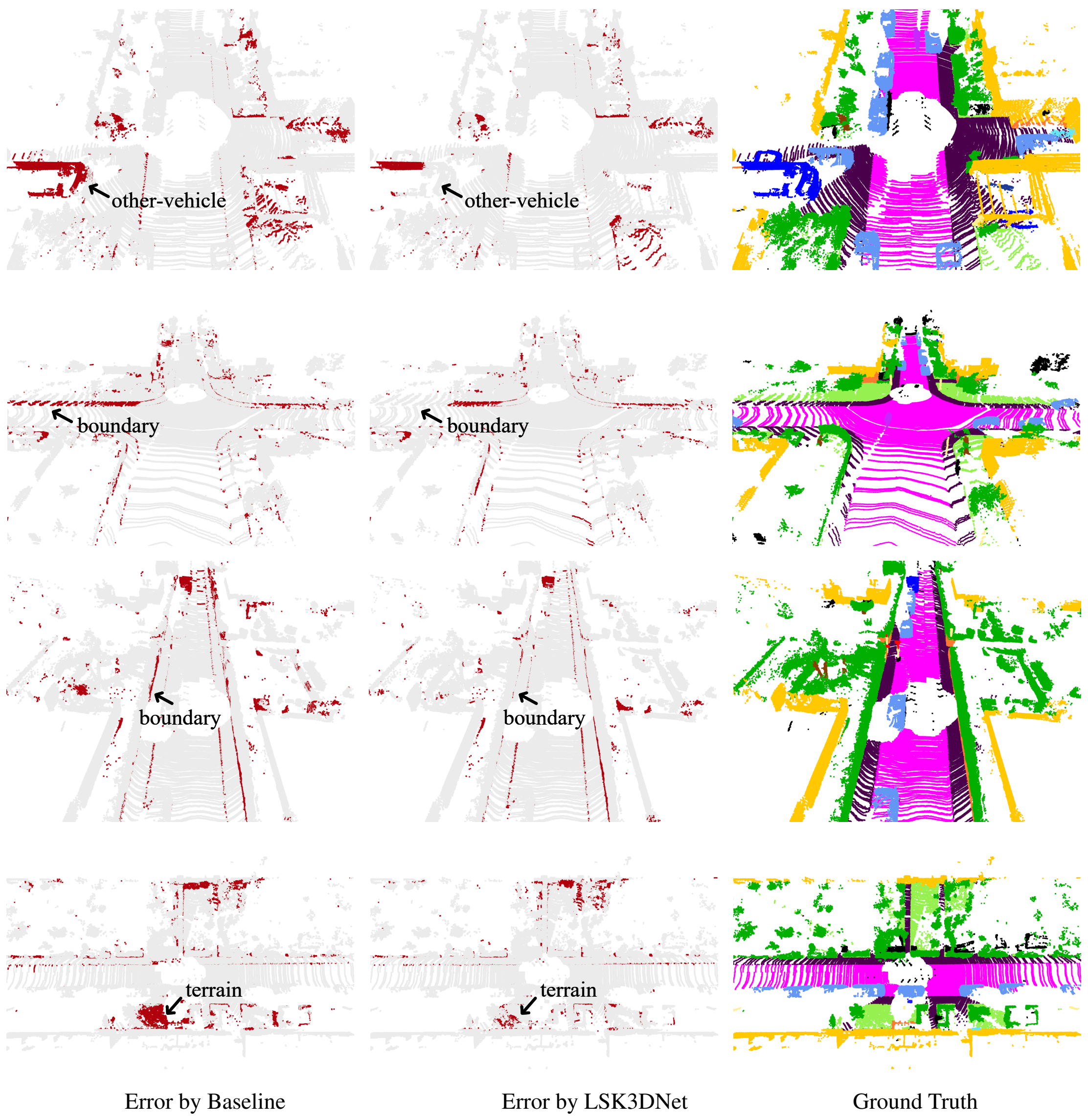


FIGURE B.3: **Error maps** of Baseline and Ours on SemanticKITTI [1] multi-scan `val` (§B).

Bibliography

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019.

[2] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, 2016.

[3] Yukang Chen, Jianhui Liu, Xiaojuan Qi, Xiangyu Zhang, Jian Sun, and Jiaya Jia. Scaling up kernels in 3d cnns. *CVPR*, 2023.

[4] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[5] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019.

[6] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *CVPR*, 2021.

[7] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021.

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[10] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[11] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM TOG*, 35(6):1–12, 2016.

[12] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[13] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *ECCV*, 2020.

[14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[15] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[16] Tuo Feng, Wenguan Wang, Xiaohan Wang, Yi Yang, and Qinghua Zheng. Clustering based point cloud representation learning for 3d analysis. In *ICCV*, 2023.

[17] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *NeurIPS*, 2019.

[18] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019.

[19] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. Pstnet: Point spatio-temporal convolution on point cloud sequences. In *ICLR*, 2021.

[20] Hehe Fan, Xin Yu, Yi Yang, and Mohan Kankanhalli. Deep hierarchical representation of point cloud videos via spatio-temporal decomposition. *IEEE TPAMI*, 44(12):9918–9930, 2021.

[21] Jiahui Liu, Chirui Chang, Jianhui Liu, Xiaoyang Wu, Lan Ma, and Xiaojuan Qi. Mars3d: A plug-and-play motion-aware model for semantic segmentation on multi-scan 3d point clouds. In *CVPR*, 2023.

[22] Fabian Duerr, Mario Pfaller, Hendrik Weigel, and Jürgen Beyerer. Lidar-based recurrent 3d semantic segmentation with temporal memory alignment. In *3DV*, 2020.

[23] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *CVPR*, 2020.

[24] Yunsong Zhou, Hongzi Zhu, Chunqin Li, Tiankai Cui, Shan Chang, and Minyi Guo. Tempnet: Online semantic segmentation on large-scale point cloud series. In *ICCV*, 2021.

[25] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4d transformer networks for spatio-temporal modeling in point cloud videos. In *CVPR*, 2021.

[26] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point spatio-temporal transformer networks for point cloud video modeling. *IEEE TPAMI*, 45(2):2181–2192, 2022.

[27] Enxu Li, Sergio Casas, and Raquel Urtasun. Memoryseg: Online lidar semantic segmentation with a latent memory. In *ICCV*, 2023.

[28] Hanxiao Tan. Visualizing global explanations of point cloud dnns. In *WACV*, 2023.

[29] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. Pointcloud saliency maps. In *ICCV*, 2019.

[30] Chen Ziwen, Wenxuan Wu, Zhongang Qi, and Li Fuxin. Visualizing point cloud classifiers by curvature smoothing. In *BMVC*, 2020.

[31] Qiuxiao Chen, Pengfei Li, Meng Xu, and Xiaojun Qi. Sparse activation maps for interpreting 3d object detection. In *CVPR*, 2021.

[32] David Schinagl, Georg Krispel, Horst Possegger, Peter M Roth, and Horst Bischof. Occam's laser: Occlusion-based attribution maps for 3d object detectors on lidar data. In *CVPR*, 2022.

[33] Tuo Feng, Ruijie Quan, Xiaohan Wang, Wenguan Wang, and Yi Yang. Interpretable3d: An ad-hoc interpretable classifier for 3d point clouds. In *AAAI*, 2024.

[34] Tuo Feng, Wenguan Wang, Fan Ma, and Yi Yang. Lsk3dnet: Towards effective and efficient 3d perception with large sparse kernels. In *CVPR*, 2024.

[35] Tuo Feng, Wenguan Wang, Ruijie Quan, and Yi Yang. Shape2scene: 3d scene representation learning through pre-training on shape data. In *Proc. of European Conference on Computer Vision*. Springer, 2024.

[36] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *ICCV*, 2019.

[37] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. In *ICLR*, 2021.

[38] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In *NeurIPS*, 2022.

[39] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018.

[40] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *CVPR*, 2018.

[41] Francis Engelmann, Theodora Kontogianni, Jonas Schult, and Bastian Leibe. Know what your neighbors do: 3d semantic segmentation of point clouds. In *ECCV*, 2018.

[42] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *CVPR*, 2018.

[43] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *ICCV*, 2019.

[44] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019.

[45] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *ICCV*, 2019.

[46] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, 2020.

[47] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020.

[48] Siqi Fan, Qiulei Dong, Fenghua Zhu, Yisheng Lv, Peijun Ye, and Fei-Yue Wang. Scf-net: Learning spatial contextual features for large-scale point cloud segmentation. In *CVPR*, 2021.

[49] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017.

[50] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, 2018.

[51] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *ECCV*, 2018.

[52] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018.

[53] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. In *CVPR*, 2019.

[54] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019.

[55] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *CVPR*, 2019.

[56] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, 2019.

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

[58] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *CVPR*, 2018.

[59] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *CVPR*, 2018.

[60] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided cnn with spherical kernels for 3d point clouds. In *CVPR*, 2019.

[61] Artem Komarichev, Zichun Zhong, and Jing Hua. A-cnn: Annularly convolutional neural networks on point clouds. In *CVPR*, 2019.

[62] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S Davis. Modeling local geometric structure of 3d point clouds using geo-cnn. In *CVPR*, 2019.

[63] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019.

[64] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019.

[65] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *ICCV*, 2019.

[66] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *CVPR*, 2018.

[67] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *CVPR*, 2019.

[68] Kirill Mazur and Victor Lempitsky. Cloud transformers: A universal approach to point cloud processing tasks. In *ICCV*, 2021.

[69] Qingyong Hu, Bo Yang, Sheikh Khalid, Wen Xiao, Niki Trigoni, and Andrew Markham. Towards semantic segmentation of urban-scale 3d point clouds: A dataset, benchmarks and challenges. In *CVPR*, 2021.

[70] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *ICRA*, 2018.

[71] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *CVPR*, 2020.

[72] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 2018.

[73] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *CVPR*, 2018.

[74] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, 2019.

[75] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *IROS*, 2019.

[76] Yecheng Lyu, Xinming Huang, and Ziming Zhang. Learning to segment 3d point clouds in 2d image space. In *CVPR*, 2020.

[77] Yuqi Yang, Shilin Liu, Hao Pan, Yang Liu, and Xin Tong. Pfcnn: convolutional neural networks on 3d surfaces using parallel frames. In *CVPR*, 2020.

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

[79] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *ECCV*, 2020.

[80] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015.

[81] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017.

[82] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018.

[83] Dario Rethage, Johanna Wald, Jurgen Sturm, Nassir Navab, and Federico Tombari. Fully-convolutional point networks for large-scale point clouds. In *ECCV*, 2018.

[84] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *CVPR*, 2018.

[85] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. Vv-net: Voxel vae net with group convolutions for point cloud segmentation. In *ICCV*, 2019.

[86] Feihu Zhang, Jin Fang, Benjamin Wah, and Philip Torr. Deep fusionnet for point cloud semantic segmentation. In *ECCV*, 2020.

[87] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*, 2020.

[88] Yukang Chen, Yanwei Li, Xiangyu Zhang, Jian Sun, and Jiaya Jia. Focal sparse convolutional networks for 3d object detection. In *CVPR*, 2022.

[89] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[90] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *ICCV*, 2017.

[91] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In *ICCV*, 2021.

[92] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *ECCV*, 2022.

[93] Mohamed Afham, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *CVPR*, 2022.

[94] Longlong Jing, Yucheng Chen, Ling Zhang, Mingyi He, and Yingli Tian. Self-supervised modal and view invariant feature learning. *arXiv preprint arXiv:2005.14169*, 2020.

[95] Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In *CVPR*, 2023.

[96] Le Xue, Ning Yu, Shu Zhang, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. *arXiv preprint arXiv:2305.08275*, 2023.

[97] Shuyang Sun, Jiangmiao Pang, Jianping Shi, Shuai Yi, and Wanli Ouyang. Fishnet: A versatile backbone for image, region, and pixel level prediction. In *NeurIPS*, 2018.

[98] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. (af)2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *CVPR*, 2021.

[99] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *NeurIPS*, 2022.

[100] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. In *CVPR*, 2022.

[101] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015.

[102] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[103] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *CVPR*, 2019.

[104] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016.

[105] Yue Wang, Alireza Fathi, Abhijit Kundu, David A Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In *ECCV*, 2020.

[106] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017.

[107] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018.

[108] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *AAAI*, 2021.

[109] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *ICCV*, 2021.

[110] Junbo Yin, Dingfu Zhou, Liangjun Zhang, Jin Fang, Cheng-Zhong Xu, Jianbing Shen, and Wenguan Wang. Proposalcontrast: Unsupervised pre-training for lidar-based 3d object detection. In *ECCV*, pages 17–33, 2022.

[111] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[112] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *CVPR*, 2020.

[113] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *CVPR*, 2020.

[114] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *CVPR*, 2021.

[115] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[116] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[117] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *ICCV*, 2021.

[118] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *ICCV*, 2021.

[119] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE TPAMI*, 38(9):1734–1747, 2015.

[120] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

[121] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

[122] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[123] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

[124] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *CVPR*, 2021.

[125] Li Jiang, Shaoshuai Shi, Zhuotao Tian, Xin Lai, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Guided point contrastive learning for semi-supervised point cloud semantic segmentation. In *ICCV*, 2021.

[126] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In *NeurIPS*, 2019.

[127] Junbo Yin, Jin Fang, Dingfu Zhou, Liangjun Zhang, Cheng-Zhong Xu, Jianbing Shen, and Wenguan Wang. Semi-supervised 3d object detection with proficient teachers. In *ECCV*, 2022.

[128] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018.

[129] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018.

[130] Yongming Rao, Jiwen Lu, and Jie Zhou. Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In *CVPR*, 2020.

[131] Benjamin Eckart, Wentao Yuan, Chao Liu, and Jan Kautz. Self-supervised learning on 3d point clouds by learning discrete generative models. In *CVPR*, pages 8248–8257, 2021.

[132] Siming Yan, Zhenpei Yang, Haoxiang Li, Li Guan, Hao Kang, Gang Hua, and Qixing Huang. Iae: Implicit autoencoder for point cloud self-supervised representation learning. *arXiv preprint arXiv:2201.00785*, 2022.

[133] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *CVPR*, 2022.

[134] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, 2022.

[135] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training. In *NeurIPS*, 2022.

[136] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[137] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[138] Jason Tyler Rolfe. Discrete variational autoencoders. *arXiv preprint arXiv:1609.02200*, 2016.

[139] Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. Pointgpt: Auto-regressively generative pre-training from point clouds. In *NeurIPS*, 2023.

[140] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.

[141] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

[142] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[143] Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019.

[144] YM Asano, C Rupprecht, and A Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2019.

[145] Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. Prototypical contrastive learning of unsupervised representations. In *ICLR*, 2020.

[146] Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering. In *AAAI*, 2021.

[147] Chen Min, Dawei Zhao, Liang Xiao, Yiming Nie, and Bin Dai. Voxel-mae: Masked autoencoders for pre-training large-scale point clouds. *arXiv preprint arXiv:2206.09900*, 2022.

[148] Fuchen Long, Ting Yao, Zhaofan Qiu, Lusong Li, and Tao Mei. Pointclustering: Unsupervised point cloud pre-training using transformation invariance in clustering. In *CVPR*, 2023.

[149] Yujin Chen, Matthias Nießner, and Angela Dai. 4dcontrast: Contrastive learning with dynamic correspondences for 3d scene understanding. In *ECCV*, 2022.

[150] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019.

[151] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

[152] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *CVPR*, 2017.

[153] Ruijie Quan, Xuanyi Dong, Yu Wu, Linchao Zhu, and Yi Yang. Auto-reid: Searching for a part-aware convnet for person re-identification. In *ICCV*, 2019.

[154] Wenguan Wang, Guolei Sun, and Luc Van Gool. Looking beyond single images for weakly supervised semantic segmentation learning. *IEEE TPAMI*, 2022.

[155] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[156] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[157] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[158] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[159] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *CVPR*, 2022.

[160] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[161] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Mykola Pechenizkiy, Decebal Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*, 2022.

[162] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*, 2017.

[163] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.

[164] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

[165] Shiwei Liu, Decebal Constantin Mocanu, Amarsagar Reddy Ramapuram Matavalam, Yulong Pei, and Mykola Pechenizkiy. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications*, 33:2589–2604, 2021.

[166] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *ICML*, 2020.

[167] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *ICML*, 2019.

[168] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. In *NeurIPS*, 2020.

[169] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In *NeurIPS*, 2021.

[170] Shiwei Liu, Decebal Constantin Mocanu, Yulong Pei, and Mykola Pechenizkiy. Selfish sparse rnn training. In *ICML*, 2021.

[171] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. In *ICLR*, 2016.

[172] Decebal Constantin Mocanu et al. *Network computations in artificial intelligence*. Technische Universiteit Eindhoven, 2017.

[173] Xiaoliang Dai, Hongxu Yin, and Niraj K Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019.

[174] Xiaoliang Dai, Hongxu Yin, and Niraj K Jha. Grow and prune compact, fast, and accurate lstms. *IEEE Transactions on Computers*, 69(3):441–452, 2019.

[175] Md Aamir Raihan and Tor Aamodt. Sparse weight activation training. In *NeurIPS*, 2020.

[176] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85, 2022.

[177] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The dangers of post-hoc interpretability: unjustified counterfactual explanations. In *IJCAI*, 2019.

[178] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[179] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[180] Feng-Lei Fan, Jinjun Xiong, Mengzhou Li, and Ge Wang. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 5(6):741–760, 2021.

[181] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*, 2018.

[182] Jung-Ho Hong, Woo-Jeoung Nam, Kyu-Sung Jeon, and Seong-Whan Lee. Towards better visualizing the decision basis of networks via unfold and conquer attribution guidance. In *AAAI*, 2023.

[183] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. In *NeurIPS*, 2019.

[184] Linchao Zhu and Yi Yang. Label independent memory for semi-supervised few-shot video classification. *IEEE TPAMI*, 44(1):273–285, 2020.

[185] Xiaodong Wang, Zhedong Zheng, Yang He, Fei Yan, Zhiqiang Zeng, and Yi Yang. Progressive local filter pruning for image retrieval acceleration. *IEEE TMM*, 2023.

[186] Cynthia Rudin and Joanna Radin. Why are we using black box models in ai when we don't need to? a lesson from an explainable ai competition. *Harvard Data Science Review*, 1(2): 1–9, 2019.

[187] Zhiyuan Min, Yawei Luo, Wei Yang, Yuesong Wang, and Yi Yang. Entangled view-epipolar information aggregation for generalizable neural radiance fields. *arXiv preprint arXiv:2311.11845*, 2023.

[188] Tianfei Zhou, Wenguan Wang, Ender Konukoglu, and Luc Van Gool. Rethinking semantic segmentation: A prototype view. In *CVPR*, 2022.

[189] James Chenhao Liang, Tianfei Zhou, Dongfang Liu, and Wenguan Wang. Clustseg: Clustering for universal segmentation. In *ICML*, 2023.

[190] Yuesong Wang, Zhaojie Zeng, Tao Guan, Wei Yang, Zhuo Chen, Wenkai Liu, Luoyuan Xu, and Yawei Luo. Adaptive patch deformation for textureless-resilient multi-view stereo. In *CVPR*, 2023.

[191] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, 2018.

[192] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *3DV*, 2017.

[193] Zeyu Hu, Xuyang Bai, Jiaxiang Shang, Runze Zhang, Jiayu Dong, Xin Wang, Guangyuan Sun, Hongbo Fu, and Chiew-Lan Tai. Vmnet: Voxel-mesh network for geodesic-aware 3d semantic segmentation. In *ICCV*, 2021.

[194] Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. Dilated point convolutions: On the receptive field size of point convolutions on 3d point clouds. In *ICRA*, 2020.

[195] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, 2018.

[196] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *ICLR*, 2019.

[197] Philip A Knight. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

[198] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013.

[199] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. Cross-batch memory for embedding learning. In *CVPR*, 2020.

[200] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[201] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop*, pages 1–5, 2015.

[202] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In *AAAI*, 2021.

[203] Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for lidar semantic segmentation. In *CVPR*, 2022.

[204] Shi Qiu, Saeed Anwar, and Nick Barnes. Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In *CVPR*, 2021.

[205] Tao Lu, Limin Wang, and Gangshan Wu. Cga-net: Category guided aggregation for point cloud semantic segmentation. In *CVPR*, 2021.

[206] Liyao Tang, Yibing Zhan, Zhe Chen, Baosheng Yu, and Dacheng Tao. Contrastive boundary learning for point cloud segmentation. In *CVPR*, 2022.

[207] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *CVPR*, 2022.

[208] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *NeurIPS*, 2022.

[209] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[210] Haoran Zhou, Yidan Feng, Mingsheng Fang, Mingqiang Wei, Jing Qin, and Tong Lu. Adaptive graph convolution for point cloud analysis. In *ICCV*, 2021.

[211] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020.

[212] Junbo Yin, Jianbing Shen, Runnan Chen, Wei Li, Ruigang Yang, Pascal Frossard, and Wenguan Wang. Is-fusion: Instance-scene collaborative fusion for multimodal 3d object detection. In *CVPR*, 2024.

[213] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020.

[214] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021.

[215] Han Cai, Chuang Gan, Ji Lin, and Song Han. Network augmentation for tiny deep learning. In *ICLR*, 2022.

[216] Vidya Kamath and A Renuka. Deep learning based object detection for resource constrained devices-systematic review, future trends and challenges ahead. *Neurocomputing*, 2023.

[217] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *CVPR*, 2023.

[218] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[219] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In *ICLR*, 2022.

[220] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[221] Graham H Diering, Raja S Nirujogi, Richard H Roth, Paul F Worley, Akhilesh Pandey, and Richard L Huganir. Homer1a drives homeostatic scaling-down of excitatory synapses during sleep. *Science*, 355(6324):511–515, 2017.

[222] Luisa De Vivo, Michele Bellesi, William Marshall, Eric A Bushong, Mark H Ellisman, Giulio Tononi, and Chiara Cirelli. Ultrastructural evidence for synaptic scaling across the wake/sleep cycle. *Science*, 355(6324):507–510, 2017.

[223] Qinghao Meng, Wenguan Wang, Tianfei Zhou, Jianbing Shen, Luc Van Gool, and Dengxin Dai. Weakly supervised 3d object detection from lidar point cloud. In *ECCV*, 2020.

[224] Radu Alexandru Rosu, Peer Schütt, Jan Quenzel, and Sven Behnke. Latticenet: Fast point cloud segmentation using permutohedral lattices. *arXiv preprint arXiv:1912.05905*, 2019.

[225] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *ECCV*, 2018.

[226] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. In *IROS*, 2019.

[227] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019.

[228] Hung-Yueh Chiang, Yen-Liang Lin, Yueh-Cheng Liu, and Winston H Hsu. A unified point-based framework for 3d segmentation. In *3DV*, 2019.

[229] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, 2020.

[230] Huan Lei, Naveed Akhtar, and Ajmal Mian. Seggcn: Efficient 3d point cloud segmentation with fuzzy spherical kernel. In *CVPR*, 2020.

[231] Zeyu Hu, Mingmin Zhen, Xuyang Bai, Hongbo Fu, and Chiew-lan Tai. Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds. In *ECCV*, 2020.

[232] Venice Erin Liong, Thi Ngoc Tho Nguyen, Sergi Widjaja, Dhananjai Sharma, and Zhuang Jie Chong. Amvnet: Assertion-based multi-view fusion network for lidar semantic segmentation. *arXiv preprint arXiv:2012.04934*, 2020.

[233] Kyle Genova, Xiaoqi Yin, Abhijit Kundu, Caroline Pantofaru, Forrester Cole, Avneesh Sud, Brian Brewington, Brian Shucker, and Thomas Funkhouser. Learning 3d semantic segmentation with only 2d image supervision. In *3DV*, 2021.

[234] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019.

[235] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE TPAMI*, 43(8):2647–2664, 2020.

[236] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *NeurIPS*, 2016.

[237] Qinghao Meng, Wenguan Wang, Tianfei Zhou, Jianbing Shen, Yunde Jia, and Luc Van Gool. Towards a weakly supervised framework for 3d point cloud object detection and annotation. *IEEE TPAMI*, 44(8):4454–4468, 2021.

[238] Zhedong Zheng, Tao Ruan, Yunchao Wei, Yi Yang, and Tao Mei. Vehiclenet: Learning robust visual representation for vehicle re-identification. *IEEE TMM*, 23:2683–2693, 2020.

[239] Wenguan Wang, Cheng Han, Tianfei Zhou, and Dongfang Liu. Visual recognition with deep nearest centroids. In *ICLR*, 2022.

[240] Plamen Angelov and Eduardo Soares. Towards explainable deep neural networks (xdnn). *Neural Networks*, 130:185–194, 2020.

[241] Qinglong Zhang, Lu Rao, and Yubin Yang. A novel visual interpretability for deep neural networks by optimizing activation maps with perturbation. In *AAAI*, 2021.

[242] Naveed Akhtar and Mohammad Amir Asim Khan Jalwana. Rethinking interpretation: Input-agnostic saliency mapping of deep visual classifiers. In *AAAI*, 2023.

[243] Brian Hu, Paul Tunison, Brandon RichardWebster, and Anthony Hoogs. Xaitk-saliency: An open source explainable ai toolkit for saliency. In *AAAI*, 2023.

[244] Anh Viet Phan, Minh Le Nguyen, Yen Lam Hoang Nguyen, and Lam Thu Bui. Dgcnn: A convolutional neural network over large-scale labeled graphs. *Neural Networks*, 108: 533–543, 2018.

[245] Xu Yan. Pointnet/pointnet++ pytorch. `https://github.com/yanx27/Pointnet_Pointnet2_pytorch`, 2019.

[246] David Nova and Pablo A Estévez. A review of learning vector quantization classifiers. *Neural Computing and Applications*, 25(3):511–524, 2014.

[247] Michael Biehl, Barbara Hammer, and Thomas Villmann. Prototype-based models in machine learning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 7(2):92–111, 2016.

[248] Eleanor Rosch. Cognitive representations of semantic categories. *Journal of experimental psychology: General*, 104(3):192, 1975.

[249] Barbara J Knowlton and Larry R Squire. The learning of categories: Parallel brain systems for item memory and category knowledge. *Science*, 262(5140):1747–1749, 1993.

[250] John R Taylor. *Linguistic categorization*. OUP Oxford, 2003.

[251] Tianyi Lin, Nhat Ho, and Michael Jordan. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *ICML*, 2019.

[252] Teuvo Kohonen. Improved versions of learning vector quantization. In *IJCNN*, 1990.

[253] Teuvo Kohonen. *Self-organizing maps*, volume 30. Springer Science & Business Media, 2012.

[254] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[255] Helge Ritter, Thomas Martinetz, Klaus Schulten, et al. *Neural computation and self-organizing maps: an introduction*. Addison-Wesley Reading, MA, 1992.

[256] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *3DV*, 2016.

[257] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *NeurIPS*, 2016.

[258] Christoph Molnar. Interpretable machine learning. `https://christophm.github.io/interpretable-ml-book/`, 2020.

[259] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. In *preprint*. OpenAI, 2018.

[260] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[261] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[262] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

[263] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[264] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[265] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

[266] Runpei Dong, Zekun Qi, Linfeng Zhang, Junbo Zhang, Jianjian Sun, Zheng Ge, Li Yi, and Kaisheng Ma. Autoencoders as cross-modal teachers: Can pretrained 2d image transformers help 3d representation learning? *arXiv preprint arXiv:2212.08320*, 2022.

[267] GitHub. https://github.com/.

[268] Thingiverse. https://www.thingiverse.com/.

[269] Sketchfab. https://sketchfab.com/.

[270] Polycam. https://poly.cam/.

[271] Smithsonian 3D Digitization. https://3d.si.edu//.

[272] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. https://github.com/threestudio-project/threestudio, 2023.

[273] Chao-Yuan Wu, Justin Johnson, Jitendra Malik, Christoph Feichtenhofer, and Georgia Gkioxari. Multiview compressive coding for 3D reconstruction. *arXiv:2301.08247*, 2023.

[274] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *ECCV*, 2022.

[275] Junbo Yin, Jianbing Shen, Runnan Chen, Wei Li, Ruigang Yang, Pascal Frossard, and Wenguan Wang. Is-fusion: Instance-scene collaborative fusion for multimodal 3d object detection. In *CVPR*, 2024.

[276] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *ICCV*, 2019.

[277] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *AAAI*, 2019.

[278] Kosmann-Schwarzbach Yvette. The noether theorems. invariance and conservation laws in the twentieth century. *Translated by Bertram E. Schwarzbach. NY: Springer*, 2011.

[279] Xiangwen Kong and Xiangyu Zhang. Understanding masked image modeling via learning occlusion invariant feature. In *CVPR*, 2023.

[280] Rumen Dangovski, Li Jing, Charlotte Loh, Seungwook Han, Akash Srivastava, Brian Cheung, Pulkit Agrawal, and Marin Soljacic. Equivariant self-supervised learning: Encouraging equivariance in representations. In *ICLR*, 2021.

[281] Wenguan Wang, Tianfei Zhou, Fisher Yu, Jifeng Dai, Ender Konukoglu, and Luc Van Gool. Exploring cross-image pixel contrast for semantic segmentation. In *ICCV*, 2021.

[282] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[283] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D Wegner, Konrad Schindler, and Marc Pollefeys. Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847*, 2017.

[284] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[285] I Loshchilov and F Hutter. Stochastic gradient descent with warm restarts. In *ICLR*, 2016.

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

[287] Xiao Zheng, Xiaoshui Huang, Guofeng Mei, Yuenan Hou, Zhaoyang Lyu, Bo Dai, Wanli Ouyang, and Yongshun Gong. Point cloud pre-training with diffusion models. *arXiv preprint arXiv:2311.14960*, 2023.

[288] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *ICCV*, 2021.

[289] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *ICCV*, 2021.

[290] Ziyi Wang, Xumin Yu, Yongming Rao, Jie Zhou, and Jiwen Lu. Take-a-photo: 3d-to-2d generative pre-training of point cloud models. In *ICCV*, 2023.

[291] Iñigo Alonso, Luis Riazuelo, Luis Montesano, and Ana C Murillo. 3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation. In *IROS*, 2020.