Robust and Generalizable Deep Geometric Representation Learning for 3D Point Clouds

by

Ali BAHRI

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, MARCH 14, 2026

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Christian Desrosiers, Thesis supervisor
Department of Software Engineering and IT, at École de Technologie Supérieure

Mr. Ismail Ben Ayed, Thesis Co-Supervisor
Department of Systems Engineering, at École de Technologie Supérieure

Mr. Mohamad Forouzanfar, Chair, Board of Examiners
Department of Systems Engineering, at École de Technologie Supérieure

Mr. Sheldon Andrews, Member of the Jury
Department of Software Engineering and IT, at École de Technologie Supérieure

Mr. Yang Wang, External Independent Examiner
Department of Computer Science and Software Engineering, at Concordia University

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON MARCH 10, 2026

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

The completion of this PhD represents far more than an academic achievement — it marks the end of a long, demanding, and deeply meaningful chapter of my life. This journey has been shaped by moments of discovery, uncertainty, perseverance, and growth, and it would not have been possible without the support and presence of many people to whom I am sincerely grateful.

I would first like to express my profound appreciation to my supervisors, Professors Christian Desrosiers and Ismail Ben Ayed. Their guidance, insight, and steady encouragement have shaped my development as a researcher and have influenced the way I think, question, and approach scientific problems. Their mentorship has been invaluable, and I am truly grateful for the trust and freedom they gave me throughout this work.

I would also like to extend my sincere thanks to Arnaud Lina, Steve Massicotte, and Siavosh Shadpay, my collaborators at Zebra Technologies. Their constant support, openness, and technical expertise greatly enriched my research experience. Working with them has been both inspiring and enjoyable, and I am thankful for the opportunities and challenges we shared during our projects together.

My gratitude also goes to my thesis committee members, Professor Sheldon Andrews and Professor Mohamad Forouzanfar, for generously dedicating their time to evaluate my work and for providing thoughtful and constructive feedback that helped strengthen this thesis.

A special and heartfelt acknowledgement is reserved for my family — especially my mother and father. Their love, patience, and unwavering support have been the foundation of every step I have taken. Being away from them for the past four years has been one of the most difficult parts of this journey, yet their encouragement has never diminished. Their sacrifices and belief in me have been a constant source of motivation, and this accomplishment is deeply shared with them.

Finally, I wish to thank the remarkable friends I have met in Montréal. To Moslem Yazdanpanah, for our collaborations, discussions, and the many moments that made this experience richer. And to my friends and colleagues Mehrdad, Milad, Gustavo, David, Farzad, and Samuel —

thank you for the support, the conversations, and the memories that brought balance and joy beyond the research environment. Your friendship has made this journey truly meaningful.

Apprentissage profond géométrique robuste et généralisable pour les représentations 3D de nuages de points

Ali BAHRI

RÉSUMÉ

L'apprentissage profond a révolutionné la perception tridimensionnelle, permettant la reconnaissance, la segmentation et la reconstruction précises des nuages de points dans des domaines tels que la robotique, la navigation autonome et la réalité augmentée. Cependant, deux défis majeurs persistent. Premièrement, les modèles 3D à grande échelle dépendent fortement de jeux de données annotés, dont la création est coûteuse et chronophage. Cela motive le développement de méthodes d'apprentissage auto-supervisé capables de préentraîner des modèles sur des données non annotées et d'apprendre des représentations géométriques transférables pour des tâches en aval disposant de peu d'annotations. Deuxièmement, les modèles entraînés dans des conditions fixes échouent souvent à se généraliser lorsqu'ils sont exposés à des décalages de distribution réels causés par le bruit, les variations des capteurs ou les changements d'environnement. Ce problème motive la conception de cadres d'apprentissage à la fois *robustes* face aux décalages de distribution et *adaptatifs* à de nouveaux environnements, sans nécessiter de données annotées.

Cette thèse fait progresser la robustesse et la généralisabilité de l'apprentissage profond 3D à travers une exploration unifiée de l'*apprentissage de représentations auto-supervisé* et de l'*apprentissage en phase de test (TTL)*. La première partie étudie comment construire des *priors* géométriques permettant aux modèles d'apprendre des représentations 3D significatives et transférables. (1) GeoMask3D introduit une stratégie de modélisation masquée sensible à la géométrie, alignant explicitement le préentraînement masqué sur les indices structurels des formes 3D, améliorant ainsi l'interprétabilité et l'invariance des caractéristiques apprises. (2) Spectral-Informed Mamba étend les modèles d'espace d'état aux nuages de points en exploitant le spectre laplacien des graphes de variétés, définissant un ordre de parcours invariant à l'isométrie qui améliore la robustesse aux changements de point de vue et la précision de la segmentation, tout en atteignant une complexité computationnelle linéaire par rapport aux conceptions quadratiques des Transformers.

La seconde partie traite de l'adaptation à des conditions de test inédites, en proposant des méthodes efficaces et fiables d'*entraînement en phase de test (TTT)* et d'*adaptation en phase de test (TTA)*. (3) Sampling-Variation Weight Averaging (SVWA) présente la première stratégie TTA complète pour les nuages de points, combinant la variation d'échantillonnage et la moyenne des poids afin d'obtenir une adaptation robuste grâce à une optimisation autour de minima plats. (4) SMART-PC introduit un cadre TTT basé sur le squelette qui apprend des abstractions géométriques compactes durant le préentraînement, permettant une adaptation en temps réel sans rétropropagation en mettant à jour uniquement les statistiques de BatchNorm.

Ensemble, ces contributions établissent un cadre cohérent pour un apprentissage profond géométrique 3D à la fois robuste et généralisable. En unifiant le préentraînement auto-supervisé et l'apprentissage efficace en phase de test, cette thèse fait progresser les systèmes de perception

3D vers des performances stables, adaptatives et résilientes face aux décalages de distribution réels, tout en préservant l'interprétabilité et l'efficacité computationnelle.

Mots-clés: Apprentissage auto-supervisé, Apprentissage de représentations 3D, Adaptation en phase de test, Entraînement en phase de test

Robust and Generalizable Deep Geometric Representation Learning for 3D Point Clouds

Ali BAHRI

ABSTRACT

Deep learning has revolutionized 3D perception, enabling accurate recognition, segmentation, and reconstruction of point-cloud data across robotics, autonomous navigation, and augmented reality. However, two major challenges remain. First, large-scale 3D models rely heavily on labeled datasets, which are costly and time-consuming to acquire. This motivates the development of self-supervised learning methods that can pretrain models on unlabeled data and learn transferable geometric representations for downstream tasks with limited annotations. Second, models trained under fixed conditions often fail to generalize when exposed to real-world distribution shifts caused by noise, sensor variation, or environmental changes. This challenge motivates the development of learning frameworks that are both *robust* to distribution shifts and *adaptive* to new environments without requiring labeled data.

This thesis advances the robustness and generalizability of 3D deep learning through a unified exploration of *self-supervised representation learning* and *test-time learning (TTL)*. The first part investigates how to construct geometric priors that enable models to learn meaningful and transferable 3D representations. (1) GeoMask3D introduces a geometry-aware masked modeling strategy that explicitly aligns masked pretraining with structural cues of 3D shapes, improving the interpretability and invariance of learned features. 2) Spectral-Informed Mamba extends state-space models to point clouds by leveraging the Laplacian spectrum of the underlying graph manifold, producing an isometry-invariant traversal order that strengthens the quality of self-supervised geometric representations while maintaining linear computational complexity compared to quadratic Transformer designs.

The second part addresses adaptation under unseen test conditions, proposing efficient and reliable methods for *test-time training (TTT)* and *test-time adaptation (TTA)*. (3) Sampling-Variation Weight Averaging (SVWA) presents the first fully TTA strategy for point clouds, combining sampling variation and weight averaging to achieve robust adaptation through flat-minima optimization. (4) SMART-PC introduces a skeleton-based TTT framework that learns compact geometric abstractions during pretraining, enabling real-time adaptation without backpropagation by updating only BatchNorm statistics.

Together, these contributions establish a cohesive framework for robust and generalizable deep geometric representation learning in 3D. By unifying self-supervised pretraining with efficient test-time learning, this thesis advances toward 3D perception systems that are stable, adaptive, and resilient to real-world distribution shifts while preserving interpretability and computational efficiency.

Keywords: Self-Supervised Learning, 3D Representation Learning, Test-Time Adaptation, Test-Time Training

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS AND ACRONYMS

TTA	Test-Time Adaptation
TTT	Test-Time Training
MAE	Masked Auto Encoders
MIM	Masked Image Modeling
CNN	Convolution Neural Network
GNN	Graph Neural Network
SOTA	State-Of-The-Art
GM3D	GeoMask3D
LAME	Laplacian Adjusted Maximum-likelihood Estimation
FPS	Farthest Point Sampling
KNN	K-Nearest Neighbours
IP	Informative Patches
DRC	Dense Relation Comparison
MSE	Mean Square Error
SVM	Support Vector Machine
GC	Geometric Complexity
SSMs	State Space Models
NLP	Natural Language Processing
AE	Autoencoder

S4	Structured State Space Sequence Model
SSR	Spectral Spatial Reordering
SWAD	Stochastic Weight Averaging for Domain generalization
VR	Virtual Reality
AR	Augmented Reality
CSM	Cross-Scan Module
SAST	Surface-Aware Spectral Traversing
HLT	Hierarchical Local Traversing
SST	Spectral Spatial Traversing
WA	Weight Averaging
LN	Layer Normalization
BN	Batch Normalization

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

$\mathcal{P} = \{p_i\}_{i=1}^{N}$	3D point cloud as a finite set of points $p_i \in \mathbb{R}^3$ (underlying representation in all contributions).
c_i	Farthest Point Sampling (FPS) center selected from $\mathcal{P}$ to form a point-cloud token.
$\mathcal{G}_i$	Local patch (group) around center c_i, constructed by K-Nearest Neighbors (KNN), $\mathcal{G}_i \in \mathbb{R}^{K\times 3}$.
$\text{Chamfer}(S, S')$	Chamfer distance between two point sets S and S', used as the reconstruction metric in masked autoencoders.
$\mathcal{L}^{\text{rec}_p}$	Patch-wise reconstruction loss for masked point-cloud patches, defined via the Chamfer distance.
$\text{GC} \in \mathbb{R}^N$	Geometric Complexity scores predicted for each patch in GeoMask3D, measuring the relative difficulty of reconstructing patches.
$\mathcal{L}_{\text{GC}}$	Loss function used to train the GC prediction head in GeoMask3D by enforcing consistency between GC scores and reconstruction difficulty.
L_{rw}	Random-walk graph Laplacian used for spectral traversal on the patch graph.
$\mathbf{v}^{(k)}$	k-th non-constant eigenvector of L_{rw}, used to define smooth traversal orders over point-cloud patches.
s	Number of smallest non-constant Laplacian eigenvectors retained to build the spectral traversal.
N_V	Number of sampling variations (independent resamplings of the input point cloud) used in Sampling-Variation Weight Averaging (SVWA) during test-time adaptation.

$P_{\text{local}} = \text{kNN}(C, P)$	Tensor of local neighborhoods $P_{\text{local}} \in \mathbb{R}^{M \times K \times 3}$ built around FPS centers for skeletal prediction.
$s = (c_s, r(c_s))$	Skeletal sphere in SMART-PC, defined by its center $c_s \in \mathbb{R}^3$ and radius $r(c_s) \in \mathbb{R}$.
S	Set of skeletal spheres composing the skeletal mesh that summarizes the 3D structure of the point cloud.
$\mathcal{L}_{\text{p2s}}$	Point-to-sphere loss that enforces alignment between input points and their closest skeletal spheres.
$\mathcal{L}_{\text{sampling}}$	Sampling loss based on the Chamfer distance between input points and points sampled on skeletal spheres, reducing high-frequency noise.
$\mathcal{L}_{\text{radius}}$	Radius regularization loss encouraging larger and more stable skeletal radii to improve robustness under corruption.

INTRODUCTION

Deep learning has transformed 3D perception, establishing point-cloud analysis as a foundation for applications in autonomous driving, augmented and virtual reality, and robotics. Despite remarkable progress achieved through supervised models such as pointnet (Qi, Su, Mo & Guibas, 2017a), DGCNN (Phan, Le Nguyen, Nguyen & Bui, 2018), and Transformer-based encoders, large-scale 3D learning continues to face a fundamental bottleneck: the scarcity and cost of labeled data. While progress has been made in semi-automatic and synthetic 3D annotation, creating large-scale labeled datasets still remains far more demanding than in 2D. Point-level segmentation, correspondence, and registration typically require specialized tools and substantial expert supervision, making large 3D datasets costly and time-intensive to curate. This high labeling burden limits the scalability and generalization of purely supervised 3D networks.

This limitation has spurred the rapid development of *self-supervised representation learning (SSL)* as an alternative paradigm for 3D perception. Self-supervised methods aim to learn meaningful and transferable representations from unlabeled data by defining auxiliary pretext tasks that exploit the intrinsic structure of 3D geometry. Instead of relying on human supervision, these tasks encourage the network to capture spatial organization, local surface characteristics, and global topology directly from raw point clouds. Once pretrained, the encoder can be fine-tuned for downstream tasks such as classification, segmentation, or reconstruction using only a small fraction of labeled data. This approach not only reduces annotation cost but also encourages the learning of geometric representations that are more stable across variations in sampling density and viewpoint.

Among self-supervised frameworks, masked representation learning has emerged as one of the most influential approaches. Inspired by the success of Masked Autoencoders (MAEs) in vision and language modeling, 3D masked modeling pretrains a network by reconstructing masked regions of the point cloud from their visible context. The principle is straightforward: by

recovering the missing geometry, the encoder is forced to learn meaningful spatial correlations between observed and unobserved regions. In all transformer-based methods, each patch is represented by its patch center, which serves as a positional embedding for both the encoder and decoder. This center-based positional encoding provides an approximately translation-invariant reference frame for point patches and allows the model to reason about local geometry without relying on a fixed grid. However, early 3D MAE approaches adopted random masking strategies, where the masked patches were selected arbitrarily, without considering their geometric importance. While this random selection ensures coverage, it overlooks which regions of the point cloud contribute most to structural understanding. Smooth or flat regions, for example, provide little geometric information compared to canonical or high-curvature areas that define the shape's structure. As a result, purely random masking may cause the model to overemphasize trivial regions, limiting the richness and discriminative power of learned features.

Beyond the design of pretext tasks, another major limitation lies in the computational efficiency of existing architectures. Transformer-based networks, which dominate current 3D SSL approaches, scale quadratically with the number of tokens, making them computationally expensive for dense or large-scale point clouds. Their reliance on global self-attention also introduces redundancy, as all token pairs are treated equally regardless of spatial relevance. This inefficiency becomes particularly critical in 3D domains, where point sets can contain tens or even hundreds of thousands of points. Recently, state-space models with linear computational complexity, originally developed for sequence modeling in text, have emerged as direction-sensitive alternatives to Transformers and have shown promising results in 2D vision tasks. However, extending these models to 3D point clouds remains challenging due to the irregular, unordered, and sparse nature of 3D data, which lacks a natural sequential structure or canonical ordering. This absence of inherent spatial sequence makes it difficult to define meaningful traversal paths or neighborhood relationships required by state-space formulations.

While self-supervised representation learning alleviates the dependency on large annotated datasets, another critical challenge persists in 3D perception: ensuring model robustness when exposed to data that differs from the training distribution. In real-world deployment, point clouds are often subject to unpredictable variations caused by changes in sensor type, scanning distance, sampling density, illumination, occlusion, or environmental conditions such as fog and dust. These factors introduce *distribution shifts* between the training (source) and testing (target) domains. Even small discrepancies in input statistics can cause substantial degradation in performance, as deep networks tend to memorize domain-specific patterns rather than learn representations that are truly invariant to such changes. This sensitivity undermines the reliability of 3D models in safety-critical applications, where consistent behavior across varying conditions is essential.

Traditional approaches to mitigate distribution shift, such as domain adaptation (DA), assume access to both labeled source data and unlabeled or partially labeled target data during training. These methods aim to align the source and target distributions through adversarial learning, feature alignment, or discrepancy minimization before deployment. However, in many practical scenarios, target data are unavailable prior to deployment, and models must operate in unseen environments where collecting or labeling new samples is infeasible. Furthermore, privacy restrictions, limited computational resources, and real-time processing requirements make it impractical to retrain or fine-tune large models for each new domain. These constraints underscore the need for test-time learning frameworks that can adapt online and autonomously to unseen distributions without requiring access to target data during training.

This has led to the emergence of *test-time learning (TTL)*, a paradigm that allows models to adapt on-the-fly to unseen domains using only unlabeled test data. TTL encompasses two complementary directions: *Test-Time Training (TTT)* and *TTA*. Both seek to mitigate performance degradation under distribution shift but differ in their requirements and adaptation strategies.

TTT assumes that, during pretraining, an auxiliary self-supervised objective—such as rotation prediction, reconstruction, or feature alignment—has been integrated alongside the main task. At test time, the model leverages this auxiliary loss as a proxy signal to update its parameters without requiring labels. TTA, on the other hand, performs adaptation directly during inference by optimizing unsupervised objectives derived from the model's predictions, such as entropy minimization, pseudo-label consistency, or confidence calibration. Unlike TTT, TTA operates entirely without access to source data or auxiliary branches. It adapts the pretrained model solely based on the statistical properties of unlabeled test samples encountered during inference.

Despite their promise, both TTT and TTA face fundamental challenges when extended to 3D point-cloud data. The inherent irregularity and geometric variability of 3D structures make it difficult to define stable and generalizable adaptation objectives that remain consistent across different shapes, sampling densities, and corruption types. Moreover, most existing approaches depend on iterative gradient-based updates during inference, which are computationally demanding and impractical for real-time systems such as robotics and autonomous navigation.

0.1 Research Statement

The primary objective of this thesis is to construct a unified lifecycle for 3D deep learning that is both data-efficient in training and robust in deployment. We posit that achieving this requires addressing four fundamental limitations in current methodologies, ranging from how representations are learned offline to how models adapt online.

1. Integrating Geometry into Self-Supervised Masking: Current masked representation learning treats all point patches as equally informative, often employing random masking strategies. We argue that this approach ignores the intrinsic topological hierarchy of 3D shapes. The first objective of this thesis is to determine how geometric complexity can guide the masking process to ensure models learn features that are structurally significant rather than trivially smooth.

2. Reconciling Computational Efficiency with Geometric Fidelity: There exists a dichotomy in 3D backbone design: Transformers offer high modeling power but suffer from quadratic complexity, while efficient linear models (such as state-space models) struggle to process the unordered nature of point clouds. The second objective is to investigate how spectral graph theory can be leveraged to define a canonical, isometry-invariant traversal order, thereby enabling linear-complexity models to capture complex 3D structures effectively.

3. Mitigating Sampling Instability through Weight Averaging: Real-world point clouds are subject to significant variations in sampling and noise. In the context of TTA, the final objective is to turn this randomness into an advantage. We investigate a strategy where we adapt the model on multiple variations of the same object and then average the model weights. We hypothesize that this averaging process leads the model to a stable "flat minimum," making it robust to noise and sampling differences without needing labeled data.

4. Enabling Adaptation via Geometric Abstraction: Models trained on static data often fail when the testing environment changes (distribution shift). We hypothesize that while fine surface details may change, the high-level structure—specifically the object's skeleton—remains stable. The third objective is to develop a TTT framework that utilizes these stable geometric priors to guide self-supervised adaptation, allowing models to adjust to new environments without requiring backpropagation.

0.2 Contributions

This thesis addresses three overarching challenges in 3D point-cloud perception—learning meaningful geometric representations, defining effective traversal strategies for efficient state-space models, and improving robustness under distribution shifts—through four complementary contributions. Two contributions focus on advancing self-supervised representation learning, while the other two introduce novel strategies for test-time learning. Together, these works

enable deep 3D models to learn effectively from limited annotations and adapt reliably to diverse real-world conditions.

In all papers for which I am the first author, I made the primary contributions to the work, including the formulation of the main research ideas, development of the proposed methods, implementation, experimental evaluation, and preparation of the manuscript. The second author contributed to discussions related to the research ideas and assisted in preparing some of the visual figures and the manuscript. The remaining co-authors, who are members of our research group, provided comments and suggestions on the writing, presentation, and visualization of the work, as well as general feedback during the research process. My supervisor and co-supervisor guided the overall direction of the research, supervised the development of the ideas, and provided continuous feedback throughout the preparation of the papers.

- **Chapter 2**: This chapter introduces *GeoMask3D*, a geometry-informed masking strategy for Self-Supervised Learning (SSL) in point clouds. Unlike prior Masked Autoencoder (MAE) frameworks that rely on random patch selection, GeoMask3D prioritizes informative regions based on geometric complexity, enabling the model to focus on structurally significant areas while ignoring less informative smooth surfaces. The method incorporates a feature-level knowledge distillation mechanism that transfers geometric cues from a pretrained teacher network to the student, improving the stability and expressiveness of learned representations. Integrated into the pretraining process of existing MAE-based architectures, GeoMask3D achieves consistent gains across multiple downstream tasks such as classification, part segmentation, and reconstruction, demonstrating its ability to capture richer and more discriminative geometric features.

  Related publication:

  - GeoMask3D: Geometrically Informed Mask Selection for Self-Supervised Point Cloud Learning in 3D (Bahri *et al.*, 2024b), published in *Transactions on Machine Learning Research (TMLR)*, 2024.

- **Chapter 3**: The second contribution introduces Spectral-Informed Mamba, a novel methodology that integrates state-space sequence models with masked autoencoders for efficient and geometry-aware 3D point-cloud representation learning. While state-space models (SSMs) such as Mamba have shown remarkable performance in natural language processing and 2D vision, their extension to 3D data remains non-trivial due to the irregular and unordered nature of point clouds. Spectral-Informed Mamba addresses this challenge by leveraging the spectrum of a graph Laplacian to capture patch connectivity, defining an isometry-invariant traversal that preserves geometric continuity and robustness to viewpoint changes. In addition, a hierarchical local traversal strategy recursively partitions spectral components to enhance segmentation and local feature reasoning. Finally, a token-alignment strategy restores masked tokens to their original spatial order within the sequence, ensuring consistency between geometric structure and sequence dynamics. Together, these spectral traversal, hierarchical segmentation, and token-alignment mechanisms allow Spectral-Informed Mamba to achieve linear computational complexity while maintaining geometric fidelity, substantially improving classification, segmentation, and few-shot performance across standard 3D benchmarks.

  Related publication:

  - Spectral Informed Mamba for Robust Point Cloud Processing (Bahri *et al.*, 2025b), published in *Computer Vision and Pattern Recognition (CVPR)*, 2025.

- **Chapter 4**: This chapter introduces *Sampling-Variation Weight Averaging (SVWA)*, a novel TTA strategy for 3D point cloud classification that combines the complementary strengths of sampling diversity and weight averaging. Unlike prior TTA methods that rely on single test samples, SVWA exploits the inherent stochasticity of 3D sampling by generating multiple geometric variations of each test instance through FPS and KNN. For each variation, the model is adapted using an entropy-minimization objective based on the TENT algorithm, updating only normalization parameters without requiring any source data or labels. The

adapted models from all variations are then merged via weight averaging, which naturally converges toward flatter minima in the loss landscape—yielding a model that is more stable and generalizable under distribution shifts. This strategy effectively reduces sensitivity to sampling noise and improves robustness to real-world corruptions. Extensive experiments demonstrate that SVWA consistently outperforms existing adaptation methods such as TENT and MATE while maintaining minimal computational overhead.

Related publication:

- Test-Time Adaptation in Point Clouds: Leveraging Sampling Variation with Weight Averaging (Bahri *et al.*, 2025d), published in *Winter Conference on Applications of Computer Vision (WACV)- Oral Presentation*, 2025.

- **Chapter 5**: This chapter introduces *SMART-PC (Skeletal Model Adaptation for Robust Test-Time Training)*, a skeleton-based strategy designed to improve 3D model robustness under distribution shift. SMART-PC integrates skeletal prediction and classification within a unified framework, where the network learns to infer skeletal points and their radii as geometric abstractions of 3D objects. These skeletal representations encode the intrinsic topology and structural continuity of shapes, providing stable and noise-resistant features that generalize well across domains. During pretraining, the model jointly optimizes a skeletal reconstruction loss—composed of point-to-sphere, sampling, and radius-regularization terms—and a classification loss, ensuring that the encoder captures both global semantics and geometric abstraction. At test time, SMART-PC eliminates gradient-based optimization entirely. Instead, adaptation is achieved by updating only the BatchNorm statistics using incoming test samples, guided by the learned skeletal priors. This backpropagation-free strategy enables efficient real-time adaptation. Experiments confirm that SMART-PC achieves SOTA robustness and efficiency, surpassing previous TTT frameworks such as MATE in both performance and computational cost.

Related publication:

- SMART-PC: Skeletal Model Adaptation for Robust Test-Time Training in Point Clouds (Bahri *et al.*, 2025a), published in *International Conference on Machine Learning (ICML)*, 2025.

This thesis advances the field of 3D representation learning by addressing key challenges in geometric understanding, efficient token traversal for state-space modeling, and robustness to distribution shifts. In the SSL domain, it introduces geometry-informed masking and spectral traversal strategies that enhance structural awareness and improve the scalability of point-cloud pretraining. In the test-time learning domain, it proposes novel strategies for both TTT and adaptation, enabling models to dynamically adjust to unseen environments without labeled supervision. Collectively, these works contribute to the development of data-efficient, geometry-aware, and resilient 3D perception models for real-world deployment.

0.2.1 Additional Contributions

In addition to the main works presented in this thesis, one complementary study was conducted that further contributes to the broader understanding of model robustness under distribution shifts.

- We introduce TRUST, a test-time adaptation method that exploits the unique traversal structure of State Space Models to enhance robustness under distribution shifts. By generating multiple directional traverses of the input, TRUST obtains diverse causal views and uses their predictions as pseudo-labels to refine Mamba-specific parameters. The adapted weights from each traversal are then averaged, yielding a more stable and generalizable model. This approach provides the first traversal-aware TTA strategy for SSMs and achieves strong improvements across multiple robustness benchmarks.

Related publication:

- TRUST: Test-Time Refinement using Uncertainty-Guided SSM Traverses (Dastani *et al.*, 2025), published in *Neural Information Processing Systems (NeurIPS)*, 2025.

CHAPTER 1

LITERATURE REVIEW

1.1 Scope and positioning

Research on 3D point-cloud perception has expanded rapidly in recent years, driven by applications in autonomous driving, robotics, and mixed reality. Within this landscape, three lines of work are particularly relevant to this thesis: (i) self-supervised representation learning for 3D point clouds, which seeks to reduce dependence on large annotated datasets; (ii) the challenge of defining meaningful token traversal strategies that enable state-space models to achieve linear scalability while preserving geometric adjacency in unordered point clouds; and (iii) test-time learning strategies that aim to improve robustness when models are deployed under distribution shift. Although these research directions are typically examined independently, they are inherently interconnected. The quality of self-supervised representations shapes the model's capacity for reliable fine-tuning, as stronger geometric features provide a more stable foundation for downstream updates; meanwhile, the design of a meaningful traversal for state-space models is essential for reducing computational complexity and for preserving geometric continuity in point clouds, since these models are direction-sensitive and require an ordering that reflects spatial adjacency; and test-time learning determines how the model behaves once deployed in shifting environments. Together, these factors jointly shape the overall reliability and robustness of 3D perception systems in real-world conditions.

This chapter reviews the literature along these three axes, with a particular emphasis on methods and ideas that form the conceptual foundation of the contributions developed in later chapters. We first survey SSL for 3D point clouds, covering masked representation learning, and the design of pretext tasks that exploit geometric structure. Special attention is given to masked autoencoder (MAE)-style methods and their masking strategies, as they directly relate to the problem of how to expose informative regions of a shape during pretraining. We then examine recent state-space models for 2D and 3D vision, which offer linear computational complexity. In the 3D setting, we outline the challenges these models face—particularly the absence of a

natural ordering in point clouds—and discuss how tokenization, patchification, and traversal strategies influence their ability to preserve geometric structure. The final part of the chapter focuses on robustness under distribution shift and the emerging family of *test-time learning* methods, including TTT and TTA. We review normalization-based adaptation, entropy- and consistency-driven objectives, and recent work on source-free adaptation in 3D.

This review sets the stage for the four research directions developed in the remainder of the thesis. Chapters 2 and 3 focus on advancing SSL for 3D point clouds: Chapter 2 introduces a geometry-aware masking strategy that improves the structural fidelity of pretrained representations, while Chapter 3 develops a spectral traversal approach that enables efficient state-space modeling despite the lack of natural ordering in point clouds. Chapters 4 and 5 shift the focus to robustness at deployment. Chapter 4 presents a sampling-variation and weight-averaging method for TTA under distribution shift, and Chapter 5 proposes a skeletal-based TTT strategy that adapts models without backpropagation. Together, these chapters illustrate how improved pretraining, efficient architectural design, and adaptive inference contribute complementary solutions to the broader challenge of reliable 3D perception in real-world environments.

1.2 Fundamentals of 3D Point Clouds

1.2.1 Definition and Properties

A 3D point cloud is a finite set of points in Euclidean space,

$$\mathcal{P} = \{p_i\}_{i=1}^{N}, \quad p_i \in \mathbb{R}^3, \tag{1.1}$$

where each point $p_i = (x_i, y_i, z_i)$ typically encodes the spatial location of a sampled surface element. Depending on the acquisition device, additional attributes such as color, intensity, surface normals, or semantic labels may be attached to each point, but in this thesis, we primarily focus on the geometric coordinates. Figure 1.1 shows an RGB image of an airplane alongside its corresponding 3D point-cloud representation. The point cloud contains only discrete XYZ

coordinates, emphasizing the sparse, unordered, and purely geometric form of the data that underlies all 3D analysis in this thesis. Point clouds differ fundamentally from grid-structured data such as images or volumes, in several ways.

First, point clouds are unordered. The set $\mathcal{P}$ has no canonical indexing: any permutation of the points represents the same underlying geometry. As a result, architectures designed for sequential or grid-ordered data cannot be applied directly, and 3D encoders must be permutation invariant (for global representations) or permutation equivariant (for point-wise predictions).

Second, point clouds are irregular. In contrast to images, where pixels lie on a regular lattice with fixed and known neighborhoods, point samples are distributed non-uniformly in space. Local density can vary significantly across the object due to sensor characteristics, occlusions, or reconstruction pipelines. Neighborhoods must therefore be defined explicitly through geometric criteria such as KNN or radius-based queries, and these neighborhoods change under resampling.

Third, point clouds are typically sparse. Even for high-resolution scans, the number of points is small relative to the continuous surface they approximate. Large empty regions separate local clusters of points, and there is no guarantee of complete coverage of the underlying geometry. This sparsity complicates the definition of local operators (analogous to convolutions on images) and makes models sensitive to how points are sampled, grouped into patches, and aggregated.

These properties—unorderedness, irregular sampling, and sparsity—make point clouds a flexible but challenging representation. They motivate the development of specialized architectures and learning frameworks that can respect geometric structure without relying on a fixed grid or canonical ordering.

1.2.2 Patchification

Point-cloud architectures differ in how they process raw 3D data, and not all of them require tokenization or patchification. Classical architectures such as PointNet (Qi *et al.*, 2017a) operate directly on the unordered point set by applying shared pointwise MLPs followed by global

Figure 1.1 A 2D RGB image of an airplane (left) and its corresponding 3D point-cloud representation (right), illustrating the fundamental difference between dense pixel grids and sparse, unordered 3D coordinates

pooling. Likewise, graph-based models such as DGCNN (Phan *et al.*, 2018) construct dynamic graphs at each layer and aggregate local features without explicitly creating fixed tokens or patches. In these architectures, the model learns representations from raw points or dynamically constructed neighborhoods without a predefined token structure.

In contrast, transformer-based and state-space–based architectures do require a tokenized representation. Transformers operate on sequences, and thus point clouds must be converted into a set of discrete tokens. Similarly, recent state-space models (SSMs) such as Mamba-based architectures process inputs sequentially, making tokenization essential both for defining the input elements and for determining the traversal order.

A common tokenization strategy begins by selecting a reduced set of representative points using FPS, which provides approximately uniform coverage of the point cloud. FPS selects center points iteratively by maximizing their distances from previously chosen centers:

$$c_i = \arg\max_{p \in \mathcal{P}} \min_{c_j \in \{c_1, \ldots, c_{i-1}\}} \|p - c_j\|_2. \tag{1.2}$$

These centers form a compact set of tokens that capture the global geometric structure.

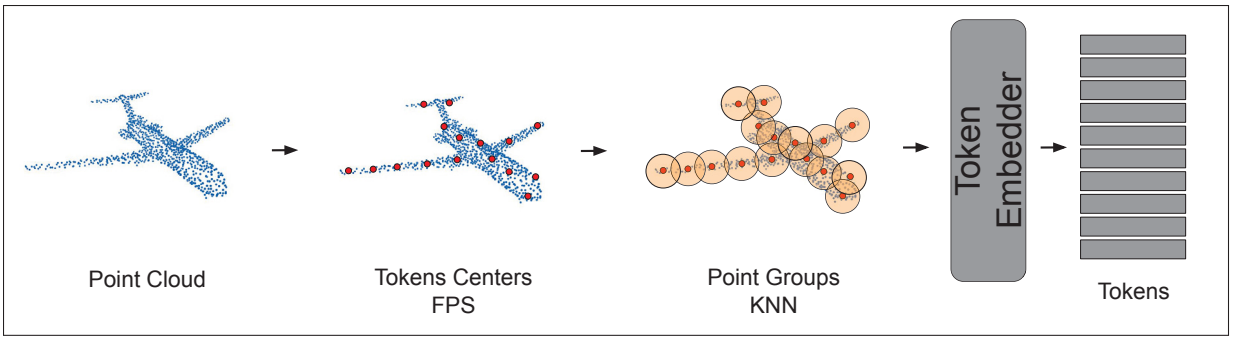

Figure 1.2 Given an input point cloud (left), token centers are first selected using farthest point sampling (FPS). Each center then forms a local neighborhood via k-nearest neighbors (KNN), producing grouped point patches. These patches are subsequently fed into a token embedder to generate the final sequence of tokens

For models requiring local context, each center point is expanded into a patch via K-Nearest Neighbors. For each center c_i, the neighborhood is defined as:

$$\mathcal{G}_i = \mathrm{KNN}(c_i, \mathcal{P}) \in \mathbb{R}^{K\times 3}. \tag{1.3}$$

Each patch $\mathcal{G}_i$ aggregates local geometric information, and the collection of such patches becomes the token sequence consumed by the model. The tokenization pipeline for 3D point clouds is illustrated in Figure 1.2.

This FPS + KNN patchification pipeline is now standard in transformer-based methods such as (Yu, Shen, Qu *et al.*, 2022; Pang *et al.*, 2022; Zhang, Zhang, Jin *et al.*, 2022b) and in state-space architectures for 3D data, where tokens and traversal define the computational pathway. Tokenization thus provides an essential bridge between irregular 3D point sets and sequence-based deep learning architectures.

1.3 Self-Supervised Geometric Representation Learning

1.3.1 Problem Definition: Learning Geometry Without Labels

SSL for 3D point clouds aims to learn geometric representations from unlabeled data so that the resulting encoders can be fine-tuned on downstream tasks with minimal annotations (Guo *et al.*, 2024; Xiao, Wang, Liu *et al.*, 2023). A point cloud contains no inherent structure beyond its 3D coordinates, and the objective in this setting is to capture both local surface patterns and global shape structure without relying on labels. Pretext objectives typically include reconstruction, contrastive matching, or transformation prediction, all of which must operate directly on the irregular and unordered nature of point-set data (Pang *et al.*, 2022; Yu *et al.*, 2022).

A central challenge arises from the widespread use of random masking strategies in masked reconstruction frameworks (Liu, Cai & Lee, 2022; Pang *et al.*, 2022; Zhang *et al.*, 2022b). Inspired by BERT and MAE in NLP and 2D vision (Devlin, Chang, Lee & Toutanova, 2019; He, Chen, Xie *et al.*, 2022), many 3D methods randomly drop points or patches and train a network to infer the missing geometry. However, random masking is oblivious to geometric importance: flat surfaces, smooth regions, and structurally informative areas are treated equally, even though they contribute very differently to the shape's topology (Wang, Qian *et al.*, 2023). As a result, the model may expend most of its capacity reconstructing geometrically trivial regions while failing to learn features sensitive to edges, high-curvature zones, or distinctive object parts.

A second challenge is the sensitivity of learned representations to sampling variation and noise. Real-world point clouds are acquired under diverse sensor resolutions, viewpoints, and environmental conditions, resulting in fluctuations in sampling density, occlusions, and various perturbations (Ren, Pan & Liu, 2022b). SSL frameworks typically assume relatively clean and uniformly sampled data during pretraining, which makes them vulnerable to common corruptions such as jitter, point dropouts, and density shifts (Levi & Avidan, 2023). Moreover, tokenization schemes based on FPS and kNN introduce stochasticity during patch construction: different random seeds can lead to different patch groupings, potentially degrading downstream

performance if the SSL objective is not robust to such variability (Liu, Shapira *et al.*, 2021a). This highlights a broader limitation of current SSL approaches—the representations they learn are not inherently stable under realistic acquisition perturbations.

Together, these challenges highlight the need for self-supervised methods that go beyond random masking, explicitly capture intrinsic geometric structure, and remain robust to sampling variation and sensor noise.

1.3.2 Masked Pretraining for 3D (MAE Variants)

Masked pretraining has become a dominant paradigm for self-supervised representation learning in 3D vision, following the success of BERT-style objectives in language (Devlin, Chang, Lee & Toutanova, 2018) and MAE frameworks in 2D vision (He *et al.*, 2022). Early point-based networks such as PointNet (Qi *et al.*, 2017a) and PointNet++ (Qi *et al.*, 2017b) established the foundation by introducing permutation-invariant feature extraction and hierarchical neighborhood aggregation. PointNet++ (Qi *et al.*, 2017b) is a foundational architecture for point-cloud representation learning that extends the original PointNet (Qi *et al.*, 2017a) by introducing a hierarchical, multi-scale feature extraction mechanism. Whereas PointNet applies a shared MLP and a symmetric pooling operator over all points globally, PointNet++ recursively applies PointNet modules over local neighborhoods, allowing the network to capture both fine-grained geometric details and global shape context.

Given an input point cloud

$$\mathcal{P} = \{(x_i, f_i)\}_{i=1}^{N}, \qquad x_i \in \mathbb{R}^3,\ f_i \in \mathbb{R}^{d_{\text{in}}}, \tag{1.4}$$

PointNet++ builds a hierarchy through a sequence of *Set Abstraction* (SA) layers. In each SA layer ℓ:

1. **Sampling.** FPS selects a representative subset of points,

$$C^{(\ell)} = \{x_j^{(\ell)}\}_{j=1}^{N_\ell}, \qquad N_\ell < N_{\ell-1}, \tag{1.5}$$

ensuring spatial coverage across the shape.

2. **Grouping.** For each sampled center $x_j^{(\ell)}$, a local neighborhood is constructed via a ball query or kNN search,

$$\mathcal{N}\left(x_j^{(\ell)}\right) = \left\{(x_k^{(\ell-1)}, f_k^{(\ell-1)}) : \|x_k^{(\ell-1)} - x_j^{(\ell)}\|_2 \le r_\ell\right\}, \tag{1.6}$$

where r_ℓ is the neighborhood radius.

3. **Local Feature Extraction.** A PointNet module is applied to each neighborhood. For center $x_j^{(\ell)}$, the output feature is

$$f_j^{(\ell)} = \gamma_\ell\left(\left\{\, \phi_\ell([\, x_k^{(\ell-1)} - x_j^{(\ell)},\ f_k^{(\ell-1)}\,]) \;\middle|\; (x_k^{(\ell-1)}, f_k^{(\ell-1)}) \in \mathcal{N}(x_j^{(\ell)})\right\}\right), \tag{1.7}$$

where ϕ_ℓ is a shared MLP and γ_ℓ is a symmetric pooling operator (commonly max pooling). Using relative coordinates $x_k^{(\ell-1)} - x_j^{(\ell)}$ allows the network to encode local geometric structure invariant to global translation.

By stacking multiple SA layers with progressively larger radii ($r_1 < r_2 < \dots$), the receptive field grows while the number of points decreases, forming a *fine-to-coarse* hierarchy analogous to the downsampling path in CNNs.

To address variations in sampling density, PointNet++ introduces two important extensions:

- **Multi-Scale Grouping (MSG).** Multiple neighborhoods with different radii ($r_{\ell,1}, r_{\ell,2}, \dots$) are constructed around each center, and their corresponding features are concatenated:

$$f_j^{(\ell)} = \text{concat}\left(f_{j,1}^{(\ell)}, f_{j,2}^{(\ell)}, \dots\right). \tag{1.8}$$

- **Multi-Resolution Grouping (MRG).** To handle non-uniform density, features from the current abstraction level are fused with features directly summarized from the *previous, finer level*. This allows the network to rely on coarser patterns when the local density is too sparse to support fine-grained feature extraction.

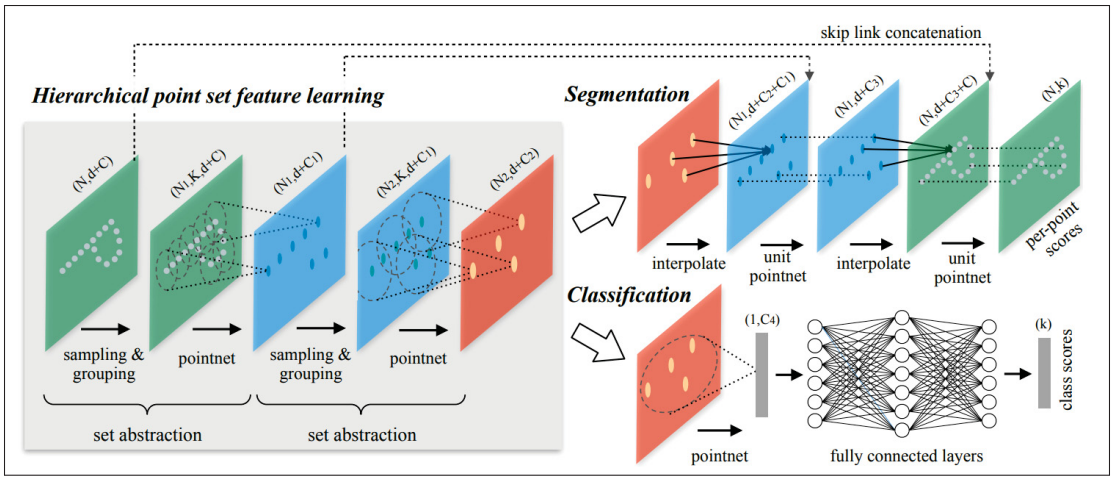

Figure 1.3 Illustration of hierarchical feature learning architecture and its application for set segmentation and classification using points in 2D Euclidean space as an example. Single scale point grouping is visualized here
Taken from (Qi *et al.*, 2017b)

Overall, PointNet++ provides a principled tokenization and aggregation mechanism for point sets, forming the backbone of many modern 3D architectures. Figure 1.3 illustrates the hierarchical set-abstraction pipeline used in PointNet++.

However, these architectures relied heavily on supervised labels, motivating the development of self-supervised alternatives capable of learning geometry directly from raw point sets.

A central line of work adapts masked reconstruction to irregular 3D data. Point-BERT (Yu *et al.*, 2022) demonstrated that point clouds can be tokenized into discrete latent codes using a dVAE and reconstructed through a Transformer encoder–decoder, enabling BERT-style pretraining on point patches.

Point-MAE (Pang *et al.*, 2022) extends the Masked Autoencoder (MAE) paradigm to irregular 3D point sets by introducing a transformer-based framework that reconstructs masked point patches. Given an input point cloud

$$\mathcal{P} \in \mathbb{R}^{N\times 3}, \tag{1.9}$$

the method first applies FPS to select M patch centers. For each center, the K nearest neighbors are gathered using kNN, producing a collection of local patches

$$\{\mathcal{P}_i\}_{i=1}^{M}, \qquad \mathcal{P}_i \in \mathbb{R}^{K\times 3}. \tag{1.10}$$

Each patch is normalized (e.g., by subtracting its centroid) and encoded with a lightweight PointNet-style network to obtain a patch embedding

$$z_i = f_{\text{patch}}(\mathcal{P}_i) \in \mathbb{R}^d. \tag{1.11}$$

A fixed masking ratio r is applied by randomly selecting a subset of patch indices to hide. Only the visible tokens are fed into a ViT-style encoder, while the decoder receives both the encoded visible tokens and learnable mask tokens inserted at the masked positions. The decoder performs several transformer layers and reconstructs the masked patches in 3D coordinate space. The reconstruction objective is defined as a Chamfer Distance loss between the original and generated masked patches:

$$\mathcal{L}_{\text{CD}} = \sum_{i\in\mathcal{M}} d_{\text{CD}}\left(\mathcal{P}_i, \hat{\mathcal{P}}_i\right), \tag{1.12}$$

where $\mathcal{M}$ denotes the set of masked patch indices and $\hat{\mathcal{P}}_i$ is the reconstructed patch. The Chamfer Distance between two point sets X and Y is given by

$$d_{\text{CD}}(X,Y) = \frac{1}{|X|}\sum_{x\in X}\min_{y\in Y}\|x-y\|_2^2 + \frac{1}{|Y|}\sum_{y\in Y}\min_{x\in X}\|x-y\|_2^2. \tag{1.13}$$

Through large-scale pretraining on ShapeNet and other datasets, Point-MAE learns rich geometric priors that transfer effectively to downstream tasks such as classification (ModelNet40, ScanObjectNN) and part segmentation (ShapeNetPart). Figure 1.4 illustrates the overall Point-MAE pipeline, including patch grouping, random masking, the transformer encoder–decoder, and masked patch reconstruction.

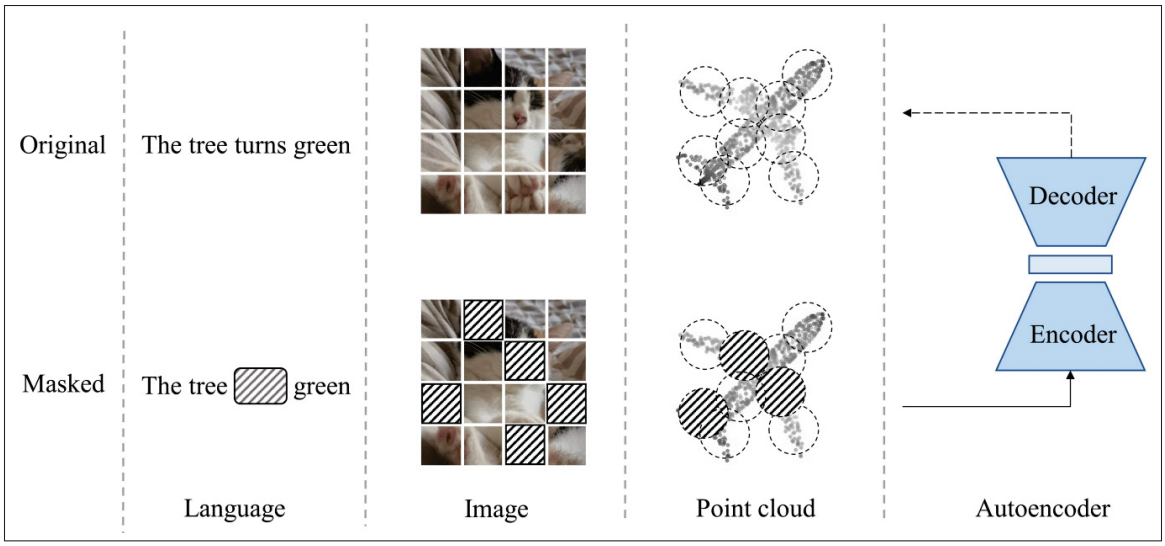

Figure 1.4 Overview of Point-MAE (Pang *et al.*, 2022). The point cloud is partitioned into local patches using FPS+KNN. A random subset of patch tokens is masked, and only visible tokens are encoded. The decoder inserts mask tokens and reconstructs masked patches via a transformer-based autoencoder
Taken from (Pang *et al.*, 2022)

MaskPoint (Liu *et al.*, 2022) introduced a discriminative alternative to the generative paradigm predominantly used in masked modeling. Identifying that standard reconstruction objectives—such as minimizing Chamfer distance—are sensitive to sampling variance and often lead to trivial solutions where the model memorizes coordinates rather than learning structure, the authors reformulate the pretext task as an occupancy learning problem. In this framework, the encoder processes only the visible subset of point patches, while the decoder is tasked with a binary classification objective: distinguishing between "real" points sampled from the masked regions and "fake" noise points sampled uniformly from the 3D space. By forcing the network to hallucinate whether a specific coordinate in the masked region belongs to the object or empty space, MaskPoint encourages the learning of global geometric semantics without the ambiguity inherent in dense point reconstruction.

Several follow-up methods have investigated different tokenization schemes and reconstruction formulations. Point-M2AE (Zhang *et al.*, 2022b) introduces a *multi-scale* masked autoencoding framework built upon a pyramid-structured Transformer backbone, specifically tailored for

hierarchical point cloud pre-training. Unlike single-scale Point-MAE, the encoder in Point-M2AE progressively downsamples point tokens across stages and applies a coordinated multi-scale masking strategy that preserves consistent visible regions across different resolutions. This enables the model to jointly capture coarse structural cues and fine-grained geometric detail. The decoder is implemented as a lightweight upsampling pyramid with skip connections, allowing multi-scale features to be fused gradually from global to local representations during reconstruction (Zhang *et al.*, 2022b). Through this hierarchical design, Point-M2AE effectively models both semantic shape structure and detailed local geometry, achieving strong performance on linear probing, full fine-tuning, part segmentation, and few-shot recognition tasks.

MAE3D (Jiang, Lu, Zhao, Dazeley & Wang, 2023) extends masked autoencoding to point-cloud representation learning by designing a patch-wise framework tailored to irregular 3D geometry. The method splits a point cloud into FPS–KNN patches, applies high-ratio block masking, and extracts latent features using a PointNet-based patch embedding module. An asymmetric Transformer architecture encodes only visible patches, while the decoder reconstructs both patch centers and complete geometry using a folding-based reconstruction head with a multi-task Chamfer loss. This patch-wise design enables MAE3D to capture strong local geometric structure and leads to improved performance on downstream classification tasks.

Point-GPT (Chen *et al.*, 2024) extends GPT-style autoregressive pretraining to point clouds by treating a 3D shape as an ordered sequence of point patches rather than an unordered set. The input point cloud is first partitioned into local patches and then arranged into a geometry-aware sequence based on spatial proximity, producing an approximate traversal that preserves global structure. An extractor–generator Transformer decoder is trained with a dual-masking strategy in an auto-regressive manner: conditioned on previously observed tokens, the model predicts the next patch in the sequence, thereby learning long-range dependencies and global geometric relations. After pretraining, the backbone is fine-tuned on downstream tasks such as classification and segmentation, where Point-GPT demonstrates strong performance on different datasets, highlighting the effectiveness of autoregressive generative modeling for 3D representation learning. Figure 1.5 illustrates the overall Point-GPT pipeline.

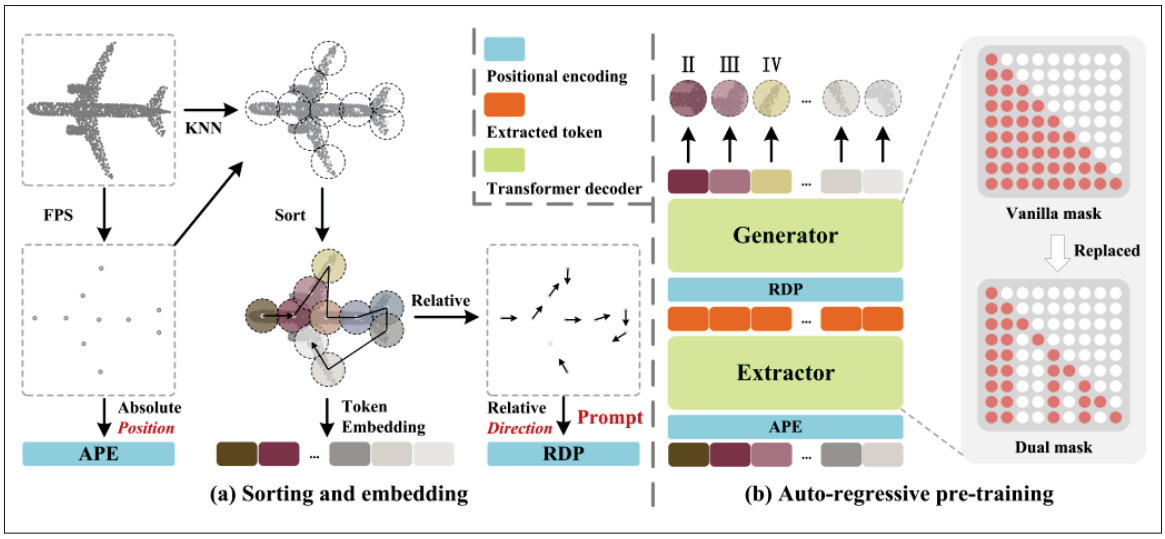

Figure 1.5 Overall architecture of PointGPT. (a) The input point cloud is divided into multiple point patches, which are then sorted and arranged in an ordered sequence. (b) An extractor-generator based transformer decoder is employed along with a dual masking strategy for the auto-regressively prediction of the point patches. In this example, the additional mask of the dual masking strategy is applied to the same group of random tokens for better illustration purposes
Taken from (Chen *et al.*, 2024)

Cross-modal transfer has also been explored as a way to inject strong inductive biases into point-cloud encoders. ACT (Dong *et al.*, 2022) transfers knowledge from pretrained 2D vision Transformers by aligning intermediate features across modalities: a point-cloud encoder is trained to mimic the representations of an image encoder through contrastive and reconstruction objectives, enabling 3D models to inherit structural priors such as texture–geometry consistency and global spatial organization. Similarly, I2P-MAE (Zhang *et al.*, 2023) performs masked autoencoding in the point domain while guiding the reconstruction process with features extracted from a pretrained image MAE. By projecting 2D image features onto 3D points and enforcing cross-modal consistency, I2P-MAE compensates for the sparse and irregular nature of point clouds, leading to richer geometric representations and improved generalization across downstream tasks.

Despite their success, most masked pretraining methods share two fundamental limitations. First, they rely heavily on *random masking* of points or patches. Random selection is agnostic to

geometric importance, treating flat surfaces and high-curvature regions as equally informative. This makes masked reconstruction dominated by smooth or uninformative areas, while the model learns little about edges, thin structures, or distinctive local geometry, as observed in several recent analyses.

This limitation motivates geometry-aware masking strategies that prioritize informative regions and strengthen local structural encoding—an idea developed in this thesis through contributions such as GeoMask3D in Chapter 2.

1.3.3 Computational Scalability in 3D SSL

Transformer-based architectures have become a dominant backbone for SSL on point clouds due to their ability to model long-range dependencies via self-attention. However, a core limitation of these models is the *quadratic* computational and memory complexity of self-attention with respect to the number of tokens. Given a token sequence of length N, a standard multi-head attention layer computes all pairwise interactions between tokens, leading to $O(N^2)$ time and space complexity. This is already challenging for high-resolution 2D images, and becomes even more restrictive in 3D, where raw point sets can easily contain tens or hundreds of thousands of points (Vaswani *et al.*, 2017; Guo *et al.*, 2021).

To make Transformer-based 3D SSL feasible, most methods apply aggressive token reduction before attention. Previous methods, such as Point-MAE (Pang *et al.*, 2022), Point-M2AE (Zhang *et al.*, 2022b), and MAE3D (Jiang *et al.*, 2023), first partition the point cloud into a relatively small number of local patches using FPS and kNN, then treat each patch as a single token. While this reduces N and makes masked pretraining tractable, it also introduces a trade-off: coarse patchification may blur fine-grained geometric details, whereas finer patch granularity leads to prohibitive attention cost. Similar issues arise in point-cloud Transformers such as Point Transformer (Zhao, Jiang, Fu, Jia *et al.*, 2021), which either restrict attention to local neighborhoods or rely on hierarchical pooling to control token count, at the cost of increased architectural complexity and potential loss of detail at early stages.

Another difficulty is that, unlike images where a regular grid induces a natural notion of locality, point clouds are irregular and unstructured. Local/self-attention schemes that constrain each token to attend only to its k nearest neighbors can mitigate the quadratic scaling, but their effectiveness depends heavily on the neighborhood construction and positional encoding used to approximate the underlying manifold (Zhao *et al.*, 2021; Wu, Qi & Fuxin, 2019). Poorly chosen neighborhoods or positional priors can lead to fragmented receptive fields or unstable geometry-aware attention, especially under non-uniform sampling and noise. As a result, scaling self-supervised Transformers to large, dense point clouds remains challenging: one must simultaneously control the $O(N^2)$ cost, preserve local geometric fidelity, and maintain sufficient receptive field for global context.

These limitations have motivated growing interest in alternative backbones with *linear* complexity in sequence length, such as state-space models and Mamba-style architectures, for 3D representation learning (Gu, Dao, Erichson, Rudin & Darrell, 2023). However, applying sequence-based architectures to point clouds remains challenging because point sets have no natural spatial order. Effective use of state-space models therefore requires constructing an explicit token traversal that maintains computational efficiency while accurately reflecting the geometry of the 3D surface.

1.3.4 State-Space Models in 2D and 3D SSL

State–space models (SSMs) were originally developed to describe the evolution of continuous-time dynamical systems, but recent advances have adapted them into deep sequence-processing layers capable of modeling long-range dependencies efficiently. Early neural SSMs such as HiPPO-Legendre memory units (Gu, Dao, Ermon, Rudra & Ré, 2020) and the LSSL layer introduced continuous-time representations but suffered from high computational overhead when applied to long sequences. The S4 architecture (Gu, Goel & Ré, 2022a) addressed this limitation by using structured state matrices, parameter normalization, and efficient kernelization, enabling sequence modeling with linear-time complexity. Subsequent extensions investigated diagonal

state parametrizations (Gupta, Faqir-Rhazoui, Dao & Gu, 2022), and liquid state formulations with adaptable dynamics (Hasani *et al.*, 2022).

The introduction of Mamba (Gu & Dao, 2024) marked a major shift in the design of state-space models by combining continuous-time SSM dynamics with a content-dependent *selection mechanism* that modulates how information is propagated across a sequence. In contrast to classical Linear Time-Invariant (LTI) SSMs, whose parameters (A, B, C) define fixed dynamics after discretization (as in Eq. 1.14 and 1.15), Mamba incorporates learnable, input-conditioned gating that allows the model to selectively retain or discard information at each step. This improves contextual awareness while preserving the linear-time recurrence structure of the discrete SSM,

$$h_k = \bar{A} h_{k-1} + \bar{B} x_k, \tag{1.14}$$

$$y_k = \bar{C} h_k, \tag{1.15}$$

making Mamba competitive with Transformers, but with significantly lower computational cost.

Building on these ideas, VMamba (Liu *et al.*, 2024) proposes a visual state-space backbone tailored to 2D images through a *selective scan* design. VMamba (Liu *et al.*, 2024) adapts the Mamba state-space architecture to vision by introducing Visual State Space (VSS) blocks equipped with the 2D-Selective-Scan (SS2D) module. Unlike standard Mamba, which operates on 1D sequences, SS2D unfolds image features along four complementary traversal paths, enabling each location to aggregate global context through directional selective-scanning while maintaining linear-time complexity. This design bridges the gap between the ordered structure of sequence models and the spatially non-sequential nature of images, offering an efficient alternative to quadratic self-attention. Built on stacks of VSS blocks across a hierarchical backbone, VMamba achieves strong performance on classification and dense prediction tasks, demonstrating that state-space models can serve as competitive vision transformers with superior input-scaling efficiency. Figure 1.6 illustrates the core selective-scan mechanism and its integration into the VMamba architecture.

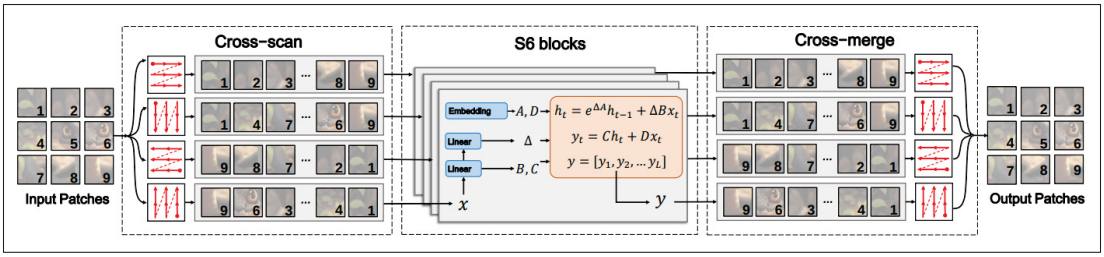

Figure 1.6 Illustration of 2D-Selective-Scan (SS2D). Input patches are traversed along four different scanning paths (Cross-Scan), with each sequence independently processed by separate S6 blocks. The results are then merged to construct a 2D feature map as the final output (Cross-Merge)
Taken from (Liu *et al.*, 2024)

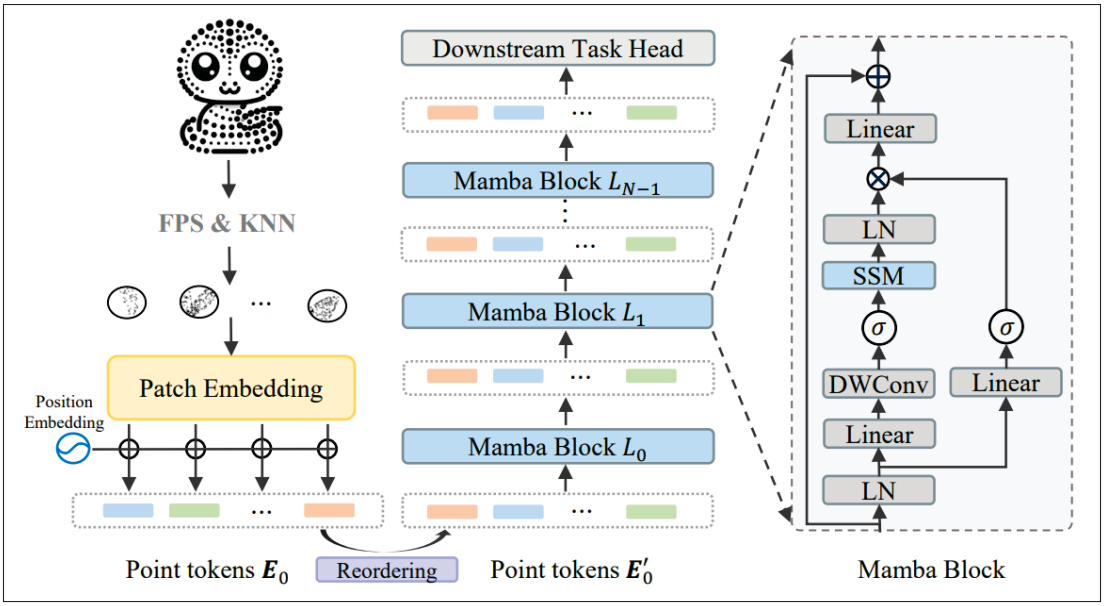

Figure 1.7 The pipeline of PointMamba. PointMamba starts by using FPS and KNN to obtain point patches, and then embed them into point tokens. After that, it reorders and triples the sequence. Finally, it feeds the token sequence into the Mamba blocks
Taken from (Liang *et al.*, 2024)

Similarly, Vision Mamba (Zhu, Gao, Shen & Li, 2024) introduces bidirectional Mamba blocks for building high-performance vision backbones, demonstrating that carefully designed SSM architectures can serve as competitive alternatives to Transformers in large-scale visual representation learning.

Extending SSMs to 3D point clouds is considerably more challenging. Unlike text or images, point clouds lack any canonical spatial ordering: points are unordered, irregularly sampled, and exhibit no grid-aligned structure that can be naturally traversed. As a result, the fundamental requirement of SSMs—processing inputs as sequences—is nontrivial in 3D. Recent attempts have relied on handcrafted traversal heuristics. Point-Mamba (Liang *et al.*, 2024) introduces one of the earliest attempts to adapt state-space models to 3D point clouds by designing a pipeline that converts unordered patches into a geometry-aware 1D sequence suitable for Mamba processing. As illustrated in the main architecture figure of the paper (Figure 1.7), the method begins by tokenizing the point cloud into local patches using FPS and kNN, followed by a lightweight PointNet encoder that produces a set of point tokens. Since Mamba performs causal, direction-sensitive sequence modeling, the tokens cannot remain in the random order typically used in Transformer-based point models. To address this, Point-Mamba proposes a simple but effective *axis-based reordering* strategy that sorts the patch tokens according to the coordinates of their patch centers along the x, y, and z dimensions. A detailed illustration of this reordering mechanism is shown in Figure 1.8. Instead of relying on the original random sequence—which lacks spatial coherence and disrupts the causal modeling behavior of Mamba—the tokens are sorted three times (once per axis) and concatenated into a tripled sequence. This produces a more orderly geometric scan that better matches Mamba's autoregressive dynamics while still preserving local spatial relationships among patches. The reordered sequence is then processed through stacked Mamba blocks, including layer normalization, depth-wise convolution, selective state-space updates, and residual connections. Despite its simplicity, this reordering strategy significantly improves global modeling capability and enables Point-Mamba to outperform Transformer baselines under comparable computational budgets.

Point Cloud Mamba (PCM) (Zhang *et al.*, 2025) adapts Mamba's state-space modeling to 3D point clouds by converting unordered coordinates into structured 1-D sequences using a Consistent Traverse Serialization strategy that preserves spatial adjacency. Multiple traversal variants are combined to provide diverse geometric perspectives, while order prompts and coordinate-based positional encodings enable the model to handle different sequence layouts.

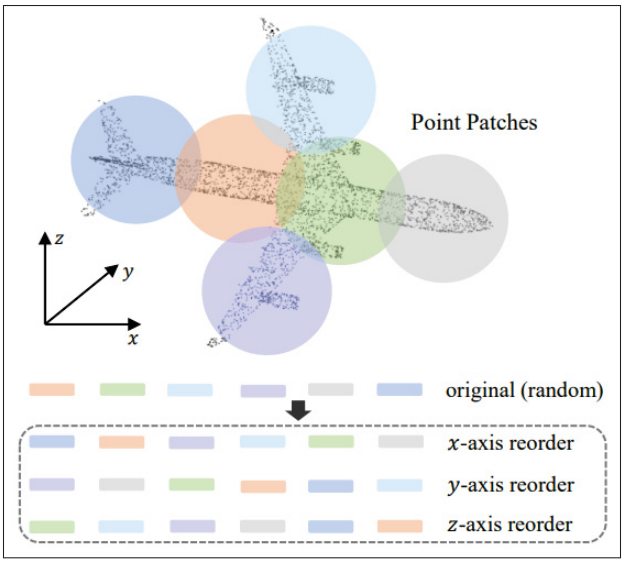

Figure 1.8 A simple illustration of the reordering strategy. The point patches are reordered and concatenated along the x, y, and z axes
Taken from (Liang *et al.*, 2024)

Integrated with local feature extraction modules, PCM achieves global modeling with linear complexity and improves performance across classification and segmentation benchmarks. This demonstrates that Mamba-based architectures can outperform transformer- and MLP-based point cloud networks when equipped with geometry-aware serialization and encoding mechanisms.

Some existing approaches share a common limitation: their traversal orders are based on handcrafted or heuristic rules that may introduce viewpoint dependency and are not inherently grounded in the intrinsic geometry of the point cloud. As a result, such orderings may not consistently preserve locality or rotational stability, and the resulting sequences may fail to fully capture the true geometric connectivity of the underlying 3D structure. This challenge motivates methods that compute traversal orders directly from the geometry itself. In Chapter 3, we build on this direction by introducing a spectral traversal strategy derived from the Laplacian eigenstructure of the point cloud's patch-connectivity graph. Unlike voxel or grid-based heuristics, this spectral ordering is isometry-invariant, preserves manifold continuity, and enables efficient Mamba-based masked autoencoding for 3D SSL.

1.4 Test-Time Learning for Robust 3D Perception

1.4.1 Test-Time Adaptation (TTA)

Modern machine learning systems frequently encounter distribution shift, where the data observed at test time differ from the distribution seen during training. Such shifts may arise from changes in sensors, environments, acquisition conditions, or noise characteristics. When this mismatch occurs, even well-trained models can experience substantial performance degradation, particularly in open-world or safety-critical applications.

TTA addresses the problem of distribution shift in the fully *source-free* and *label-free* setting, where a pretrained model must adapt to unseen target domains using only the incoming test samples. Unlike traditional domain adaptation, no source data, source statistics, or target labels are available during deployment. The key objective is therefore to update a model online so that its predictions remain stable when the test distribution deviates from the source distribution.

Since no ground-truth labels or source data are accessible, TTA methods optimize an *unsupervised test-time objective* on the target inputs. A common formulation is *entropy minimization*, which encourages confident predictions:

$$\mathcal{L}_{\text{ent}}(x_t;\theta) = -\sum_{c=1}^{C} p_\theta(y=c \mid x_t)\log p_\theta(y=c \mid x_t). \tag{1.16}$$

The model parameters are then updated online via

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_{\text{ent}}(x_t;\theta_t), \tag{1.17}$$

where η is the adaptation learning rate. In practice, many methods restrict updates to a small subset of parameters—such as the affine BatchNorm parameters (γ, β)—to avoid instability and computational overhead.

More generally, TTA aims to learn an adaptation operator

$$\theta_{t+1} = A(\theta_t, x_t), \tag{1.18}$$

that adjusts model parameters using only unlabeled test samples from the shifted target distribution $\mathcal{D}_t$, while preventing catastrophic drift in the absence of supervision. The overarching objective is to maintain reliable performance even as the test-time distribution evolves in ways unseen during training.

Early 2D advances in this direction introduced lightweight adaptation objectives operating directly on unlabeled test inputs. TENT (Wang, Shelhamer, Liu, Olshausen & Darrell, 2021a) instantiated entropy minimization in the TTA setting by updating only BatchNorm affine parameters during inference, offering a simple and efficient adaptation rule. SHOT (Liang, Hu & Feng, 2020) is a source-free unsupervised domain adaptation framework that adapts a pretrained source model to a target domain without any access to source data. The method freezes the source classifier (the "hypothesis") and learns a target-specific feature encoder using information maximization (certainty + diversity) to align target features to the fixed classifier. To further stabilize adaptation, SHOT introduces a self-supervised pseudo-labeling mechanism based on clustering target features, yielding SOTA results across closed-set, partial-set, and open-set UDA settings. MEMO (Zhang, Menon, Jiang, Andreassen & Kumar, 2022a) shows that performing entropy minimization on augmented versions of a single test input can also stabilize predictions without storing source data.

TTA for 3D Point Clouds. Applying TTA to point clouds introduces additional challenges due to sampling irregularity, noise sensitivity, and the absence of fixed spatial structure. Only a small number of works have explored TTA in the 3D domain. CoTTA (Wang *et al.*, 2024a) addresses continual TTA, where the target domain evolves over time rather than remaining static. To mitigate error accumulation from unreliable pseudo-labels, CoTTA uses weight-averaged and augmentation-averaged predictions for more stable adaptation. To prevent catastrophic forgetting, it periodically restores a random subset of neurons back to their source weights,

enabling long-term, fully-parameterized adaptation in dynamic environments. Other direction-specific TTA designs include MM-TTA (Shin, Choi, Kim & Choi, 2022) for multimodal 3D semantic segmentation, Point-TTA (Hatem, Taleb, Xiao & Xu, 2023b) for point cloud registration using auxiliary self-supervised objectives, and a meta-learned strategy for point-cloud upsampling (Hatem, Qian & Wang, 2023a). BFTT3D (Wang *et al.*, 2024b) introduces a backpropagation-free framework for 3D TTA, addressing the high computational cost and instability of gradient-based TTA methods. The model employs a two-stream architecture that preserves source-domain knowledge while extracting target-specific features through a non-parametric point cloud network composed of training-free operations such as FPS, k-NN grouping, trigonometric embeddings, and pooling. To bridge the source–target gap, BFTT3D performs subspace alignment via Transfer Component Analysis and derives target-specific logits through similarity matching against a compact prototype memory. These logits are combined with the source-model predictions using a novel entropy-based adaptive fusion mechanism, enabling robust adaptation without pseudo-labeling, self-supervision, or parameter updates. The overall pipeline is illustrated in Figure 1.9.

Although TTA has been extensively explored in 2D vision, its development for 3D point-cloud classification remains limited, and directly applying 2D methods to point clouds is fundamentally inadequate. Unlike images, point clouds are unordered, irregularly sampled, and highly sensitive to changes in density, occlusion, and FPS/KNN neighborhood construction; ignoring these geometric properties leads to unstable or unreliable adaptation. Existing TTA approaches for point clouds do not explicitly address these geometry-specific challenges and therefore lack robustness under realistic corruptions. To date, no method has been designed specifically for stable, corruption-resilient 3D point-cloud classification, leaving a critical gap in real-world deployment where models must adapt reliably using only unlabeled test samples.

In this direction, Chapter 4 presents Sampling-Variation Weight Averaging (SVWA), which takes an initial step toward 3D-specific TTA by exploiting sampling variation and stabilizing model behavior during adaptation.

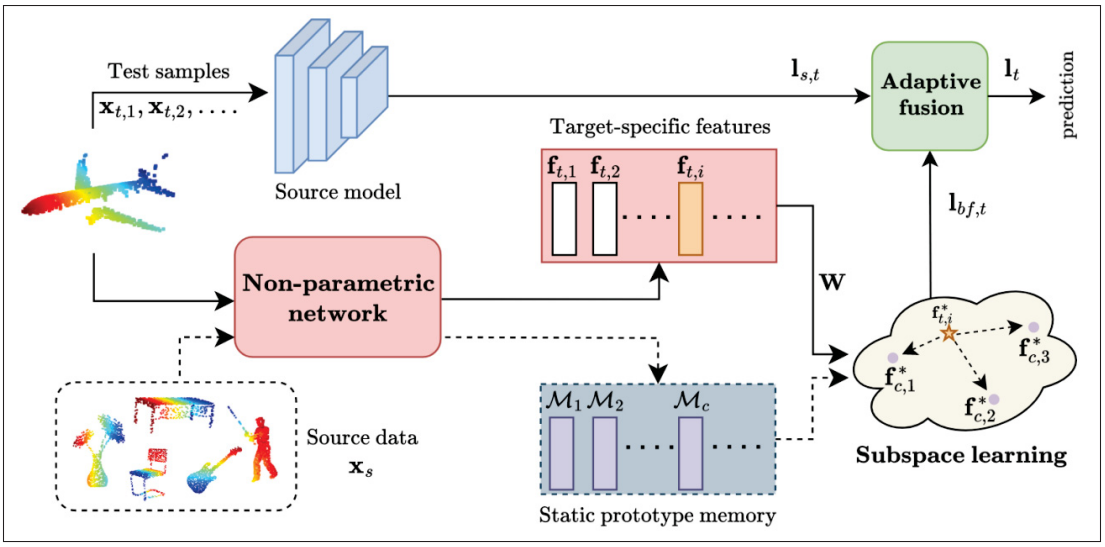

Figure 1.9 BFTT3D stores a compact prototype memory by extracting source features with a non-parametric network. At test time, the same non-parametric network produces a target feature f_t, which is projected into a shared subspace and matched against the stored prototypes to compute a target-specific logit l_{bf}. This logit is then fused adaptively with the source-model logit l_s using an entropy-based mechanism to produce the final prediction l_t. All components—feature extraction, subspace mapping, and fusion— operate without any backpropagation, enabling efficient test-time adaptation
Taken from (Wang *et al.*, 2024b)

1.4.2 Test-Time Training (TTT)

As mentioned, deep models often suffer from performance degradation when the test distribution differs from the training distribution. Unlike TTA, which performs unsupervised adaptation directly on test inputs, *TTT* equips the model with an auxiliary self-supervised task during pretraining, which is then re-used at test time to update the model parameters. Formally, given an auxiliary objective $\mathcal{L}_{\text{aux}}(x)$ learned during training, TTT performs online updates at test time via:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_{\text{aux}}(x_t; \theta_t), \tag{1.19}$$

allowing the model to correct itself on-the-fly without requiring source data or labels.

The first TTT method, TTT (Sun *et al.*, 2020), introduced the idea of learning an auxiliary rotation-prediction task during training and updating the network at test time via this self-

supervised signal. TTT++ (Liu, Wang, Zhu, Darrell & Gonzalez, 2021b) extends TTT by introducing an online feature alignment strategy—using offline feature summarization and moment matching—to keep test-time features close to the source distribution, and by replacing the rotation-prediction auxiliary task with a stronger contrastive-learning objective (SimCLR), which together yield more robust adaptation under severe distribution shifts. TTT-MAE (Gandelsman *et al.*, 2022) proposed a masked autoencoding-based auxiliary task to tackle one-sample learning and demonstrated strong performance on diverse visual benchmarks. These works highlight the central philosophy of TTT: a model should be equipped during training with an internal mechanism that allows it to self-correct when facing unseen test conditions.

TTT for 3D Point Clouds. TTT for 3D perception remains largely unexplored, with only a few emerging works extending the idea to geometric data. A representative example of TTT for 3D point clouds is MATE (Masked Autoencoders are Online 3D Test-Time Learners) (Mirza *et al.*, 2023). MATE targets point-cloud classification under distribution shift by attaching a masked autoencoder branch to a standard backbone (e.g., a PointNet++/Transformer-style encoder) and using it as an auxiliary self-supervised task during adaptation. At test time, each incoming point cloud is first partitioned into local patches; a large subset of these patches is then masked, and the MAE decoder is trained to reconstruct the missing geometry from the visible context. The classification head is never supervised on target labels; instead, the encoder is updated by backpropagating only the reconstruction loss, so that improvements in geometric encoding indirectly benefit the downstream classifier. MATE operates in an online fashion on a stream of test samples, repeatedly re-masking and reconstructing each target point cloud to refine the encoder's features under the shifted distribution. This design allows the method to remain label-free, while still leveraging powerful masked reconstruction signals tailored to 3D geometry. As illustrated in Figure 1.10, the overall pipeline consists of a shared encoder feeding both the classification head and a lightweight MAE decoder, with heavy masking applied to target point clouds and reconstruction-driven updates performed on-the-fly during inference.

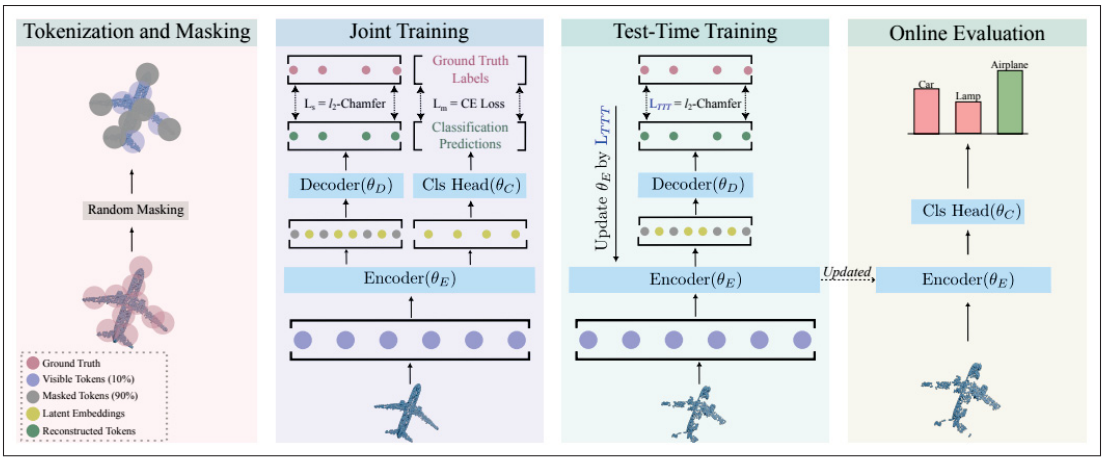

Figure 1.10 Overview of the 3D Test-Time Training pipeline in MATE. A point cloud is first tokenized into patches, after which a large portion of patches is randomly masked. During pretraining, visible patches are encoded to obtain latent features used simultaneously for classification and for reconstructing the masked patches via a lightweight MAE decoder. At test time, only the masked-reconstruction branch is used: the encoder is adapted on a single out-of-distribution input by minimizing the MAE reconstruction loss, and the updated encoder is then used for final classification
Taken from (Mirza *et al.*, 2023)

Other recent works explore meta-learning or backpropagation-free strategies for geometric tasks, but they primarily target segmentation, upsampling, or registration and do not directly address stable TTT for point cloud *classification*.

Despite recent progress, existing TTT methods for 3D point clouds remain extremely limited. The only method specifically designed for point-cloud classification, MATE, requires full per-sample backpropagation during inference, which makes adaptation slow and impractical for real-time 3D perception systems where latency is critical. Moreover, the broader TTT literature has focused almost entirely on 2D vision, leaving very few works exploring how to design efficient and stable TTT objectives for irregular, unordered 3D point sets. As a result, there is currently no TTT framework tailored for fast, stable, and corruption-resilient 3D point-cloud classification, creating an important gap between research progress and the demands of real-world 3D applications.

In this direction, Chapter 5 presents SMART-PC, a geometry-aware TTT framework that leverages skeletal structure to enable faster, more stable, and more corruption-resilient adaptation for 3D point-cloud models.

1.5 Final Remarks

This chapter has surveyed three tightly connected research axes that underpin modern 3D point-cloud perception: self-supervised geometric representation learning, scalable state-space architectures with meaningful traversal strategies, and test-time learning methods for robustness under distribution shift.

From the perspective of SSL, existing masked pretraining frameworks have shown strong downstream performance but mostly rely on geometry-agnostic random masking and coordinate reconstruction, which can overemphasize smooth, uninformative regions while underusing structurally salient parts. In parallel, state-space models such as Mamba and its visual variants offer linear-time sequence modeling, yet their success in 3D hinges on constructing traversal orders that preserve locality and manifold continuity in the absence of a natural grid. Finally, while TTA and TTT have been extensively explored in 2D vision, their direct application to point clouds remains challenging due to the unique properties of 3D data and the high computational cost of per-sample backpropagation, leaving a gap in stable, geometry-aware test-time learning for 3D classification.

These observations motivate the four research directions developed in the remainder of this thesis. Chapter 2 introduces GeoMask3D, a geometry-aware masking strategy that steers masked pretraining toward structurally informative regions. Chapter 3 proposes a spectral traversal mechanism that derives token orderings directly from the point-cloud geometry, enabling efficient Mamba-based masked autoencoding. Chapter 4 presents Sampling-Variation Weight Averaging (SVWA), which exploits sampling diversity to stabilize TTA in a 3D-specific manner. Chapter 5 introduces SMART-PC, a skeletal-structure–aware TTT framework designed for fast, stable, and corruption-resilient adaptation. Together, these contributions advance a unified,

geometry-centric view of pretraining, architectural design, and test-time learning for reliable 3D perception in real-world environments.

CHAPTER 2

GEOMASK3D: GEOMETRICALLY INFORMED MASK SELECTION FOR SELF-SUPERVISED POINT CLOUD LEARNING IN 3D

Ali Bahri[a] , Moslem Yazdanpanah[a] , Mehrdad Noori[a] , Milad Cheraghalikhani[a] , Gustavo A. Vargas Hakim[b] , David Osowiechi[a] , Farzad Beizaee[a] , Ismail Ben Ayed[b] , Christian Desrosiers[a]

[a] Department of Information Technologies Engineering, École de Technologie Supérieure
[b] Department of Systems Engineering, École de Technologie Supérieure
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

Paper published in *Transactions on Machine Learning Research (TMLR)* Journal, March 2025

Abstract

We introduce a novel approach to self-supervised learning for point clouds, employing a geometrically informed mask selection strategy called GeoMask3D (GM3D) to boost the efficiency of Masked Auto Encoders (MAE). Unlike the conventional method of random masking, our technique utilizes a teacher-student model to focus on intricate areas within the data, guiding the model's focus toward regions with higher geometric complexity. This strategy is grounded in the hypothesis that concentrating on harder patches yields a more robust feature representation, as evidenced by the improved performance on downstream tasks. Our method also presents a feature-level knowledge distillation technique designed to guide the prediction of geometric complexity, which utilizes a comprehensive context from feature-level information. Extensive experiments confirm our method's superiority over SOTA baselines, demonstrating marked improvements in classification, segmentation, and few-shot tasks. Code is available on *our GitHub*.

2.1 Introduction

The advent of large-scale 3D datasets (Deitke *et al.*, 2023; Slim *et al.*, 2023) has propelled research on deep learning for point clouds, leading to notable improvements in complex 3D tasks. However, the time-consuming nature of data collection, compounded by the complexity of

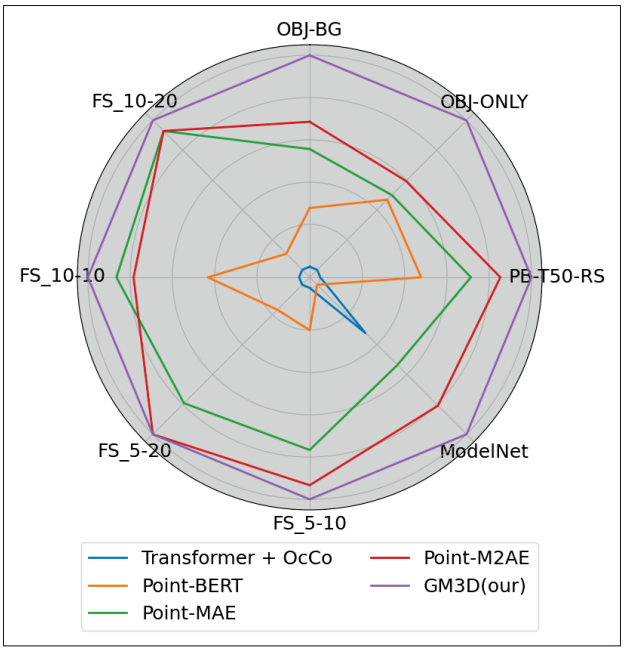

Figure 2.1 A relative comparison of the SOTA point cloud MAE methods on different tasks. Here, the center and the outer circles represent the lowest and highest values on each task, respectively

3D view variations and the mismatch between human perception and point cloud representation, significantly hinders the development of effective deep networks for this type of data (Xiao *et al.*, 2023). In response to this challenge, SSL has emerged as a promising solution, facilitating the learning of representations without relying on manual annotations. SSL not only circumvents the issues of costly and error-prone labeling but also improves the model's generalization ability, offering a pivotal advancement in the field of point cloud-based deep learning (Fei, Cao, Wang, Bai & Liu, 2023).

MAEs, as simple yet effective self-supervised learners, have gained prominence by learning to recover masked parts of data. This approach has significantly advanced NLP models and has resulted in exceptional vision-based representation learners (Devlin *et al.*, 2019; He *et al.*, 2022) when applied to vision tasks. Building on their success, MAEs have recently been adapted for point cloud representation learning, leading to SOTA methods including MaskPoint (Liu *et al.*, 2022), Point-Bert (Yu *et al.*, 2022), Point-MAE (Pang *et al.*, 2022), Point-M2AE (Zhang

et al., 2022b), I2P-MAE (Zhang *et al.*, 2023), and MAE3D (Jiang *et al.*, 2023). However, these methods share a common limitation stemming from the random masking strategy they use, where masked regions of the point cloud are selected arbitrarily without taking into account their informativeness. As demonstrated in recent work on Masked Image Modeling (MIM) (Wang *et al.*, 2023), employing a selection strategy that prioritizes informative regions over background areas can significantly enhance the robustness of the learned representation. While such a strategy has shown promising results for image processing, its application to point clouds has not been explored so far.

To bridge this gap, we study the use of a targeted masking strategy for point clouds within the MAE framework, applicable across both single and multi-scale methods. We introduce GeoMask3D (GM3D), a novel geometrically-informed mask selection strategy for object point clouds. Due to the lack of background data in the object's point clouds, GM3D enables models to concentrate on more complex areas, such as canonical ones with higher connections to the other areas, and pay less attention to the geometrically simple areas like smooth surfaces, as depicted in Figure 2.2. To showcase the effectiveness of our method, we integrate it into the pretraining process of both single-scale Point-MAE and multi-scale Point-M2AE, the leading MAE methods for point clouds. As illustrated in Figure 2.1, our method exhibits notable enhancements over the earlier SOTA approaches across a range of challenging tasks.

To the best of our knowledge, this represents the first attempt to implement a masking strategy for point clouds, independent of additional modalities such as multi-view images. Our contributions are summarized as follows:

1. We propose a novel masking approach for point cloud MAEs, which selects patches based on their geometric complexity rather than selecting them randomly. This approach employs an easy-to-hard curriculum learning strategy where the ratio of patches selected using geometric complexity is gradually increased during training.
2. We also introduce a feature-level knowledge distillation technique to further guide the prediction of geometric complexity. Instead of relying on the noisy and incomplete information of 3D points, this efficient technique transfers latent features from a frozen

teacher model, encoding higher-level information on the geometry, to the student model learning the point cloud representation.

3. We integrate these mechanisms into the pretraining process of single-scale Point-MAE and multi-scale PointM2AE, both of which are SOTA point cloud MAEs, significantly enhancing their performance in diverse downstream tasks.

2.2 Related Works

Point Cloud Learning. PointNet (Qi *et al.*, 2017a) established point cloud processing as a key method in 3D geometric data analysis by addressing the permutation issue of point clouds with a max-pooling layer. To further enhance performance and capture both local and global features, PointNet++ (Qi *et al.*, 2017b) introduced a hierarchical structure, expanding the receptive fields of its kernels recursively for improved results over PointNet. Another study (Jaritz *et al.*, 2019) focused on point cloud scene processing, where multi-view image features are combined with point clouds. In this study, 2D image features are aggregated into 3D point clouds and a point-based network fuses these features in 3D space for semantic labeling, demonstrating the substantial benefits of multi-view information in point cloud analysis.

MAE for Representation Learning. Leveraging the success of MAEs in text and image modalities, Point-BERT (Yu *et al.*, 2022) introduced an approach inspired by BERT (Devlin *et al.*, 2018) adapting Transformers to 3D point clouds. This approach creates a Masked Point Modeling task, partitioning point clouds into patches and using a Tokenizer with a discrete Variational AutoEncoder (dVAE) to produce localized point tokens, with the goal of recovering original tokens at masked points. Similarly, Point-GPT (Chen *et al.*, 2024) introduced an auto-regressive generative pretraining (GPT) approach to address the unordered nature and low information density of point clouds. ACT (Dong *et al.*, 2022) proposed a cross-modal knowledge transfer method using pretrained 2D or natural language Transformers as teachers for 3D representation learning. MaskPoint (Liu *et al.*, 2022), a discriminative mask pretraining framework for point clouds, represents the point cloud with discrete occupancy values and performs binary classification between object points and noise points, showing resilience to point

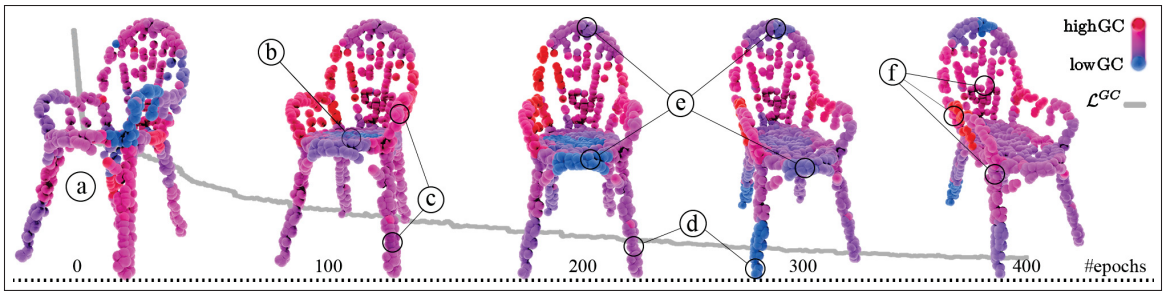

Figure 2.2 Visualization of estimated GC progression throughout training is depicted. The color spectrum denotes GC, ranging from low (Blue) to high (Red). GC values are normalized per object to reflect relative complexity across patches within each object's point cloud. As training progresses (from left to right), initial GC rankings display a random distribution (a). After 100 epochs, the model learns to assign lower complexity rankings to smooth areas (b) and higher rankings to complex regions (c). Through GC guided masking, the model increasingly focuses on complex areas from epochs 200 to 300, resulting in a reduction of GC ranking (d) and smoothing of the complexity ranking distribution, accompanied by a decrease in total complexity loss $\mathcal{L}^{GC}$ (e). Eventually, the model converges to a low $\mathcal{L}^{GC}$ value, consistently targeting canonical patches while maintaining a smoother GC distribution (f)

sampling variance. Point-MAE (Pang *et al.*, 2022) adapted MAE-style pretraining to 3D point clouds, employing a specialized Transformer-based autoencoder to reconstruct masked irregular patches and demonstrating strong generalization in various tasks. Following this, MAE3D (Jiang *et al.*, 2023) used a Patch Embedding Module for feature extraction from unmasked patches. Point-M2AE (Zhang *et al.*, 2022b) introduced a Multi-scale MAE framework with a pyramid architecture for self-supervised learning, focusing on fine-grained and high-level semantics. I2P-MAE (Zhang *et al.*, 2023) further improved the self-supervised point cloud learning process by leveraging pretrained 2D models through an Image-to-Point transformation.

Our proposed method, which can be integrated into any point cloud MAE architecture, differs from previous approaches like Point-MaskPoint, MAE, MAE3D and Point-M2AE that are based on random patch selection. Moreover, unlike recent point cloud learning approaches such as I2P-MAE (Zhang *et al.*, 2023), which rely on image information and 2D backbones, our method only requires 3D coordinates as input.

2.3 Method

2.3.1 Preliminaries

Masked Auto Encoders. Autoencoders use an encoder $\mathcal{E}$ to map an input X to a latent representation $Z = \mathcal{E}(X)$ and a decoder to reconstruct the input as $\hat{X} = \mathcal{D}(Z)$. Masked Auto Encoders (MAE) are a special type of autoencoder that receive a masked-patch input composed of a set of visible patches (with positional encoding) X^v and the index set of masked patches M, and reconstruct the masked patches as follows:

$$[X^v, \hat{X}^m] = \text{MAE}(X^v, M) = \mathcal{D}\big([\mathcal{E}(X^v), T_M]\big) \tag{2.1}$$

The encoder $\mathcal{E}$ and decoder $\mathcal{D}$ are both transformer-based networks. The encoder only transforms the visible patches to their latent representations $Z^v = \mathcal{E}(X^v)$. On the other hand, the decoder takes as input Z^v and a set of tokens $T_M = \big\{\big(t_{\text{mask}}, E_{\text{pos}}(i)\big) \mid i \in M\big\}$ where t_{mask} is a global learnable mask token and $E_{\text{pos}}(i)$ is the positional embedding of masked patch $i \in M$. Let $N = N^v + N^m$ be the number of patches in the input, with $N^v = |X^v|$ and $N^m = |M|$, the masking ratio is defined as $m^{\text{ratio}} = N^m/N$.

MAE for Point Clouds. A 3D point cloud P is a set of N^p points $p_j \in \mathbb{R}^3$. For this type of data, patches correspond to possibly overlapping subsets of K points in P. While there are various ways to generate patches from a point cloud, we follow the strategy employed by several related methods (Pang *et al.*, 2022; Zhang *et al.*, 2022b) where each patch x_i is defined as a center point c_i and the set $P_i \subset P$ of KNN to this center. To uniformly represent the whole point cloud, patch centers c_i are obtained using a FPS algorithm, where a first center is randomly chosen from P and then the next one is selected as the point in P furthest away from previously selected centers. Assuming that c_i is included as the first element of its nearest-neighbor list P_i, we can represent the patchified version of the point cloud as a tensor $X \in \mathbb{R}^{N\times K\times 3}$.

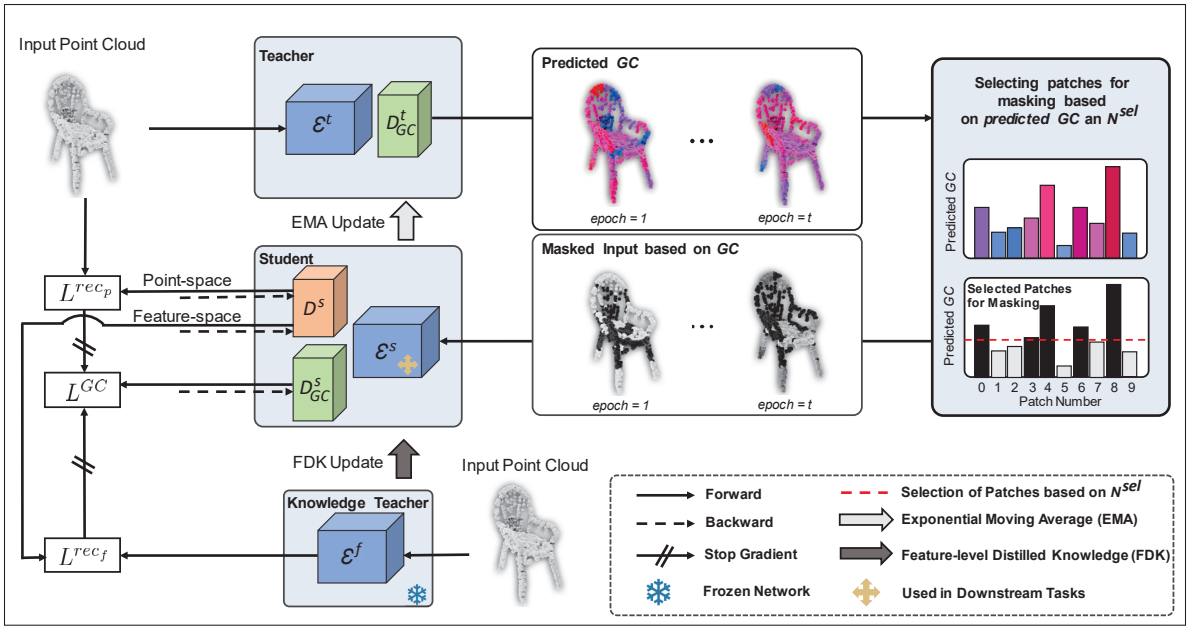

Figure 2.3 Overview of the GeoMask3D (GM3D) method for self-supervised representation learning in point clouds. The Teacher network predicts Geometric Complexity (GC), and patches with the highest GC, denoted by N^{sel}, are selected for masking. The Student network is then trained to reconstruct these masked tokens while simultaneously learning GC through the loss $\mathcal{L}^{GC}$. The reconstruction loss is defined as $\mathcal{L}^{\text{rec}} = \mathcal{L}^{\text{rec}_p} + \mathcal{L}^{\text{rec}_f}$. The Teacher network's weights are updated using the Exponential Moving Average (EMA) of the Student's weights, while the Knowledge Teacher remains frozen and is used for generating encoder features essential for the Student's training with $\mathcal{L}^{\text{rec}_f}$

In autoencoders for images, a pixel-wise L2 reconstruction loss is typically used for training the MAE. In our case, since patches $x_i \in X$ are points sets, we instead employ the Chamfer distance to measure the reconstruction error $\mathcal{L}^{\text{rec}_p}$ of masked patches:

$$\mathcal{L}^{\text{rec}_p} = \frac{1}{N^m} \sum_{i=1}^{N^m} \text{Chamfer}(x_i^m, \hat{x}_i^m), \tag{2.2}$$

where the Chamfer distance between two sets of points S and S' is defined as

$$\text{Chamfer}(S, S') = \sum_{p \in S} \min_{p' \in S'} \|p - p'\|_2^2 + \sum_{p' \in S'} \min_{p \in S} \|p - p'\|_2^2. \tag{2.3}$$

In the next section, we build on these definitions and present our GeoMask3D method for self-supervised point cloud representation learning.

2.3.2 GeoMask3D

In the originally proposed MAE, masked patches are selected randomly during each iteration without considering the varying impacts that different patches may have on the training process. This random masking approach may not be efficient, as patches in a point cloud can exhibit varying levels of Geometric Complexity (GC), which pose different degrees of challenge to the learning network. Inspired by the principles of human learning—where repeatedly tackling challenging tasks enhances performance over time—we propose prioritizing geometrically complex patches during the pre-training phase of a MAE network. A MAE network can achieve more efficient and effective learning by shifting from random masking to a strategy focusing on complex patches.

This strategy raises a fundamental question: what is Geometric Complexity (GC), and how can it be measured? We define GC for a patch as the relative difficulty of reconstructing that patch using an MAE network. Specifically, a patch is considered complex if the MAE network demonstrates difficulty in reconstructing it, as indicated by higher reconstruction loss $\mathcal{L}^{\text{rec}_p}$.

To this end, we propose GeoMask3D (GM3D) as a modular component that integrates with any point cloud MAE backbone. This component is incorporated into the pretraining phase of a chosen method, shifting from a basic naive random masking approach to a selective focus on geometrically complex patches for masking. The architecture of GM3D employs an auxiliary head $\mathcal{D}_{GC}$ for predicting the geometric complexity $GC \in \mathbb{R}^N$ across the input patches, which is trained with a loss $\mathcal{L}^{GC}$.

A teacher-student framework is utilized to integrate GM3D with a target network. We denote as $\text{GM3D}^s = (\mathcal{E}^s, \mathcal{D}^s, \mathcal{D}^s_{GC})$ the student and as $\text{GM3D}^t = (\mathcal{E}^t, \mathcal{D}^t, \mathcal{D}^t_{GC})$ the teacher, both of them having their own encoder, decoder and auxiliary head. In line with (He, Fan, Wu, Xie & Girshick, 2020), we apply a momentum update method to maintain a consistent teacher, updating it in

each iteration,

$$\text{GM3D}^t \;=\; \mu\cdot\text{GM3D}^t \;+\; (1-\mu)\cdot\text{GM3D}^s \tag{2.4}$$

where μ represents the momentum coefficient. Both networks predict the GC based on the patch's informational content, as elaborated in Section 2.3.2.1. We employ the prediction of GC in the masking strategy of the method during its pretraining stage. The GC of the student network (GC^s) is predicted based on the masked input X^v, while the GC of the teacher (GC^t) is calculated in inference mode using the complete input X:

$$GC^a = \begin{cases} \mathcal{D}^a_{GC}(\mathcal{E}(X)), & \text{if } a = t \;\; \text{(teacher)} \\ \mathcal{D}^a_{GC}\big([\mathcal{E}(X^v), T_M]\big), & \text{if } a = s \;\; \text{(student)} \end{cases} \tag{2.5}$$

The overview of our method for self-supervised representation learning in the point cloud is depicted in Figure 2.3. The GeoMask3D (GM3D) approach involves three interconnected steps, which will be explained in the following sections. Additionally, we provide a detailed explanation of the Knowledge-Distillation-Guided GC strategy in Section 2.3.3.

2.3.2.1 Prediction of Geometric Complexity (GC)

During this stage, our goal is to evaluate GC of each patch in X^m, relative to the others within the same set. We achieve this by using a Dense Relation Comparison (DRC) loss (Wang *et al.*, 2023) which enforces the GC of masked patch pairs (k, l), predicted by the student (i.e., GC^s_k and GC^s_l), to follow the same relative order as their loss values $\mathcal{L}^{\text{rec}}_k$, $\mathcal{L}^{\text{rec}}_l$:

$$\mathcal{L}^{GC} = \sum_{k=1}^{N^m} \sum_{\substack{l=1 \\ l\neq k}}^{N^m} \mathcal{I}^+_{kl} \log\left(\sigma(GC^s_k - GC^s_l)\right) \;-\; \mathcal{I}^-_{kl} \log\left(1 - \sigma(GC^s_k - GC^s_l)\right) \tag{2.6}$$

where $\mathcal{I}^+_{kl} = \mathbf{1}(\mathcal{L}^{\text{rec}}_k > \mathcal{L}^{\text{rec}}_l)$, $\mathcal{I}^-_{kl} = \mathbf{1}(\mathcal{L}^{\text{rec}}_k < \mathcal{L}^{\text{rec}}_l)$, $\sigma(\cdot)$ is the sigmoid function, and $\mathcal{L}^{rec}$ is detailed in Section 2.3.3.

This loss function enforces consistency between the predicted GC^s values and $\mathcal{L}^{\text{rec}}$ as the *ground truth*, effectively guiding the student model to learn a meaningful ranking of geometric complexities for the masked patches. By comparing all pairs of patches, the loss ensures a robust evaluation of relative complexity within X^m.

2.3.2.2 Geometric-Guided Masking

Patches with a high GC score are typically those that the model struggles to reconstruct accurately (see Figure 3.1). This difficulty often stems from their complex geometry, compounded by the absence of color and background information. While choosing those patches for masking might seem straightforward, there are two challenges to this approach. First, during training, the GC is evaluated by the student for masked patches X^m, yet we need to pick candidate patches from the entire set. Second, the student's GC estimation can be noisy, making the training unstable. We address both these challenges by instead selecting patches based on the teacher's score (GC^{t}). Thus, at each iteration, the teacher predicts the GC for all patches in X, including unmasked ones. Thanks to the momentum update of Figure 2.4, the teacher's predictions are more consistent across different training iterations.

2.3.2.3 Curriculum Mask Selection

During the initial phases of training, the model may struggle to reconstruct fine details and is often overwhelmed by the complexity of the point cloud structure. To mitigate this problem, we follow a curriculum easy-to-hard mask selection strategy by starting from pure random masking at the initial training epoch and gradually increasing the portion of geometric-guided masking until the maximum epoch $e_{\max}$. Let $A \in [0, 1]$ be the maximum ratio of patches that can be selected using GC^{t}. At each epoch e_t, we select the $N^{\text{sel}} = \lfloor e_t / e_{\max} \times A \times N^m \rfloor$ patches with highest GC^{t} value, and the remaining $N^m - N^{\text{sel}}$ ones are selected randomly based on a uniform distribution.

2.3.3 Knowledge-Distillation-Guided GC

Instead of relying exclusively on point geometry, our approach employs a knowledge distillation strategy to also learn from latent features. This strategy involves transferring geometric knowledge from a frozen teacher network F = $(\mathcal{E}^f, \mathcal{D}^f)$ that processes the full set of patches to the student GM3D^s observing unmasked patches. This encourages the student GM3D^s to replicate the feature activations of the knowledge teacher F, indirectly learning from the full structure of data. This unique setup enables the student network to benefit from the global geometric context provided by the teacher network, which is constructed from the complete point cloud. As a result, this process facilitates the learning of robust and meaningful representations, which improve performance on downstream tasks. The complexity of patches in the feature space is determined by employing the Mean Square Error loss between the output of $\mathcal{E}^f$ and the output of $\mathcal{D}^s$ before converting back to point space:

$$\mathcal{L}^{\text{rec}_f} = \frac{1}{N^m} \sum_{i=1}^{N^m} \left\| \mathcal{E}^f(X)_i - \mathcal{D}^s([\mathcal{E}^s(X^v), T_M])_i \right\|^2 \tag{2.7}$$

This loss, combined with the Chamfer loss $\mathcal{L}^{\text{rec}_p}$ applied in the point space, serves as the ground-truth loss $\mathcal{L}^{\text{rec}}$ for the prediction of GC:

$$\mathcal{L}^{\text{rec}} = \mathcal{L}^{\text{rec}_p} + \mathcal{L}^{\text{rec}_f} \tag{2.8}$$

The total training loss $\mathcal{L}$ is calculated as

$$\mathcal{L} = \alpha \mathcal{L}^{GC} + \beta \mathcal{L}^{\text{rec}_p} + \gamma \mathcal{L}^{\text{rec}_f} \tag{2.9}$$

where α, β, and γ are hyper-parameters.

2.4 Experiments

Several experiments are carried out to evaluate the proposed method. First, we pretrain both Point-MAE and Point-M2AE networks utilizing our GM3D approach on the ShapeNet (Chang *et al.*, 2015) training dataset. Moreover, we assess the performance of these pretrained models across a range of standard benchmarks, such as object classification, few-shot learning, and part segmentation. It is important to note that, to maintain a completely fair comparison, we exclusively utilize the encoder of the student network for downstream tasks, ensuring it is identical to the encoder used in the method of interest.

In our approach, we adopt network configurations consistent with those used in the Point-MAE and Point-M2AE models to guarantee a fully fair comparison, notably using masking ratios of 60% for Point-MAE and 80% for Point-M2AE. This involves the technique of dividing point clouds into patches, along with employing the KNN algorithm with predetermined parameters for consistent patch uniformity. While our autoencoder architecture, including the configuration of Transformer blocks in both encoder and decoder, generally follows the patterns established in these models, we have uniquely tailored the decoder's design specifically for the GC estimation purposes. Moreover, the specifics of our network's hyperparameters for the pretraining and fine-tuning phases are comprehensively detailed in the Supplementary Materials.

2.4.1 Pretraining Setup

We adopt the ShapeNet dataset (Chang *et al.*, 2015) for the pretraining of our technique, in line with the practices established by Point-MAE and Point-M2AE. This dataset, known for its diverse and extensive collection of 3D models across various categories, provides a robust basis for training and evaluation. It contains 57,448 synthetic 3D shapes of 55 categories.

After this pretraining phase, we assess the quality of 3D representations produced by our approach through a linear evaluation on the ModelNet40 dataset (Wu *et al.*, 2015). We extract 1,024 points from every 3D model in ModelNet40 and then pass them through our encoder, which remains unchanged during this phase to preserve the learned features. The linear evaluation is

performed by a SVM fitted on these features. This classification performance is quantified by the accuracy metrics detailed in Table 2.2. The results clearly indicate that our technique, when applied to Point-MAE and Point-M2AE, enhances the network's performance.

2.4.2 Downstream Tasks

Object Classification on Real-World Dataset. In self-supervised learning for point clouds, it is crucial to create a model that exhibits strong generalization abilities across various scenarios. The ShapeNet dataset, which is favored for pretraining, contains clean, isolated object models, lacking any intricate scenes or background details. Inspired by this limitation, and building on prior approaches, we put our methods to the test on the ScanObjectNN dataset (Uy *et al.*, 2019b), a more demanding dataset that represents about 15,000 real-world objects across 15 categories. This dataset presents a realistic challenge, with objects that are embedded in cluttered backgrounds, making it an ideal benchmark for assessing our model's robustness and generalization in real-world scenarios.

We carry out tests on three different variants: OBJ-BG, OBJ-ONLY, and PBT50-RS. It is important to note that we do not employ any voting techniques or data augmentation during the testing phase. The outcomes of these experiments can be found in Table 2.1. These results demonstrate that integrating the GM3D module with Point-MAE and Point-M2AE significantly boosts their object classification accuracy on this dataset. These findings underscore our method's effectiveness in complex real-world scenarios.

Object Classification on Clean Objects Dataset. For the task of object classification on the ModelNet40 dataset(Wu *et al.*, 2015), we evaluated our pretrained models using the same protocols and configurations as the Point-MAE approach. ModelNet40, featuring 12,311 pristine 3D CAD models across 40 categories, was divided into a training set of 9,843 models and a testing set of 2,468 models, adhering to established norms. Throughout the training, we employed common data augmentation strategies, including random scaling and shifting. To ensure fair comparisons, the standard voting method (Liu, Fan, Xiang & Pan, 2019b) was also

Table 2.1 Object classification on real-world ScanObjectNN dataset (Uy *et al.*, 2019b). We evaluate our approach on three variants, among which PB-T50-RS is the hardest setting. Accuracy (%) for each variant is reported. ꕤ represents self-supervised pretraining

Method	OBJ-BG	OBJ-ONLY	PB-T50-RS
PointNet (Qi *et al.*, 2017a)	73.3	79.2	68.0
SpiderCNN (Xu, Fan, Xu, Zeng & Qiao, 2018)	77.1	79.5	73.7
PointNet++ (Qi *et al.*, 2017b)	82.3	84.3	77.9
DGCNN (Wang *et al.*, 2019)	82.8	86.2	78.1
PointCNN (Li *et al.*, 2018c)	86.1	85.5	78.5
BGA-DGCNN (Uy *et al.*, 2019b)	-	-	79.7
BGA-PN++ (Uy *et al.*, 2019b)	-	-	80.2
GBNet (Qiu, Anwar & Barnes, 2021)	-	-	80.5
PRANet (Cheng, Chen, He, Liu & Bai, 2021)	-	-	81.0
Transformer (Yu *et al.*, 2022)	79.86	80.55	77.24
ꕤ Transformer-OcCo (Yu *et al.*, 2022)	84.85	85.54	78.79
ꕤ Point-BERT (Yu *et al.*, 2022)	87.43	88.12	83.07
ꕤ I2P-MAE (Zhang *et al.*, 2023)	94.15	91.57	90.11
ꕤ Point-GPT-S (Chen *et al.*, 2024)	91.6	90.0	86.9
ꕤ ACT (Dong *et al.*, 2022)	93.29	91.91	88.21
ꕤ Point-MAE (Pang *et al.*, 2022)	90.02	88.29	85.18
ꕤ **Point-MAE + GM3D**	**93.11**	**90.36**	**88.30**
ꕤ Point-M2AE (Zhang *et al.*, 2022b)	91.22	88.81	86.43
ꕤ **Point-M2AE + GM3D**	**94.14**	**90.70**	**87.64**

applied during the testing phase. According to Table 2.4, integrating our GM3D module with Point-MAE has yielded a classification accuracy of 94.20%, which surpasses the performance of the standalone Point-MAE and even the more complex Point-M2AE on this dataset.

Few-shot Learning. Following the protocols of earlier studies (Yu *et al.*, 2022; Sharma & Kaul, 2020; Wang *et al.*, 2021b), we conduct few-shot learning experiments on ModelNet40 (Wu *et al.*, 2015), using an n-way, m-shot configuration. Here, n is the number of classes randomly chosen from the dataset, and m is the count of objects randomly selected for each class. The $n \times m$ objects are utilized for training. During the test phase, we randomly sample 20 additional unseen objects from each of the n classes for evaluation.

Table 2.2 Linear evaluation on ModelNet40 (Wu *et al.*, 2015) by SVM

Method	SVM
MAP-VAE (Wang *et al.*, 2019)	88.4
VIP-GAN (Guo *et al.*, 2021)	90.2
DGCNN + Jiasaw (Yu *et al.*, 2022)	90.6
DGCNN + OcCo (Yu *et al.*, 2022)	90.7
DGCNN + CrossPoint (Yu *et al.*, 2022)	91.2
Transformer + OcCo (Yu *et al.*, 2022)	89.6
Point-BERT (Yu *et al.*, 2022)	87.4
Point-MAE (Pang *et al.*, 2022)	91.05
Point-MAE + GM3D	**92.30**
Point-M2AE (Zhang *et al.*, 2022b)	92.90
Point-M2AE + GM3D	**93.15**

Table 2.3 Part segmentation on ShapeNetPart (Yi *et al.*, 2016). mIoU_c (%) and mIoU_i (%) denote the mean IoU across all part categories and all instances in the dataset, respectively. $\mathfrak{ss}$ represents self-supervised pretraining

Method	mIoU_c	mIoU_i
PointNet (Qi *et al.*, 2017a)	80.39	83.70
PointNet++ (Qi *et al.*, 2017a)	81.85	85.10
DGCNN (Wang *et al.*, 2019)	82.33	85.20
Transformer (Yu *et al.*, 2022)	83.42	85.10
$\mathfrak{ss}$ Transformer + OcCo (Yu *et al.*, 2022)	83.42	85.10
$\mathfrak{ss}$ Point-BERT (Yu *et al.*, 2022)	84.11	85.60
$\mathfrak{ss}$ I2P-MAE (Zhang *et al.*, 2023)	85.15	86.76
$\mathfrak{ss}$ Point-GPT-S (Chen *et al.*, 2024)	84.10	86.2
$\mathfrak{ss}$ ACT (Dong *et al.*, 2022)	84.66	86.16
$\mathfrak{ss}$ Point-MAE (Pang *et al.*, 2022)	84.19	**86.10**
$\mathfrak{ss}$ **Point-MAE + GM3D**	**84.49**	86.04
$\mathfrak{ss}$ Point-M2AE (Zhang *et al.*, 2022b)	84.86	86.51
$\mathfrak{ss}$ **Point-M2AE + GM3D**	**84.91**	**86.52**

The results of our few-shot learning experiments are summarized in Table 2.5. In this highly saturated benchmark, the combination of the GM3D module exhibits outstanding performance across all tested scenarios. It is worth noting that I2P-MAE (Zhang *et al.*, 2023) which *additionally benefits from multiple 2D views* provides only marginal improvements in results. Furthermore, Point-GPT (Chen *et al.*, 2024) and ACT (Dong *et al.*, 2022), despite being SOTA

Table 2.4 Linear evaluation on ModelNet40 (Wu *et al.*, 2015). 'points' and 'Acc' denote the number of points for training and overall accuracy. 𝒮 represents self-supervised pretraining

Method	Points	Acc (%)
PointNet (Qi *et al.*, 2017a)	1k	89.2
PointNet++ (Qi *et al.*, 2017a)	1k	90.5
𝒮 SO-Net (Li, Chen & Lee, 2018a)	5k	92.5
DGCNN (Wang *et al.*, 2019)	1k	92.9
Point Transformer (Zhao *et al.*, 2021)	1k	93.7
Transformer (Yu *et al.*, 2022)	1k	91.4
𝒮 Transformer + OcCo (Yu *et al.*, 2022)	1k	92.1
𝒮 Point-BERT (Yu *et al.*, 2022)	1k	93.2
𝒮 Point-BERT (Yu *et al.*, 2022)	4k	93.4
𝒮 Point-BERT (Yu *et al.*, 2022)	8k	93.8
𝒮 Point-M2AE (Zhang *et al.*, 2022b)	1k	94.00
𝒮 Point-GPT-S (Chen *et al.*, 2024)	1k	94.00
𝒮 ACT (Dong *et al.*, 2022)	1k	93.5
𝒮 I2P-MAE (Zhang *et al.*, 2023)	1k	94.1
𝒮 Point-MAE (Pang *et al.*, 2022)	1k	93.80
𝒮 **Point-MAE + GM3D**	**1k**	**94.20**

and complex methods, show only slight improvements compared to each other and other SOTA methods. Our findings highlight the effectiveness of our method as our Point-MAE+GM3D model has already achieved higher accuracy than single-scale Point-MAE and multi-scale Point-M2AE.

Part Segmentation. Our method's capacity for representation learning was assessed using the ShapeNetPart dataset (Yi *et al.*, 2016), which includes 16,881 objects across 16 different categories. In alignment with the approaches taken in prior studies (Qi *et al.*, 2017a,b; Yu *et al.*, 2022), we sampled 2,048 points from each object to serve as input.

For this highly competitive benchmark, our GM3D method achieves a slight improvement on both the Point-MAE and Point-M2AE networks, as detailed in Table 2.3. Considering that our approach exclusively utilizes 3D information, the observed improvement over methods like I2P-MAE(Zhang *et al.*, 2023) that *supplement 3D with additional 2D data* is reasonable,

Table 2.5 Few-shot classification on ModelNet40. We report the average accuracy (%) and standard deviation (%) of 10 independent experiments. ⇳ represents self-supervised pretraining

Method	5-way		10-way	
	10-shot	20-shot	10-shot	20-shot
DGCNN (Wang *et al.*, 2019)	91.8± 3.7	93.4± 3.2	86.3± 6.2	90.9± 5.1
⇳ DGCNN + OcCo (Wang, Liu, Yue, Lasenby & Kusner, 2021b)	91.9± 3.3	93.9± 3.1	86.4± 5.4	91.3± 4.6
Transformer (Yu *et al.*, 2022)	87.8± 5.2	93.3± 4.3	84.6± 5.5	89.4± 6.3
⇳ Transformer + OcCo (Yu *et al.*, 2022)	94.0± 3.6	95.9± 2.3	89.4± 5.1	92.4± 4.6
⇳ Point-BERT (Yu *et al.*, 2022)	94.6± 3.1	96.3± 2.7	91.0± 5.4	92.7± 5.1
⇳ I2P-MAE (Zhang *et al.*, 2023)	97.0± 1.8	98.3± 1.3	92.6± 5.0	95.5± 3.0
⇳ Point-GPT-S (Chen *et al.*, 2024)	96.8± 2.0	98.6± 1.1	92.6± 4.6	95.2± 3.4
⇳ ACT (Dong *et al.*, 2022)	96.8± 2.3	98.0± 1.4	93.3± 4.0	95.6± 2.8
⇳ Point-M2AE (Zhang *et al.*, 2022b)	96.8± 1.8	98.3± 1.4	92.3± 4.5	95.0± 3.0
⇳ Point-MAE	96.3± 2.5	97.8± 1.8	92.6± 4.1	95.0± 3.0
⇳ **Point-MAE + GM3D**	**97.0± 2.5**	**98.3± 1.3**	**93.1± 4.0**	**95.2± 3.6**

especially considering the slight enhancements achieved by I2P-MAE. Furthermore, Point-GPT (Chen *et al.*, 2024) and ACT (Dong *et al.*, 2022), despite being SOTA and complex methods, show only slight improvements over each other and other SOTA methods. Based on the results of SOTA methods presented in Table 2.3, it is evident that this dataset is highly challenging and competitive. This highlights the effectiveness of our masking strategy in enhancing the understanding of detailed, point-wise 3D patterns.

2.4.3 Additional Visualization

Geometric Complexity. Figure 2.4 illustrates the GC of randomly selected point clouds from the ShapeNet dataset. This illustration highlights the model's capability to assess GC at the patch level, where the red points denote areas of high GC and blue ones indicate areas of low GC. As mentioned in our methodology section, the model bases the masking process on the predicted GC of the patches. Consequently, patches representing regions with higher GC are preferentially masked. This strategic masking induces the model to focus intensively on intricate point cloud regions containing salient geometric information, thereby enhancing its overall

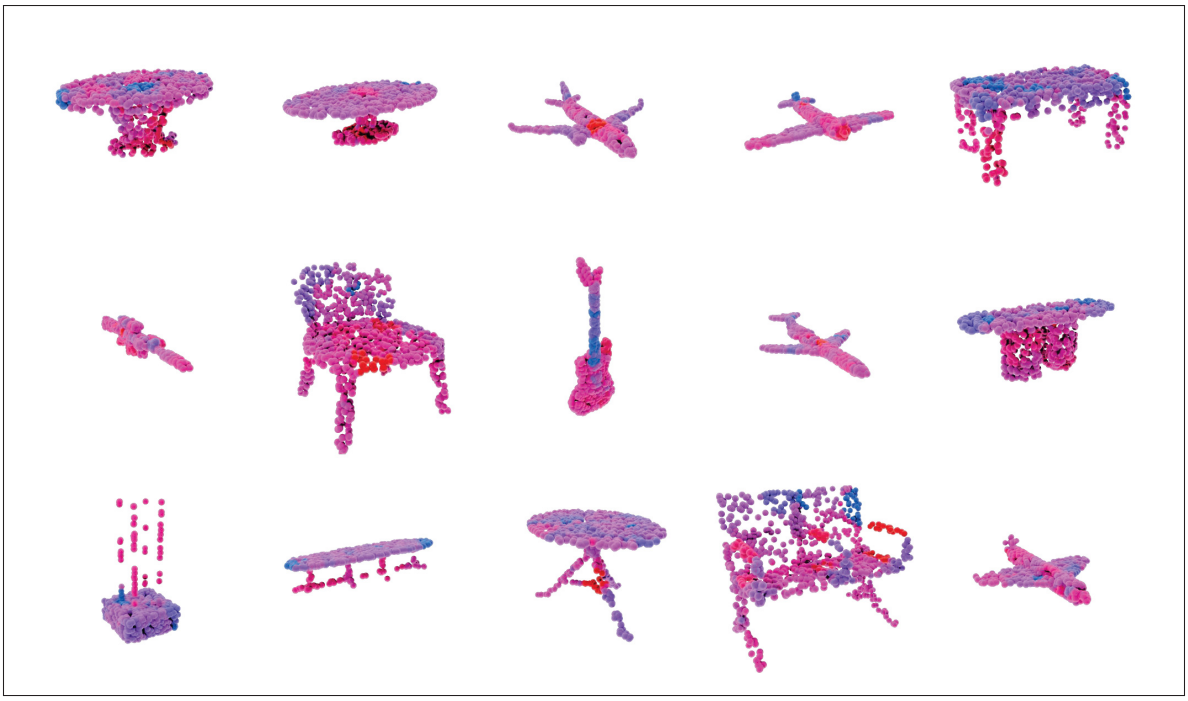

Figure 2.4 Visualization of GC values on diverse point clouds from the ShapeNet dataset (Chang *et al.*, 2015)

performance in tasks requiring nuanced geometric understanding. It is important to note that the provided normalized GC scores are computed relative to the individual patches within each point cloud sample from the ShapeNet dataset. This normalization ensures that the GC scores are a reflection of the variation in complexity within a given sample, enabling the model to internally assess and compare different regions of the same point cloud.

2.4.4 Additional Analyses

Pretraining Phase. The convergence rates shown in Figure 2.5 clearly highlight the efficiency of our proposed modules. Among the models, the Point-MAE+GM3D model stands out for its fast convergence, reaching high SVM accuracy with fewer epochs compared to the other methods. This quick convergence suggests that the GM3D module helps the model focus on important features in the data more effectively, speeding up the learning process.

The Point-MAE model, while still effective, achieves a lower accuracy and takes more epochs to get there, indicating that it learns more slowly. On the other hand, the Point-MAE+GM3D* version (integrating the GM3D method with Point-MAE alongside $\mathcal{L}^{\text{rec}_p}$ and $\mathcal{L}^{GC}$) is also effective but doesn't converge as quickly as the Point-MAE+GM3D, showing the importance of knowledge distillation alongside GM3D module. These findings highlight the practical benefits of adding the GM3D module to the Point-MAE framework. By helping the model learn faster and more reliably, the GM3D module not only improves the model's overall performance but also reduces the time needed to achieve high accuracy.

Fine-tuning Phase. Figure 2.6 displays the fine-tuning accuracy on the OBJBG dataset, providing clear evidence of the benefits brought by integrating our GM3D module with Point-MAE. The results reveal that the Point-MAE+GM3D model not only achieves the highest accuracy but also maintains this improvement consistently over the course of 400 epochs. This consistent performance highlights the stability and effectiveness of the GM3D module in guiding the model to learn more relevant features from the data.

2.4.5 Ablation Study

Our ablation study focuses on the incremental improvements offered by our proposed method GM3D when integrated with the original Point-MAE framework. The original Point-MAE serves as our baseline, using the Chamfer loss for self-supervised learning and setting a performance benchmark on subsequent pretraining and fine-tuning tasks.

Table 2.6 Comparison of Point-MAE, and Point-MAE+GM3D on Pretraining (SVM) and Fine-tuning (OBJ-ONLY) Tasks. '*' stands for our method without $\mathcal{L}^{\text{rec}_f}$

Model	Loss Function	SVM ModelNet40	OBJ-ONLY
Point-MAE	$\mathcal{L}^{\text{rec}_p}$	91.05	88.29
Point-MAE + GM3D*	$\mathcal{L}^{\text{rec}_p} + \mathcal{L}^{GC}$	**91.45**	**89.50**
Point-MAE + GM3D	$\mathcal{L}^{\text{rec}_p} + \mathcal{L}^{\text{rec}_f} + \mathcal{L}^{GC}$	**92.30**	**90.36**

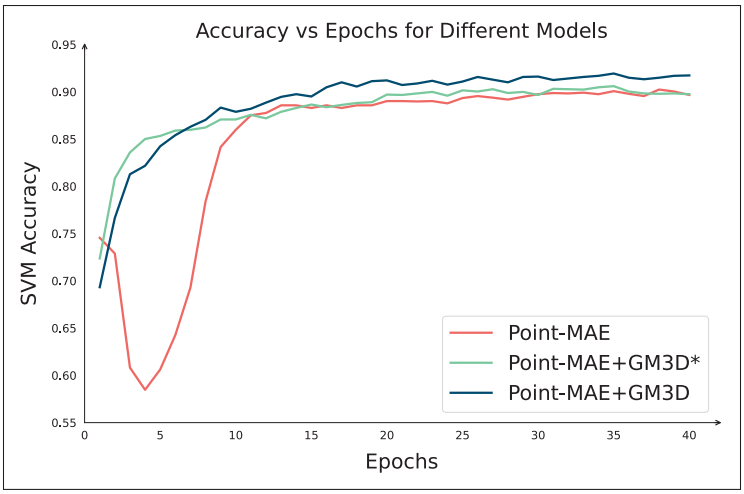

Figure 2.5 Comparison of convergence speed during the training phase (Point-MAE)

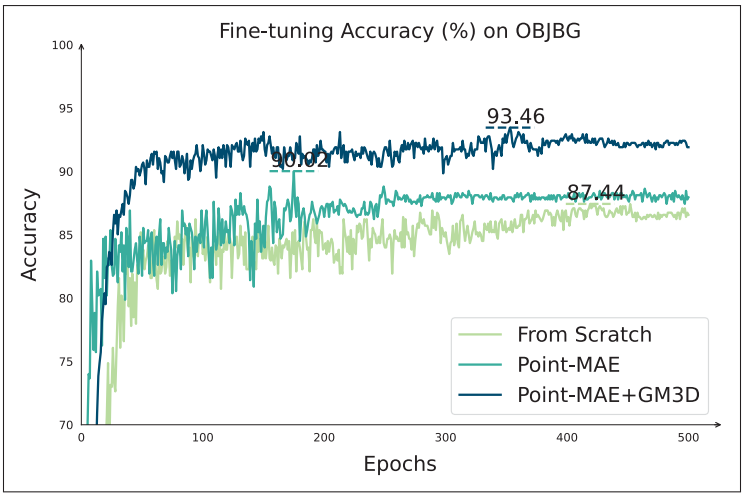

Figure 2.6 Fine-tuning vs. training from scratch on ScanObjectNN (Point-MAE)

GM3D. Initially, we integrate the GM3D method with Point-MAE alongside $\mathcal{L}^{\text{rec}_P}$ and $\mathcal{L}^{GC}$. As reported in Table 2.6, this combination, termed Point-MAE+GM3D*, shows a clear improvement over the baseline model by achieving higher pretraining SVM evaluation metrics on ModelNet40 and better fine-tuning results on ScanObjectNN (OBJ-ONLY). This supports the idea that a training focus on more geometrically complex patches contributes to improved model generalization.

Table 2.7 Ablation study on different maximum hard patch ratios (A). The highest performance is observed at 50%, where the OBJ-ONLY score reaches 90.36%

Model	A	OBJ-ONLY
Original Point-MAE	0	88.29
Point-MAE+GM3D	0.4	89.67
Point-MAE+GM3D	**0.5**	**90.36**
Point-MAE+GM3D	0.7	89.84

Building on this structure, we enhance the performance by incorporating knowledge distillation alongside the GM3D module. The improved model, Point-MAE+GM3D, which employs three distinct loss functions, not only outperforms the baseline Point-MAE but also shows further improvement over the Point-MAE+GM3D* approach that utilizes only $\mathcal{L}^{\text{rec}_p}$ and $\mathcal{L}^{GC}$. This advancement validates the effectiveness of our knowledge distillation strategy, which focuses on accurate reconstruction while also capturing the complex geometric interrelations in the data. The various impacts of knowledge distillation are further explored in Table 2.8.

Maximum Hard Patch Ratios. The data presented in Table 2.7 offer insights into the ablation study focusing on different hardness ratios, denoted by A, within the context of point cloud modeling. It is noteworthy that the inclusion of GM3D enhances the performance across different A settings when compared to the original model, with the highest performance observed at a 50% hardness ratio, where the OBJ-ONLY score reaches 90.36%.

Additional Configurations. In Table 2.8, which details our ablation study, we investigate the various configurations of our proposed method. The 'Input' column pertains to the input utilized by the knowledge-teacher network;

it specifies whether complete data X is provided or only partial data X^v are used. The second column, denoted by $\mathcal{L}$, encompasses both $\mathcal{L}^{\text{rec}_f}$ and $\mathcal{L}^{\text{rec}_p}$, representing the reconstruction loss functions in two spaces. In the third column, we analyze the impact of the chosen loss functions serving as the ground truth for $\mathcal{L}^{GC}$, which is our geometric complexity loss. As can be seen, the performance is enhanced by the geometric complexity guidance, which is informed

Table 2.8 Ablation study on different components of our method based on Point-MAE

	Input $\rightarrow F$		$\mathcal{L}$		$\mathcal{L}^{\text{rec}} \rightarrow \mathcal{L}^{GC}$		$GM3D^t$	$\mathcal{L}^{\text{rec}_f}$	OBJ-ONLY
	X^v	X	$\mathcal{L}^{\text{rec}_f}$	$\mathcal{L}^{\text{rec}_p}$	$\mathcal{L}^{\text{rec}_f}$	$\mathcal{L}^{\text{rec}_p}$	μ		
a	✓		✓		✓		✓	$\mathcal{E}^f \rightarrow \mathcal{D}^s$	89.32
b	✓		✓	✓	✓	✓	✓	$\mathcal{E}^f \rightarrow \mathcal{D}^s$	90.18
c		✓	✓	✓		✓	✓	$\mathcal{E}^f \rightarrow \mathcal{D}^s$	89.67
d		✓		✓		✓	✓	$\mathcal{E}^f \rightarrow \mathcal{D}^s$	89.50
e		✓	✓	✓	✓	✓		$\mathcal{E}^f \rightarrow \mathcal{D}^s$	89.33
f		✓	✓	✓	✓	✓	✓	$\mathcal{E}^f \rightarrow \mathcal{E}^s$	89.15
g		✓	✓	✓	✓	✓	✓	$\mathcal{E}^f \rightarrow \mathcal{D}^s$	**90.36**

by the feature-level knowledge distillation (rows c, and g). The subsequent column considers the influence of momentum, a parameter linked to the performance of GM3D$^{\text{t}}$. In the fifth column, we evaluate the impact of implementing $\mathcal{L}^{\text{rec}_f}$ on the interactions between various components of GM3D$^{\text{s}}$ and F. As evidenced by the results, each setting has been systematically varied to assess its effect on the final performance metric, OBJ-ONLY, demonstrating the significant contributions of each component to the model's learning efficacy.

2.5 Conclusion

We presented a geometrically-informed masked selection strategy for point cloud representation learning. Our GeoMask3D (GM3D) approach leverages a teacher-student model to find complex-to-reconstruct patches in the point cloud, which are more informative for learning robust representations. A knowledge distillation is further proposed to transfer rich geometric information from the teacher to the student, thereby improving the student's reconstruction of masked point clouds. Comprehensive experiments on several datasets and downstream tasks show our method's ability to boost the performance of point cloud learners.

CHAPTER 3

SPECTRAL INFORMED MAMBA FOR ROBUST POINT CLOUD PROCESSING

Ali Bahri[a] , Moslem Yazdanpanah[a] , Mehrdad Noori[a] , Sahar Dastani[a] , Milad Cheraghalikhani[a] , Gustavo A.Vargas Hakim[a] , David Osowiechi[a] , Farzad Beizaee[a] , Ismail Ben Ayed[b] , Christian Desrosiers[a]

[a] Department of Information Technologies Engineering, École de Technologie Supérieure
[b] Department of Systems Engineering, École de Technologie Supérieure
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

Paper published in *Computer Vision and Pattern Recognition (CVPR)*, June 2025

3.1 Abstract

State Space Models (SSMs) have shown significant promise in Natural Language Processing (NLP) and, more recently, computer vision. This paper introduces a new methodology that leverages Mamba and MAE networks for point-cloud data in both supervised and self-supervised learning. We propose three key contributions to enhance Mamba's capability in processing complex point-cloud structures. First, we exploit the spectrum of a graph Laplacian to capture patch connectivity, defining an isometry-invariant traversal order that is robust to viewpoints and captures shape manifolds better than traditional 3D grid-based traversals. Second, we adapt segmentation via a recursive patch partitioning strategy informed by Laplacian spectral components, allowing finer integration and segment analysis. Third, we address token placement in MAE for Mamba by restoring tokens to their original positions, which preserves essential order and improves learning. Extensive experiments demonstrate the improvements that our approach brings over state-of-the-art baselines in classification, segmentation, and few-shot tasks. The implementation is available at: https://github.com/AliBahri94/SI-Mamba.git.

3.2 Introduction

The analysis of 3D point cloud data is fundamental for various applications, including autonomous driving (Qi *et al.*, 2021; Shi, Wang & Li, 2019), VR/AR (Guo *et al.*, 2020), and robotics

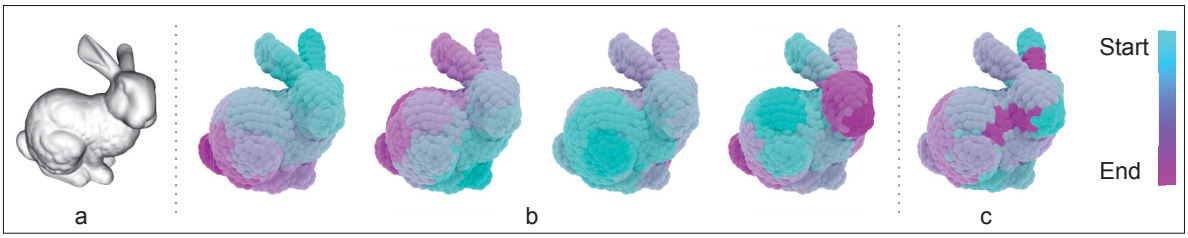

Figure 3.1 (a) Surface-Aware Spectral Traversing (SAST) applied to the patched point clouds of a mesh surface. (b) Traversal from left to right, based on the first to fourth non-constant smallest eigenvectors. (c) Traversal based on the largest eigenvector, forming a non-continuous sequence of tokens

(Rusu & Cousins, 2011). Compared to the organized structure of 2D images, point clouds consist of 3D coordinates without direct adjacency information forming an unordered bag. In recent years, considerable efforts have been dedicated to adapt deep learning models such as convolutional neural networks (CNNs) and Transformers to this type of data (Qi *et al.*, 2017b; Yu *et al.*, 2022; Pang *et al.*, 2022; Zhang *et al.*, 2022b; Bahri *et al.*, 2024b; Bahri, Desrosiers & Swamy, 2024a). Due to their permutation-invariant self-attention mechanism, Transformer networks are particularly well suited for the unordered nature of point clouds. However, the quadratic complexity of this mechanism, requiring the computation of a weight between each pair of tokens, impedes the application of these networks to large-sized inputs (e.g., 2D images or 3D point clouds represented by many patches). This has prompted researchers to explore more efficient solutions, including the Set Transformer (Lee *et al.*, 2019), Sparse Transformer (Child, Gray, Radford & Sutskever, 2019), Longformer (Beltagy, Peters & Cohan, 2020) and Sinkhorn Transformer (Tay, Bahri, Yang, Metzler & Juan, 2020).

Recently, methods based on Structured State Space Sequence (S4) (Gu *et al.*, 2022a) such as Mamba (Gu *et al.*, 2023) have gained significant traction as a more efficient alternative to Transformers (Liu *et al.*, 2024; Zhu *et al.*, 2024). So far, very few studies have investigated the potential of S4 approaches such as Mamba for 3D point clouds. Existing methods like Point-Mamba (Liang *et al.*, 2024) and PCM (Zhang *et al.*, 2025) extend the 2D grid-based traversal used for images to a 3D grid. However, this straightforward adaptation to point clouds suffers from three crucial problems. **First:** whereas patches from 2D images have adjacency

information, which could be exploited by the grid-based traversal, the 3D point patches in point clouds offer a sparse representation of the object's surface, and nearby patches on a 3D grid are not necessarily adjacent on this surface. **Second:** in the absence of self-attention, task-specific performance is highly influenced by the nature of the token traversal strategy. For example, a traversal suitable for point cloud classification may not be effective for a local task such as point-level classification (i.e., segmentation). **Third:** due to the "direction-sensitive" nature of Mamba, the self-supervised MAE pre-training step of leading point cloud models like Point-MAE (Pang *et al.*, 2022) and Point-M2AE (Zhang *et al.*, 2022b) cannot be used directly as there is no attention mechanism to learn the masked tokens' positions.

The contribution of our work focuses on addressing these problems as follows:

1. We introduce a SAST strategy based on the Laplacian spectrum of a patch-connectivity graph. Compared to the 3D grid traversal of current approaches like Point-Mamba, our strategy is invariant to isometric transformations (e.g., choice of viewpoint) and better captures the object's surface manifold.
2. We also present a HLT for point-level classification (segmentation) that partitions patches recursively based on their spectral coordinates. Unlike our SAST strategy for classification, which considers Laplacian eigenvectors separately in different traversals, this HLT combines them in a single ordering for a more precise modeling of geometry.
3. During the MAE-based SSL, we propose a TAR strategy to align the masked tokens according to their spectral adjacency. This strategy addresses the critical issue of spatial adjacency preservation unique to Mamba networks.

3.3 Related Works

Deep Point Cloud Learning. Deep neural networks have increasingly been applied to point clouds, with early architectures like PointNet (Qi *et al.*, 2017a) and PointNet++ (Qi *et al.*, 2017b) inspiring further work (Atzmon, Maron & Lipman, 2018; Deng, Zhang, Ding & Zhang, 2023; Landrieu & Simonovsky, 2018; Li *et al.*, 2018c; Zhao, Jiang, Fu & Jia, 2019) focused on capturing local context. Transformer-based frameworks (Vaswani *et al.*, 2017), including the

Point Transformer (v1–v3) (Wu *et al.*, 2023b; Wu, Lao, Jiang, Liu & Zhao, 2022; Zhao *et al.*, 2021) and Stratified Transformer (Lai *et al.*, 2022), have set new benchmarks by integrating both local and global features. Recently, self-supervised pre-training has gained traction by leveraging masked point modeling (Liu, Chen, Wang, King & Liu, 2023; Tang *et al.*, 2023) to train Transformers on unlabeled data.

Building on the effectiveness of MAE in Text and Image domains, Point-BERT (Yu *et al.*, 2022) presented a revolutionary method inspired by BERT (Devlin *et al.*, 2018), tailoring Transformers to 3D point cloud processing. Point-MAE (Pang *et al.*, 2022) applied MAE-style pre-training to 3D point clouds using a custom Transformer-based Autoencoder (AE) designed to reconstruct masked irregular patches. The use of multi-scale masking and local spatial self-attention mechanisms in Point-M2AE (Zhang *et al.*, 2022b) has led to SOTA results in 3D representation learning. Furthermore, I2P-MAE (Zhang *et al.*, 2023) improved self-supervised point cloud processing with a masking strategy leveraging pre-trained 2D models through an Image-to-Point transformation. Point-GPT (Chen *et al.*, 2024) introduced an auto-regressive generative pretraining (GPT) approach to address the unordered nature and low information density of point clouds. ACT (Dong *et al.*, 2022) proposed a cross-modal knowledge transfer method using pretrained 2D or natural language Transformers as teachers for 3D representation learning. Finally, GeoMask3D (Bahri *et al.*, 2024b) is a self-supervised point cloud method that uses a teacher-student model to select harder, geometrically intricate regions for masking, thereby improving feature representations.

3.3.1 State Space Models

State Space Models. SSMs have been foundational in control theory and signal processing for modeling dynamic systems. Recently, they have been adapted to deep learning, with advancements like the Linear State-Space Layer (LSSL), which uses a continuous-time memorization framework based on the HiPPO operator (Gu *et al.*, 2020) to model long-range dependencies. However, LSSL's high computational demands limit its practicality. To address this, the S4 model (Gu *et al.*, 2022a) introduced parameter normalization techniques,

paving the way for structured SSMs with enhancements like complex-diagonal structures (Gupta *et al.*, 2022; Gu, Goel & Ré, 2022b), support for multiple-input/output (Gu *et al.*, 2022b), and low-rank decomposition (Hasani *et al.*, 2022). Recent developments include Mamba (Gu *et al.*, 2023), which achieves linear-time inference and efficient training through selection mechanisms and hardware-aware algorithms. MoE-Mamba further boosts efficiency by integrating a Mixture of Experts (MoE), outperforming both Mamba and Transformer-MoE models (Pioro, Dao & Gu, 2024).

SSMs for Vision Tasks The above-mentioned works primarily focused on the application of SSMs to long-range or causal data types such as language and speech. In the field of vision, a notable study (Liu *et al.*, 2024) proposed the VMamba model which features a Cross-Scan Module (CSM) for enhanced 1D selective scanning in 2D spaces and architectural optimizations that significantly improve its performance and speed across various visual tasks. Another significant paper is Vision Mamba (Zhu *et al.*, 2024) which introduces a novel vision backbone called Vim utilizing bidirectional Mamba blocks.

For point cloud analysis based on Mamba, two key works are Point-Mamba (Liang *et al.*, 2024) and PCM (Zhang *et al.*, 2025). PointMamba introduces a simple approach to token reordering for point cloud analysis by strategically organizing point tokens based on a 3D grid. Similarly, PCM enhances Mamba with a Consistent Traverse Serialization (CTS) technique that converts 3D point clouds into 1D point sequences while maintaining spatial adjacency. Building upon these approaches, our method introduces the Spectral Spatial Traversing (SST) strategy, which improves token ordering and maintains spatial adjacency within Mamba networks.

3.4 Method

We begin by outlining the fundamental concepts of SSMs and spectral graph analysis which are at the core of our work. We then give an in-depth presentation of our Surface-Aware Spectral Traversing (SAST) strategy for point cloud processing that improves the model's robustness to isometric transformations and better captures the underlying manifold of the point cloud.

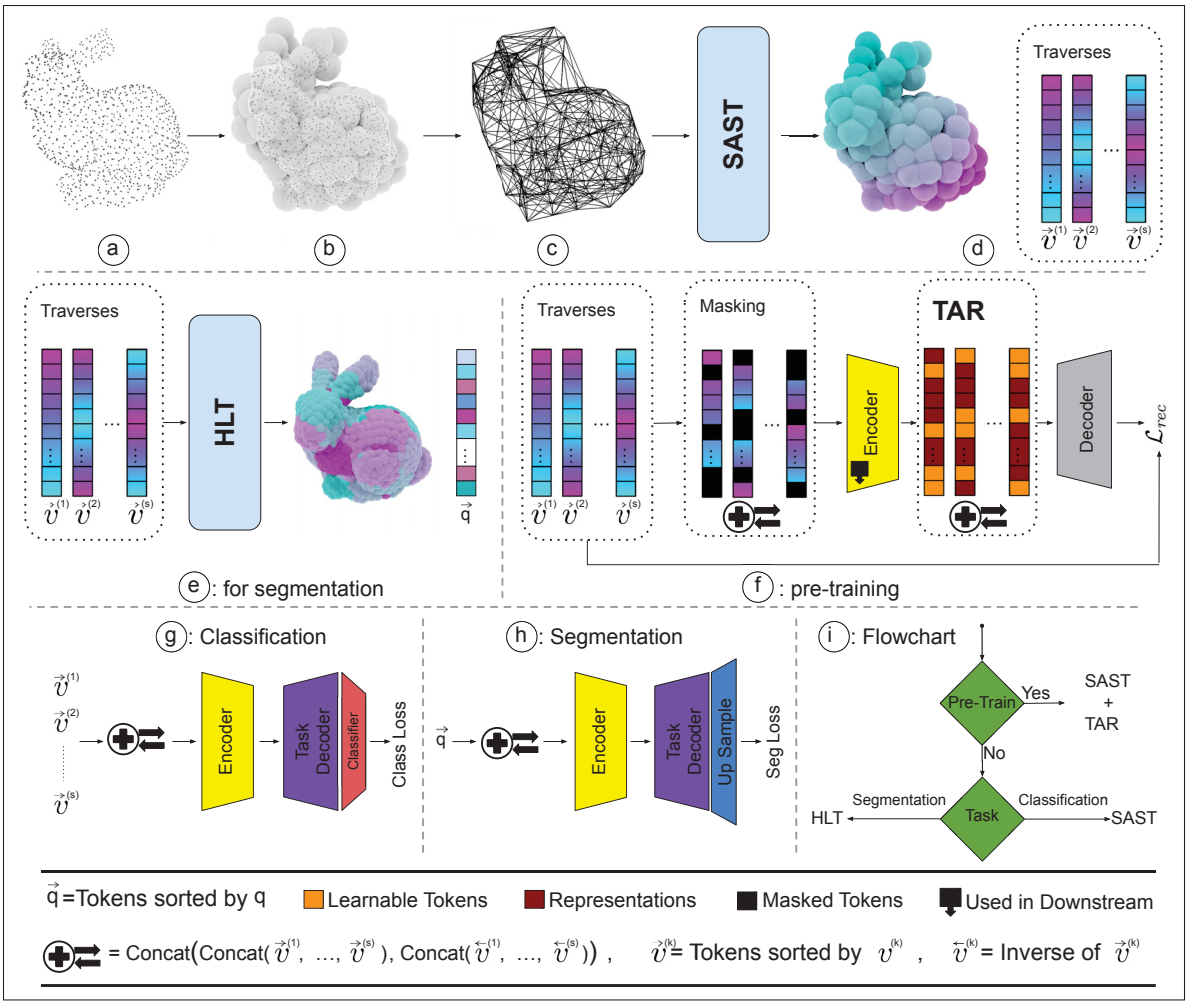

Figure 3.2 Overview of the proposed Spectral Spatial Traversing (SST) method. (a) Point cloud, (b) Patchification, (c) Forming the adjacency graph, (d) Traversal based on SAST using s non-constant smallest eigenvectors, (e) HLT for segmentation tasks, (f) TAR strategy for Masked Autoencoders. The process includes reverse and concatenation operations, with learnable tokens, representations, and masked tokens highlighted. (g) The classification task involves sorting tokens by different eigenvectors, concatenating them, and then feeding them into the network. (h) The segmentation task where HLT is applied on the tokens ($\vec{q}$) and $\vec{q}$ is fed into the network. (i) A flowchart visualizing the techniques used in self-supervised learning and various downstream tasks

Thereafter, we provide detailed specifications of our Hierarchical Local Traversing (HLT) strategy for point-level classification, which defines a more structured patch traversal order based on the recursive partitioning of spectral information. Finally, we introduce our Traverse-Aware Repositioning (TAR), which improves the handling of learnable tokens in masked autoencoders

within Mamba networks. Figure 3.2 illustrates the overview of the proposed Spectral Spatial Traversing (SST) method.

3.4.1 Preliminaries

State Space Models (SSMs) use a series of first-order differential equations to describe how the state of the linear, time-invariant system evolves over time:

$$\dot{h}(t) = Ah(t) + Bx(t), \quad y(t) = C\dot{h}(t) + Dx(t), \tag{3.1}$$

Here, $\dot{h}(t)$ denotes the time derivative of the state vector $h(t)$. The matrices A, B, C, and D are the weighting parameters.

Due to their reliance on continuous data streams $x(t)$, SSMs are not natively equipped to handle discrete inputs represented as $\{x_0, x_1, \ldots\}$. This necessitates the use of a discretized SSM version for practical applications:

$$h_k = \bar{A}h_{k-1} + \bar{B}x_k, \quad y_k = \bar{C}h_k + \bar{D}x_k. \tag{3.2}$$

The state space model in its discrete version utilizes a recursive function to link each state h_k to its preceding state, encapsulated by the matrices $\bar{A} \in \mathbb{R}^{N\times N}$, $\bar{B} \in \mathbb{R}^{N\times 1}$, and $\bar{C} \in \mathbb{R}^{N\times 1}$, which are tuned parameter matrices. While matrix $\bar{D} \in \mathbb{R}^{N\times 1}$ may be employed as a residual connection, we follow previous work and exclude it from our model. The transition from a continuous signal representation $x(t)$ to a discrete sequence involves sampling $x(t)$ at intervals defined by Δ, setting each discrete input as $x_k = x(k\Delta)$. This adjustment to a discrete framework results in revised matrix definitions:

$$\bar{A} = (I - \frac{\Delta}{2}A)^{-1}(I + \frac{\Delta}{2}A), \bar{B} = (I - \frac{\Delta}{2}A)^{-1}\Delta B, \bar{C} = C. \tag{3.3}$$

However, the fixed dynamics of Linear Time-Invariant (LTI) models, exemplified by the constant parameters A, B, and C in Eq. (3.3), restrict their capacity to selectively retain or discard

relevant information, thereby limiting their contextual awareness. To improve content-aware reasoning, we use Mamba's selection mechanism that manages the propagation and interaction of information across the sequence dimension (Gu *et al.*, 2023).

Spectral Graph Analysis. Popularized by Chung in the 90s (Chung, 1997), spectral graph analysis characterizes the properties of a graph $G = (V, E)$ by the spectrum (eigenvalues and corresponding eigenvectors) of its Laplacian matrix L. This analysis can be understood as a discretized version of the Laplace-Beltrami Operator Δ of a function f defined on a Riemannian manifold:

$$\Delta f = \text{div}(\text{grad} f) \tag{3.4}$$

where $\text{grad} f$ is the gradient of f and div the divergence on the manifold. The solution to the Laplacian eigenvalue problem $\Delta f = -\lambda f$, known as Helmholtz wave equation, is an eigenfunction corresponding to the natural vibration form of a homogeneous membrane with eigenvalue λ (Reuter, Wolter & Peinecke, 2005).

Following methods for spectral clustering (Ng, Jordan & Weiss, 2001) and normalized cuts (Shi & Malik, 2000), we consider a weighted adjacency matrix $W : V \times V \rightarrow \mathbb{R}_+$ where $W_{ij} = 0$ if $(i, j) \notin E$ to model the Euclidean distance of nearby patches (see Section 3.4.3). The Laplacian matrix of G is defined as $L = D - W$ where D is the diagonal *degree* matrix such that $D_{ii} = \sum_j W_{ij}$. To account for variability in the scale of weights W_{ij} or the distribution of node degrees D_{ii}, it is preferable to employ a normalized version of the Laplacian. In this work, we use the Random Walk Laplacian $L_{\text{rw}} = I - D^{-1}W$ which has the following useful properties:

1. L_{rw} is positive semi-definite and has $|V|$ non-negative real-valued eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_{|V|}$;
2. 0 is an eigenvalue of L_{rw} with the constant vector as eigenvector and its multiplicity equals the number of connected components in the graph;
3. Following Courant's Nodal Line Theorem (Courant & Hilbert, 2008), the n-th eigenmode of L_{rw} has at most n poles of vibration;

4. The representation of a shape by the spectrum of L_{rw} is invariant to isometry (i.e., distance-preserving transformation).

Our method uses the s first non-constant eigenvectors of L_{rw} (i.e., the eigenvectors corresponding to the s smallest non-zero eigenvalues) to define traversal orders for classification (Section 3.4.3) and segmentation (Section 3.4.4) that are robust to the viewpoint (due to isometry invariance) and provide a smooth parametrization of the surface manifold. We consider the first eigenvectors as they encode low frequency information (by Courant's Nodal Line Theorem), making the resulting traversal more robust to shape variability and noise. Figure 3.1 illustrates this concept: (a) shows the original mesh, (b) shows traversals based on the first to fourth non-constant smallest eigenvectors, and (c) shows traversal based on the largest eigenvector forming a non-continuous sequence of tokens.

3.4.2 Point Cloud Patchification

Given a point cloud $\mathcal{P} = \{p_i\}_{i=1}^{N_p}$, each point represented by 3D coordinates, we convert $\mathcal{P}$ to a reduced set of patches that can be processed more efficiently. Toward this goal, we employ the FPS algorithm to select a subset $\mathcal{C} \subset \mathcal{P}$ of N_c points offering a good coverage of the entire point cloud. These selected points will act as the centers of local patches within the point cloud. For each center point $p_{s_i} \in \mathcal{C}$, we then identify N_n nearest points $\mathcal{N}(p_{s_i}) \subset \mathcal{P}$ using the KNN algorithm. Following this, each patch is defined as a center p_{s_i} and its corresponding nearest-neighbors $\mathcal{N}(p_{s_i})$.

3.4.3 Surface-Aware Spectral Traversing (SAST)

Current point cloud processing approaches using Mamba, such as Point-Mamba (Liang *et al.*, 2024) and PCM (Zhang *et al.*, 2025), simply extend the 2D grid-based traversal for images to a 3D grid. As mentioned before, this naive strategy suffers from two issues: 1) the 3D grid is view dependent, thus rotating the point cloud or moving the camera yields a different traversal order; 2) unrelated patches may be adjacent in 3D space, hence can be traversed subsequently.

To address these problems, we define a traversal order based on the Laplacian spectrum of the patch-connectivity graph.

In this graph, each node corresponds to a patch and the weighted adjacency matrix W is defined using the Euclidean distance between patch centers. For patches i and j defined by center points p_{s_i} and p_{s_j}, we add an edge (i, j) if p_{s_j} is among the K nearest neighbors of p_{s_i} or vice-versa. The weight of this edge is computed using a Gaussian kernel: $W_{ij} = \exp\left(-\|p_{s_i} - p_{s_j}\|_2^2/\sigma\right)$ where σ is a hyperparameter controlling the kernel width.

Following Section 3.4.1, we compute the s first non-constant eigenvectors of the Random Walk Laplacian L_{rw}. This can be achieved efficiently using an iterative method like the Arnoldi algorithm (Golub & Van Loan, 2013) by exploiting the following facts: 1) matrix W is very sparse, and 2) only the first few eigenvectors need to be computed. Eigenvector $v^{(k)} \in \mathbb{R}^{N_c}$, $k \in \{1, \ldots, s\}$, assigns an eigenfunction value $v_i^{(k)}$ to each patch i. In each Mamba block of our model, we perform two separate traversals of tokens (each token corresponds to an input patch) for *every* eigenvector: a forward traversal by increasing value of $v_i^{(k)}$ and a reserve traversal by decreasing value of $v_i^{(k)}$. At the end of the block, we concatenate the tokens from the $s \times 2$ traversals. The visual effect of each traversal is illustrated in Figure 3.1(b) and Figure 3.3(a).

Canonicalization of spectrum. Although the spectrum of L_{rw} forms an isometry-invariant representation of the surface manifold, this representation may be impacted by two sources of ambiguity: 1) the sign of eigenvectors is undetermined (i.e., if $v^{(k)}$ is an eigenvector, then so is $-v^{(k)}$), and 2) the order of eigenvectors with similar eigenvalues may vary. We address these two sources of ambiguity with the following canonicalization procedure. For the first one, we flip the sign of an eigenvector $v^{(k)}$ (i.e., $v^{(k)} := -v^{(k)}$) if its first element is negative (i.e., $v_1^{(k)} < 0$). To handle the second ambiguity, we first sort the eigenvectors by non-decreasing eigenvalue. We deal with eigenvalues having a mutliplicity greater than one by finding pairs of consecutive eigenvectors $v^{(k)}$, $v^{(k+1)}$ with near-identical eigenvalues (i.e., $|\lambda_k - \lambda_{k+1}| \leq \epsilon$). For such pairs, we flip the order if $v_1^{(k)} > v_1^{(k+1)}$. This reordering process is repeated until no further change occurs.

3.4.4 Hierarchical Local Traversing (HLT) for Segmentation

While effective for classification tasks, the SAST strategy considering each eigenvector in a *separate* traversal may not capture the precise relationship between patches needed for segmentation. To address this issue, we introduce a Hierarchical Local Traversal (HLT) strategy that considers the full spectrum (all s non-constant eigenvectors) simultaneously.

Our strategy is inspired by the recursive binary partitioning technique of normalized cuts (Shi & Malik, 2000). Starting from the canonicalized spectrum (see Section 3.4.3), tokens are first split based on the first eigenvector $v^{(1)}$, by comparing their corresponding value in $v^{(1)}$ with the mean value $\overline{v}^{(1)}$. This yields a binary partition of tokens $b_i^{(1)} = \mathbf{1}\big(v_i^{(1)} \geq \overline{v}^{(1)}\big) \in \{0, 1\}$ where $\mathbf{1}$ is the indicator function. Each subgroup is then divided based on the mean of the second eigenvector $v^{(2)}$, and so on for other eigenvectors. This partitioning process can be seen as building a binary tree, where each level corresponds to a different eigenvector and leaf nodes i are uniquely identified by the sequence of bits $b_i = [b_i^{(1)}, \ldots, b_i^{(s)}]$ on the path from the root to the leaf. Our HLT method traverses groups of leaf nodes (groups of tokens) sequentially based on the lexicographic order of their binary code (e.g., [0000], [0001], [0010], [0011], ... in the case of four eigenvectors). For convenience, we convert binary codes b_i to a non-negative integer q_i (e.g., bin2Int([0011]) = 3) and define two traversal orders, by increasing or decreasing values of q_i.

For s eigenvectors, the HLT strategy described above divides tokens into 2^s segments which are traversed sequentially. In the best case scenario, $\lceil \log_2(N_c) \rceil$ eigenvectors are thus needed to split tokens into individual segments. However, it may happen that multiple tokens fall in the same segment, especially when using fewer eigenvectors. In such case, one can further sort tokens *within* each segment, for example, using the values of the first eigenvector (i.e., $v^{(1)}$). In our implementation, we simply sort these tokens randomly to add stochasticity in the training. This section is illustrated in Figure 4.1 (e), and Figure 3.3 (b).

As shown in Figure 3.3, the first Laplacian eigenvectors encode high-level spatial relations (e.g., bottom *vs.* top, left *vs.* right, torso *vs.* limbs, etc.). In the SAST, because these eigenvectors are

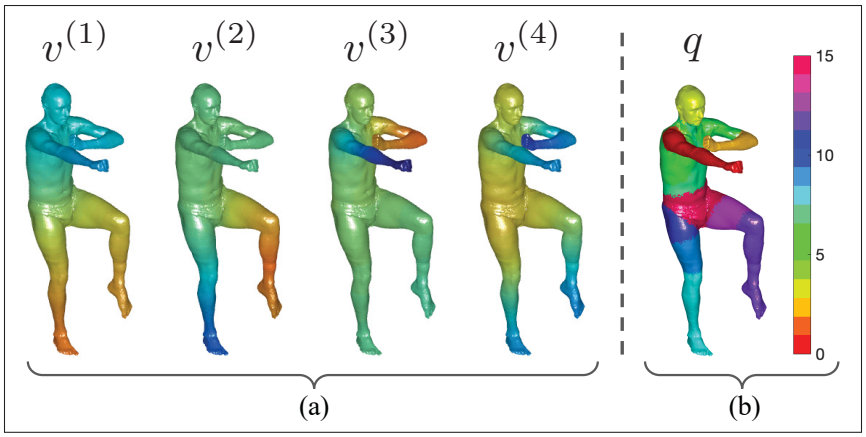

Figure 3.3 (a) Visualization of the four non-constant smallest Laplacian eigenvectors ($v^{(k)}$, $k = 1, \ldots, 4$) and (b) the discrete partitioning (q) of our HLT strategy combining the information of all four eigenvectors. Note: we assumed that patches contain a single point for better visualization

used in *separate* traversals, the network may not be able to differentiate specific regions/parts of the point cloud. In contrast, our HLT strategy can capture such specific parts (e.g., head, left or right arm/thigh/calf, pelvis, etc.).

3.4.5 Traverse-Aware Repositioning (TAR) for Masked Autoencoders

Following state-of-art Transformers for point cloud processing, such as Point-MAE (Pang *et al.*, 2022) and Point-M2AE (Zhang *et al.*, 2022b), our method leverages self-supervised pretraining based on MAE to boost performance. In Transformer-based approaches, the learnable tokens of masked patches can be inserted in any position of the sequence (typically at the end) as the self-attention mechanism can still attend to all tokens irrespective of their positions. However, this approach presents a significant problem in Mamba networks which are sensitive to the traversal order of tokens. We handle this problem via a TAR strategy that improves the placement of learnable tokens in MAE within Mamba networks. Specifically, we restore the learnable tokens to their original positions rather than appending them at the end of the sequence. This ensures that the essential order of tokens is maintained, preserving spatial adjacency and enhancing learning effectiveness within Mamba networks.

The proposed TAR strategy selects an arbitrary traversal order and randomly masks a subset of N_m tokens with the same masking ratio as the transformer-based MAEs. These tokens are then removed from the sequence, and their positions are recorded. Afterwards, the remaining (visible) tokens are fed to the encoder that outputs their representation. Before reconstructing the point cloud using the decoder, we reinsert the learnable tokens in the sequence at their recorded position. The set of masked patches is used for other traversal orders. This procedure can be seen in Figure 3.2 (f). Following previous work, we measure the reconstruction error for masked patches using the Chamfer distance:

$$\mathcal{L}_{\text{rec}} = \frac{1}{N_m} \sum_{i=1}^{N_m} \text{Chamfer}(\mathcal{S}_i, \hat{\mathcal{S}}_i), \tag{3.5}$$

where $\mathcal{S}_i \in \mathbb{R}^{N_n \times 3}$ is the set of points forming the i-th masked patch and $\hat{\mathcal{S}}_i$ the reconstructed output for these points. The Chamfer distance between two sets of points $\mathcal{S}$ and $\hat{\mathcal{S}}$ is defined as

$$\begin{aligned} \text{Chamfer}(\mathcal{S}, \mathcal{S}') = & \sum_{p \in \mathcal{S}} \min_{p' \in \mathcal{S}'} \|p - p'\|_2^2 \\ & + \sum_{p' \in \mathcal{S}'} \min_{p \in \mathcal{S}} \|p - p'\|_2^2. \end{aligned} \tag{3.6}$$

3.5 Experiments

Several experiments are conducted to evaluate the proposed method. First, we pretrain the Point-Mamba network using our techniques on the ShapeNet (Chang *et al.*, 2015) training dataset. We then assess the performance of these pretrained models across a variety of standard benchmarks, including object classification, few-shot learning, and segmentation. Additionally, we train the model from scratch on downstream datasets to demonstrate the robustness and versatility of our method. To have a fair comparison, we adopt the masking ratio (60%) that was used in the Point-Mamba model. Moreover, a comprehensive analysis of the computational

Table 3.1 Object classification on ScanObjectNN (Uy *et al.*, 2019b). Accuracy (%) is reported. † indicates that this method was fine-tuned without rotation augmentation

Methods	Backbone	Param. (M)	FLOPs (G)	OBJ-BG	OBJ-ONLY	PB-T50-RS	ModelNet40
Training from scratch							
PointNet (Qi *et al.*, 2017a)	-	3.5	0.5	73.3	79.2	68.0	89.2
PointNet++ (Qi *et al.*, 2017b)	-	1.5	1.7	82.3	84.3	77.9	90.7
DGCNN (Wang *et al.*, 2019)	-	1.8	2.4	82.8	86.2	78.1	92.9
PointNeXt (Qian *et al.*, 2022)	-	1.4	1.6	-	-	87.7	92.9
PointMLP (Ma, Qin, You, Ran & Fu, 2022a)	-	13.2	31.4	-	-	85.4	-
ADS (Hong, Chou & Liu, 2023)	-	-	-	-	-	87.5	-
Transformer (Yu *et al.*, 2022)	Transformer	22.1	4.8	79.86	80.55	77.24	-
Point-MAE (Pang *et al.*, 2022)	Transformer	22.1	4.8	86.75	86.92	80.78	92.3
PointMamba (Liang *et al.*, 2024)	Mamba	12.3	3.6	90.87	90.18	85.60	92.4
Ours	Mamba	12.3	3.6	**92.25**	**91.39**	**87.30**	**92.7**
Training from pretrained							
Transformer (Yu *et al.*, 2022)	Transformer	22.1	4.8	79.86	80.55	77.24	92.1
Transformer-OcCo (Yu *et al.*, 2022)	Transformer	-	-	84.85	85.54	78.79	-
Point-BERT (Yu *et al.*, 2022)	Transformer	22.1	4.8	87.43	88.12	83.07	92.7
Point-M2AE† (Zhang *et al.*, 2022b)	Transformer	-	-	91.22	88.81	86.43	94.0
Point-MAE (Pang *et al.*, 2022)	Transformer	22.1	4.8	92.77	91.22	89.04	93.2
PointMamba (Liang *et al.*, 2024)	Mamba	12.3	3.6	93.29	91.56	88.17	92.8
PCM (Zhang *et al.*, 2025)	Mamba	12.3	3.6	-	-	86.9	-
Ours	Mamba	12.3	3.6	**94.32**	**91.91**	**89.10**	**93.4**

efficiency, runtime, and memory usage of our SAST approach is provided in the Supplementary Material.

3.5.1 Pretraining Setup

Following Point-Mamba, we adopt the ShapeNet (Chang *et al.*, 2015) dataset for the pretraining and assess the quality of the 3D representations produced by our approach through a linear evaluation on the ModelNet40 (Wu *et al.*, 2015) dataset. The linear evaluation is performed by a SVM fitted on these features. This classification performance is quantified by the accuracy metric. Augmentation in pretraining is *scale and transform*, while in finetuning it is *rotation*.

3.5.2 Downstream Tasks

Object Classification on Real-World Dataset. To evaluate our method for point clouds, we test it on the ScanObjectNN dataset (Uy, Pham, Hua, Nguyen & Yeung, 2019a), as described in previous studies. The augmentation used during training is random rotation. The results, presented in Table 3.1, show that our strategy significantly improves object classification accuracy

in both training from scratch and fine-tuning scenarios. These findings highlight the effectiveness of our approach in enhancing the model's ability to identify and classify objects across various backgrounds, demonstrating its robustness in complex real-world scenarios.

Object Classification on Clean Objects Dataset. We also evaluated our method on the ModelNet40 (Wu *et al.*, 2015) dataset, following the protocols established in previous works. The augmentation used during training is scale and transform. As shown in Table 3.1, our approach achieves notable enhancements on this challenging dataset compared to both the original Point-Mamba and the Transformer-based Point-MAE. This demonstrates the robustness and effectiveness of our method when applied to the Point-Mamba network.

Few-shot Learning. We conducted few-shot learning experiments on ModelNet40 (Wu *et al.*, 2015) dataset, adhering to the protocols of previous studies (Pang *et al.*, 2022; Zhang *et al.*, 2022b; Liu *et al.*, 2022). The results of our few-shot learning experiments are presented in Table 3.2. Despite the competitive nature of this benchmark, our method demonstrated outstanding performance across all tested scenarios. As shown in Table 3.2, our Mamba-based method achieves results comparable to or exceeding those of transformer-based methods (Point-MAE and Point-M2AE).

Part Segmentation. We evaluated our method's representation learning ability on the ShapeNetPart dataset (Yi *et al.*, 2016) using standard experimental settings (Qi *et al.*, 2017a,b; Yu *et al.*, 2022). Table 3.3 shows that performance gains in prior methods are minimal, highlighting the dataset's challenge. For instance, Point-GPT (Chen *et al.*, 2024) and ACT (Dong *et al.*, 2022), despite being state-of-the-art and complex methods, show only marginal improvements over each other and other state-of-the-art approaches. Similarly, I2P-MAE (Zhang *et al.*, 2023), which combines 3D data with 2D information, shows limited improvement over Point-M2AE. For the "Training from scratch" setting, our method outperforms several state-of-the-art approaches. In the "Training from pretrained" setting, we further demonstrate the effectiveness of HLT strategy compared to SAST in the segmentation task.

Table 3.2 Few-shot classification on ModelNet40. We report the average accuracy (%) and standard deviation (%) of 10 independent experiments. '*' denotes reproduced results. A^{std} represents the average (A) and standard deviation (std), respectively

Method	5-way		10-way	
	10-shot	20-shot	10-shot	20-shot
DGCNN (Wang *et al.*, 2019)	$91.8^{3.7}$	$93.4^{3.2}$	$86.3^{6.2}$	$90.9^{5.1}$
DGCNN + OcCo (Wang *et al.*, 2021b)	$91.9^{3.3}$	$93.9^{3.1}$	$86.4^{5.4}$	$91.3^{4.6}$
Transformer (Yu *et al.*, 2022)	$87.8^{5.2}$	$93.3^{4.3}$	$84.6^{5.5}$	$89.4^{6.3}$
Transf. + OcCo (Yu *et al.*, 2022)	$94.0^{3.6}$	$95.9^{2.3}$	$89.4^{5.1}$	$92.4^{4.6}$
Point-BERT (Yu *et al.*, 2022)	$94.6^{3.1}$	$96.3^{2.7}$	$91.0^{5.4}$	$92.7^{5.1}$
Point-M2AE (Zhang *et al.*, 2022b)	$96.8^{1.8}$	$98.3^{1.4}$	$92.3^{4.5}$	$95.0^{3.0}$
Point-MAE (Pang *et al.*, 2022)	$96.3^{2.5}$	$97.8^{1.8}$	$92.6^{4.1}$	$95.0^{3.0}$
PointMamba* (Liang *et al.*, 2024)	$95.9^{2.1}$	$97.3^{1.9}$	$91.6^{5.3}$	$94.5^{3.5}$
Ours	$\mathbf{96.4}^{\mathbf{2.7}}$	$\mathbf{98.5}^{\mathbf{1.5}}$	$\mathbf{92.0}^{\mathbf{5.1}}$	$\mathbf{95.1}^{\mathbf{3.6}}$

3.5.3 Ablation Studies

In this section, we aim to investigate the effects of different parameters on our method. We will focus on two key aspects: the effect of the number of non-constant smallest eigenvectors and the adjacency matrix used in the SAST strategy, and the analysis of the TAR strategy.

Analysis of Eigenvectors and Graph. One of our ablation studies investigates the impact of the number of non-constant smallest eigenvectors used in the SAST and TAR strategies. As depicted in Figure 3.4 (*Left*), the best performance is achieved when using the four non-constant smallest eigenvectors (blue line) for traversing. When the number of non-constant smallest eigenvectors increases beyond four, the performance drops. This is because the additional eigenvectors are closer to the largest eigenvectors, which are less smooth and do not capture the most significant structural variations effectively. The significantly lower performance of Point-MAE (Pang *et al.*, 2022), Point-Mamba (Liang *et al.*, 2024), and Point-Mamba with random traversing (Random-Mamba) compared to ours underscores the importance of appropriate traversing for Mamba networks.

Table 3.3 Part segmentation on ShapeNetPart (Yi *et al.*, 2016). We report instance-level mIoU (Inst.). HLT denotes Hierarchical Local Traversing

Methods	Backbone	FLOPs (G)	mIoU
Training from scratch			
PointNet (Qi *et al.*, 2017a)	-	-	83.7
PointNet++ (Qi *et al.*, 2017b)	-	-	85.1
DGCNN (Wang *et al.*, 2019)	-	-	85.2
APES (Wu, Zheng, Pfrommer & Beyerer, 2023a)	-	-	85.8
Point-MAE (Pang *et al.*, 2022)	Transformer	15.5	85.7
PointMamba (Liang *et al.*, 2024)	Mamba	14.3	85.8
Ours (HLT)	Mamba	14.3	**85.9**
Training from pretrained			
Transformer (Yu *et al.*, 2022)	Transformer	15.5	85.1
Point-BERT (Yu *et al.*, 2022)	Transformer	15.5	85.6
Point-MAE (Pang *et al.*, 2022)	Transformer	15.5	86.1
Point-M2AE (Zhang *et al.*, 2022b)	Transformer	15.5	86.5
Point-GPT-S (Chen *et al.*, 2024)	Transformer	-	86.2
ACT (Dong *et al.*, 2022)	Transformer	-	86.2
I2P-MAE (Zhang *et al.*, 2023)	Transformer	-	86.8
PointMamba (Liang *et al.*, 2024)	Mamba	14.3	86.0
Ours (SAST)	Mamba	14.3	85.7
Ours (HLT)	Mamba	14.3	**86.1**

Another study examines the impact of the number of nearest neighbors used in creating the adjacency matrix. As shown in Figure 3.4 (*Right*), the best performance (blue line) is achieved with 20 nearest neighbors. The model's accuracy increases as the number of nearest neighbors increases from 10 to 20, reaching its peak at 20 nearest neighbors. Beyond this point, the performance starts to decline. All these experiments are trained and evaluated from scratch on the ScanObjectNN dataset (Uy *et al.*, 2019a) (OBJ-BG) with *scale and transform* augmentation.

Analysis of TAR Strategy. One of the studies examines the impact of the TAR strategy on the model's performance. As shown in Figure 3.5 (*Left*), the accuracy of the model with and without the TAR strategy is plotted against the number of epochs. The model incorporating the TAR strategy (light blue line) demonstrates superior performance compared to the model without

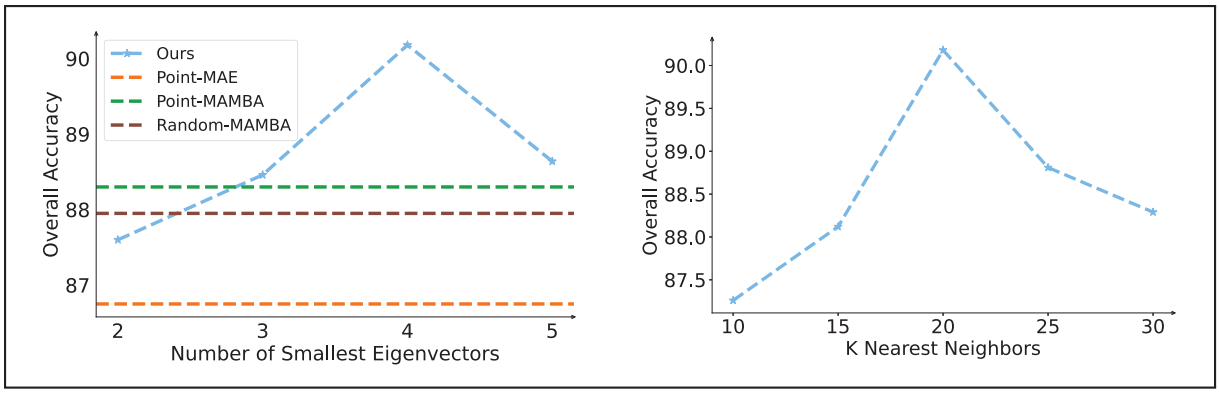

Figure 3.4 Analysis of the number of non-constant smallest eigenvectors and comparison with previous methods (*Left*) and analysis of the number of nearest neighbors K (*Right*)

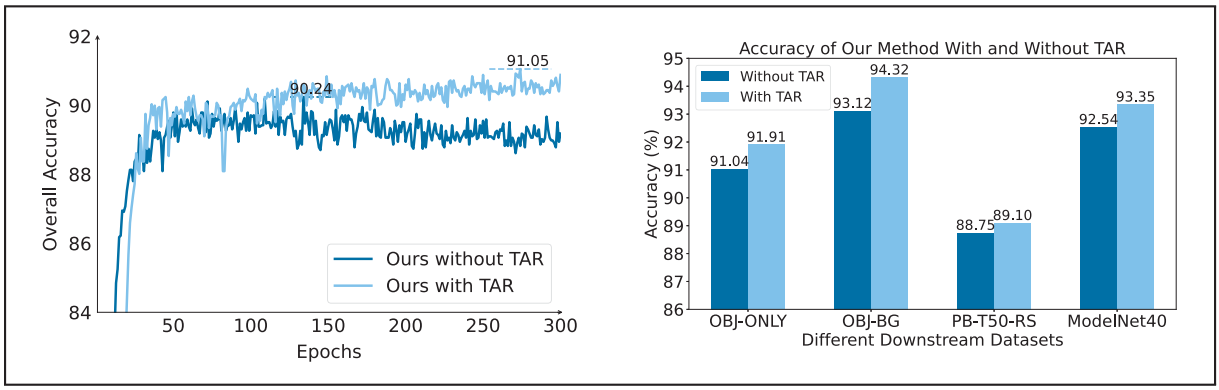

Figure 3.5 The effect of the TAR strategy in the pretraining phase (*Left*) and in fine-tuning (*Right*)

TAR (dark blue line). This figure relates to the pretraining phase on the ShapeNet (Chang *et al.*, 2015) dataset, which is subsequently tested on the ModelNet (Wu *et al.*, 2015) dataset using a SVM.

Additionally, we evaluate the effect of the TAR strategy by fine-tuning pretrained models (with or without TAR) on four downstream tasks, each corresponding to a distinct dataset. As shown in Figure 3.5 (*Right*), the model pretrained with TAR (light blue line) consistently achieves higher accuracy than the model trained without TAR (dark blue line). This demonstrates that the model pretrained with the TAR strategy has learned more meaningful features, leading to better performance in the downstream tasks.

3.6 Conclusion and Limitation

We introduced three strategies to enhance Mamba networks for point cloud data: isometry-invariant token traversal, recursive patch partitioning for segmentation, and improved learnable token placement, each contributing to a robust feature extraction framework. Our methods demonstrate superior performance over state-of-the-art baselines in classification, segmentation, and few-shot tasks. A limitation of our approach lies in the selection of the neighborhood size K, which is treated as a hyperparameter. While tuning K per dataset helps optimize performance, it may limit generalizability across diverse data distributions and require careful adjustment to maintain robustness.

CHAPTER 4

TEST-TIME ADAPTATION IN POINT CLOUDS: LEVERAGING SAMPLING VARIATION WITH WEIGHT AVERAGING

Ali Bahri[a] , Moslem Yazdanpanah[a] , Mehrdad Noori[a] , Sahar Dastani[a] , Milad Cheraghalikhani[a] , David Osowiechi[a] , Farzad Beizaee[a] , Gustavo A.Vargas Hakim[a] , Ismail Ben Ayed[b] , Christian Desrosiers[a]

[a] Department of Information Technologies Engineering, École de Technologie Supérieure
[b] Department of Systems Engineering, École de Technologie Supérieure
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

Paper published in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, February 2025

4.1 Abstract

Test-Time Adaptation (TTA) addresses distribution shifts during testing by adapting a pretrained model without access to source data. In this work, we propose a novel TTA approach for 3D point cloud classification, combining sampling variation with weight averaging. Our method leverages Farthest Point Sampling (FPS) and K-Nearest Neighbors (KNN) to create multiple point cloud representations, adapting the model for each variation using the TENT algorithm. The final model parameters are obtained by averaging the adapted weights, leading to improved robustness against distribution shifts. Extensive experiments on ModelNet40-C, ShapeNet-C, and ScanObjectNN-C datasets, with different backbones (Point-MAE, PointNet, DGCNN), demonstrate that our approach consistently outperforms existing methods while maintaining minimal resource overhead. The proposed method effectively enhances model generalization and stability in challenging real-world conditions. The implementation is available at: https://github.com/AliBahri94/SVWA_TTA.git.

4.2 Introduction

Deep neural networks have recently demonstrated impressive capabilities in classifying 3D point clouds (Qi *et al.*, 2017a,b; Wang *et al.*, 2019; Pang *et al.*, 2022; Zhang *et al.*, 2022b,

2023). However, this success typically relies on the assumption that the test data is drawn from the same distribution as the training data. In real-world applications, this assumption is often invalid. When the test distribution (*target*) differs from the training distribution (*source*), the challenge of distribution shifts arises. In 3D data, such differences can vary widely, as they may be caused by various factors including the type sensor (e.g., RGB-D camera or Lidar), conditions of the environment (e.g., low light for RGB-D camera), and occlusions. This make it impractical to pretrain the network for every possible shift encountered during testing. It is thus essential to develop methods that can adapt to these distribution changes in real-time, and without supervision, during the test phase.

By addressing a more realistic setting where distribution shifts can also occur after training, TTA recently became a focal point for researchers in machine learning and computer vision (Gandelsman *et al.*, 2022; Gao *et al.*, 2023; Iwasawa & Matsuo, 2021; Lim, Kim, Choo & Choi, 2023; Yeo, Kar, Sodagar & Zamir, 2023; Schneider *et al.*, 2020). TTA uses unlabeled test data to adapt a source-pretrained model to distribution shifts occurring in the testing phase. In this paper, we consider the fully-TTA setting where the model is pretrained on source data in a standard supervised manner, without any additional mechanism for adaptation, and the model is only adapted in testing. This setting contrast with Test-Time Training (TTT), where specialized strategies for adaptation are incorporated during source pretraining.

In recent years, various TTA methods have been introduced in the 2D image domain. Key strategies include regularizing the classifier on test data using objective functions based on the prediction entropy (Liang *et al.*, 2020; Zhang *et al.*, 2022a), or updating batch normalization statistics to align with the test data distribution (Mirza, Li & Jabri, 2022). In the context of 3D point cloud classification, TTA is a relatively new and emerging field, with only two approaches proposed for this task: MATE (Mirza *et al.*, 2023) and BFTT3D (Wang *et al.*, 2024b). However, MATE (Mirza *et al.*, 2023) can technically be categorized as Test-Time Training rather than TTA, as it involves using a masked autoencoder during the source pretraining phase. On the other hand, BFTT3D (Wang *et al.*, 2024b) employs a set of source prototypes to adapt to new

target domains. While this prototype memory maintains privacy, it does not fully align with the core principle of TTA which aims to avoid reliance on source data during adaptation.

In this paper, we propose the first fully-TTA strategy for 3D point cloud classification. Our approach is inspired by the concept of seeking flat minima via weight averaging as highlighted in the SWA (Izmailov, Podoprikhin, Garipov, Vetrov & Wilson, 2018) and SWAD (Cha, Kim, Cho & Ceylan, 2021) papers. Focusing on flat minima, our method aims to enhance model robustness against distribution shifts, which are common in real-world scenarios. A key innovation in our approach is the use of sampling variation to drive the adaptation. Specifically, we employ FPS and KNN to generate multiple variations of sampled points within the point cloud, thereby introducing controlled stochasticity during adaptation. By combining the weights obtained using differently-sampled point clouds, the model is steered away from sharp minima which are more prone to overfitting and less robust to distribution shifts.

The iterative adaptation process of our method, which is guided by the prediction entropy minimization strategy of TENT (Wang *et al.*, 2021a), ensures that each variation in the sampling contributes to a broader exploration of the loss landscape. By saving the model weights after each adaptation, and subsequently averaging them, we converge to a flatter and more stable region in the loss landscape. This weight averaging technique, inspired by the SWAD approach, mitigates the impact of noise and outliers within individual samples, leading to a more robust and generalizable model.

We outline the main contributions of our work as follows:

- **Novelty**: Addressing the lack of studies in this field, we introduce the first fully-TTA method specifically designed for 3D point cloud classification. Our method proposes a novel strategy for this challenging task, which combines sampling variation and weight averaging at test time.
- **Robustness**: Our method, which achieves complete TTA without accessing any source data, demonstrates superior efficiency compared to leading approaches like TENT even with very small batch sizes.

- **State-of-the-art performance**: Through an extensive set of experiments involving three datasets modeling a broad range of corruptions and three different backbones for point cloud classification, we show that our method achieves state-of-the-art performance in most test cases.

4.3 Related Work

Test-Time Adaptation. TTA tackles domain adaptation in the more realistic and challenging scenario where the target domain data is unlabeled and we have no access to source domain samples. The primary challenge of this task involves accurately estimating the target domain's distribution and indirectly comparing it to the source domain's characteristics. A typical approach to reduce domain shift when source data is unavailable involves fine-tuning the model using an unsupervised loss based on the target distribution. The TTT algorithm (Sun *et al.*, 2020) enhances the model by updating its parameters in real-time through a self-supervised task applied to the test data. TENT (Wang *et al.*, 2021a) updates the trainable batch normalization parameters during testing by minimizing the entropy of the model's predictions. Source hypothesis transfer (SHOT) (Liang *et al.*, 2020) combines prediction entropy minimization with a diversity regularization prior (maximizing the entropy of the class marginal) to train a robust feature extractor from a pretrained source model. TTT++ (Liu *et al.*, 2021b) incorporates an additional self-supervised branch that utilizes contrastive learning within the source model to aid in adapting to the target domain. TTTFlow (Osowiechi, Rasouli, Creighton & Lakshminarasimhan, 2024) utilizes unsupervised normalizing flows as an alternative to self-supervision for the auxiliary task.

Test-Time Point Cloud Adaptation. The concept of TTA, initially tailored for 2D images, often faces challenges when applied to 3D data, necessitating specialized approaches. So far, very few works have studied this problem in the context of 3D point cloud data. One of the first TTT approaches specifically designed for 3D data, MATE (Mirza *et al.*, 2023), employs a Masked Autoencoder (MAE) reconstruction objective to enhance the robustness of a point cloud classification network to distribution shifts in test data. The Continual Test-Time Domain Adaptation (CTDA) method (Wang *et al.*, 2024a) employs Dynamic Sample Selection

(DSS) to handle noisy pseudo-labels while adapting a pretrained model to new target domains without accessing source data. This approach, enhancing model performance through dynamic thresholding and positive-negative learning, has proven effective in both the 2D image and 3D point cloud domains.

The work in (Shin *et al.*, 2022) presents a multi-modal extension of TTA for 3D semantic segmentation. The proposed method introduces Intra-modal Pseudolabel Generation (Intra-PG) to generate reliable pseudo labels within each modality and Inter-modal Pseudo-label Refinement (Inter-PR) to refine these labels across modalities. Hatem et al. (Hatem *et al.*, 2023a) introduced a TTA technique for point cloud upsampling, leveraging meta-learning to improve model generalization during inference. In point cloud registration, Point-TTA (Hatem *et al.*, 2023b) offers a TTA framework improving model generalization by adapting to each test instance through self-supervised auxiliary tasks. This method allows the model to handle unseen data distributions during testing without prior knowledge. Finally, BFTT3D (Wang *et al.*, 2024b) introduces a backpropagation-free Test-Time Adaptation (TTA) method specifically designed for 3D data, addressing domain shifts with a two-stream architecture that maintains both source and target domain knowledge. However, this approach does not fully align with the core principle of TTA, which aims to avoid reliance on source data during adaptation. Zhang et al. (Zhang *et al.*, 2022a) introduced MEMO, a method that enhances model robustness during test time by applying data augmentations to a single test input and adapting model parameters to minimize the entropy of the averaged output distribution across these augmentations.

Weight Averaging. Weight averaging has become a prominent technique for enhancing the generalization of deep neural networks during training. Stochastic Weight Averaging (SWA) (Izmailov *et al.*, 2018) improves model generalization by averaging weights from different training epochs, promoting smoother optimization and convergence to well-generalized solutions. Building on this, SWAD (Cha *et al.*, 2021) refines the approach by densely sampling weights throughout the training process, further boosting generalization and robustness across tasks. Addressing limitations of traditional WA techniques that average weights post-training, Lookaround (Zhang, Li, Zhou, Liu & Zhao, 2024) introduces a novel optimization strategy

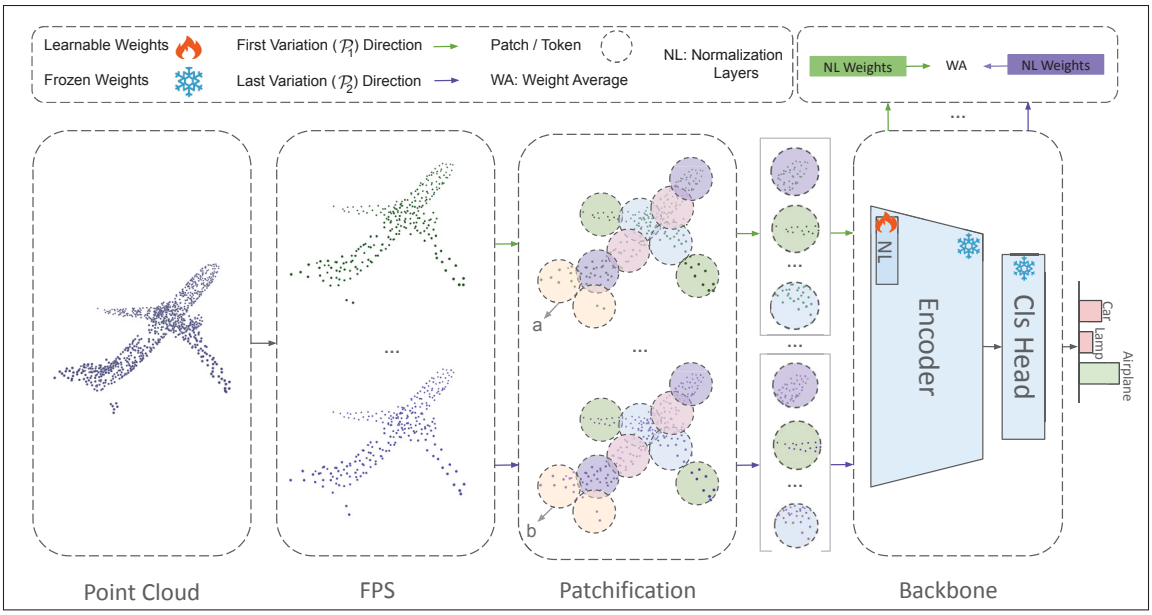

Figure 4.1 Overview of our 3D TTA methodology. First, FPS is applied to generate different samplings from the input point cloud. Patchification is then performed using FPS for patch centers and KNN to form patches (a and b). The Normalization Layer (NL) weights are adapted using the TENT algorithm for each sampling. Finally, weight averaging is applied across all adapted weights to enhance robustness and generalization

integrating diversity into the training process. This method iteratively alternates between two steps: the "around step," which enhances network diversity by training multiple models on differently augmented data, and the "average step" which consolidates these models into a single network.

4.4 Method

We propose a novel Test-Time Adaptation (TTA) strategy tailored for 3D point cloud classification, which focuses on enhancing model robustness against distribution shifts. Our method introduces a dual approach: first, we create diverse perspectives of the input data to address distributional changes by utilizing sampling variation; second, we integrate this variation with a weight averaging technique. This combination operates within a purely test-time framework, eliminating the need for any source data during the adaptation process. The overview of our TTA method for point cloud data is presented in Figure 4.1.

4.4.1 Sampling Variation

We start with a 3D point cloud $P \in \mathbb{R}^{N^p \times 3}$ consisting of N^p points. To address the challenges posed by distribution shifts, we introduce a method that leverages sampling variation during test-time adaptation. This begins by creating patches from the point cloud, where each patch is defined by a center point c_i and its neighboring points. We employ FPS to select a set of center points $\{c_1, c_2, \ldots, c_N\}$. FPS works by iteratively selecting the point in P that is furthest from all previously selected centers:

$$c_i = \arg\max_{p \in P} \min_{c_j \in \{c_1, \ldots, c_{i-1}\}} \|p - c_j\|_2 \tag{4.1}$$

Then, for each center c_i, we use KNN to find its neighboring points, forming a patch of neighboring points P_i as follows:

$$P_i = \left\{ p \in P \mid p \in \mathrm{KNN}(c_i) \right\} \tag{4.2}$$

This process converts the point cloud into a collection of patches $\mathcal{P} \in \mathbb{R}^{N \times K \times 3}$, where N is the number of patches and K is the number of neighbors in each patch (including the center).

During test-time adaptation, we generate multiple versions of $\mathcal{P}$ by varying the selection of centers and neighbors forming $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_V\}$ of size N^V. For Non-Transformer networks, we assume $K = 1$, meaning that each patch consists of only the center point selected by FPS without including any neighbors.

This variation in sampling creates different representations of the same underlying 3D structure and can be interpreted as transformations that slightly modify the local geometric structure of patches while maintaining the overall global structure of the point cloud. This helps the model generalize better by encouraging it to learn robust features that are less sensitive to such variations (e.g., a and b in Figure 4.1).

For each $\mathcal{P}_v$, the model is adapted using the TENT (Wang *et al.*, 2021a) algorithm, where normalization layers' parameters γ and β are updated to minimize the entropy $\mathcal{H}$ of the model's output:

$$\gamma_i^*, \beta_i^* = \arg\min_{\gamma,\beta} \mathbb{E}\left[\mathcal{H}\left(f_\theta\left(\mathcal{P}_v\right)\right)\right]. \tag{4.3}$$

Here, the expectation $\mathbb{E}$ is taken over the distribution of sampling variations $\mathcal{P}_v$, and $\mathcal{H}$ represents the entropy of the model's predictions, which is minimized to encourage confident and stable predictions.

4.4.2 Integrating Sampling Variation with Weight Averaging

For each variation $\mathcal{P}_v$, the model is adapted using the TENT algorithm, as outlined in Equation (4.3). The key innovation here is to combine these adaptations using a refined weight averaging technique inspired by (Cha *et al.*, 2021; Zhang *et al.*, 2024), leading to a more stable and generalizable model.

The concept of **weight averaging** aims to identify a solution in the parameter space that resides within a *flat region* of the loss landscape. Flat minima are characterized by a low loss that remains relatively constant under small variations of the model parameters. Solutions in such regions tend to generalize better to distribution shifts because the model's performance is less sensitive to minor variations or noise in the input data and model parameters.

This integrating sampling variation with weight averaging is related to Robust Risk Minimization (RRM) (Cha *et al.*, 2021). In our method, we follow a similar principle in the context of test-time adaptation. In RRM, the goal is to minimize the worst-case empirical loss within a neighborhood of model parameters. This is formulated as

$$\hat{\mathcal{E}}_{\mathcal{D}}^{\gamma}(\theta) = \max_{\|\Delta\| \leq \gamma} \hat{\mathcal{E}}_{\mathcal{D}}(\theta + \Delta), \tag{4.4}$$

where $\gamma > 0$ defines the neighborhood around the model parameters θ, and Δ represents small perturbations.

While we do not explicitly optimize this loss function, the concept of locating flat minima is analogous to our use of parameter averaging. By averaging the weights of models adapted from different $\mathcal{P}_v$, we effectively find a solution in the weight space that resides in a flatter region of the loss landscape. This improves the model's generalization during test-time adaptation and enhances its robustness to distribution shifts.

In our approach, after adapting the model for each variation $\mathcal{P}_v$, we store the adapted weights θ_v. These weights are then averaged to obtain the final model weights θ_{avg}:

$$\theta_{\text{avg}} = \frac{1}{N^V} \sum_{v=1}^{V} \theta_v \tag{4.5}$$

This averaging process provides two key benefits:

1. **Flat Minima:** Averaging the weights from different adapted models helps to locate a point in the weight space that is situated at the intersection of several flat regions. This approach reduces the model's sensitivity to specific input data configurations, thereby enhancing its robustness against distribution shifts.
2. **Error Reduction:** The averaging process mitigates the influence of errors or noise that may be present in individual model adaptations. By smoothing out fluctuations in different sub-samples, the average model becomes more stable.

4.5 Experiments

In this section, we conduct a comprehensive evaluation of our proposed method across multiple 3D point cloud datasets, focusing on both Transformer-based and non-Transformer-based backbones. To thoroughly assess the robustness and generalization capabilities of our approach, we perform experiments on three benchmark datasets: *ModelNet40-C*, *ShapeNet-C*, and *ScanObjectNN-C*. These datasets encompass a range of real-world challenges, including varying levels of corruption and noise, allowing us to demonstrate the effectiveness of our method in diverse and complex scenarios.

4.5.1 Implementation Details

During the TTA phase, we utilized multiple backbones, Point-MAE, PointNet, and DGCNN, as the source models. Each one is independently trained on its corresponding clean dataset to evaluate the robustness of our method. Pretrained backbones were adapted using our proposed approach. For all backbones, we used the AdamW optimizer with a learning rate of 0.001, consistent with the base learning rate of the TENT algorithm. Detailed information on the settings and hyperparameters are provided in the Supplementary Material. Additionally, **Resource Overhead – Time** and **Resource Overhead – Memory** related to our method are discussed in detail in the Supplementary Material.

For the Transformer-based Point-MAE backbone, we explored two distinct adaptation strategies. In the first approach, only the Batch Normalization (BN) layers were adapted, following the standard TENT adaptation approach. In the second approach, we adapted both Layer Normalization (LN) and Batch Normalization (BN) layers. This configuration was then compared with the TENT algorithm, which similarly adapts both LN and BN layers. The results of both approaches are compared in detail in the Supplementary Material.

We only adapted the BN layers for the non-transformer-based PointNet and DGCNN backbones since these architectures do not LN layers. The input point cloud size was set to 1024 points for all experiments. All experiments were conducted using a single NVIDIA A6000 GPU, ensuring consistency across all tested configurations.

4.5.2 Datasets

ModelNet-40C. ModelNet-40C (Sun *et al.*, 2022) is a robustness benchmark for point cloud classification, designed to assess how well architectures can handle real-world distribution shifts. It introduces 15 common corruption types to the original ModelNet-40 test set, categorized into three groups: transformation, noise, and density. These corruptions simulate real-world issues, such as sensor faults and noise in LiDAR scans, providing a realistic challenge for evaluating model performance under varying conditions.

Table 4.1 Top-1 Classification Accuracy (%) for all distribution shifts in the ModelNet-40C dataset. ∗ and † are explained in Section 4.5.3

	Method	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	occlsion	lidar	Mean
Point-MAE	Source-Only	66.6	59.1	7.2	31.8	74.6	67.7	69.8	59.3	75.1	74.4	38.0	53.7	70.0	38.6	23.4	53.9
	DUA* (Mirza *et al.*, 2022)	65.0	58.5	14.7	48.5	68.8	62.8	63.2	62.1	66.2	68.8	46.2	53.8	64.7	41.2	36.5	54.7
	TTT-Rot* (Sun *et al.*, 2020)	61.3	58.3	34.5	48.9	66.7	63.6	63.9	59.8	68.6	55.2	27.3	54.6	64.0	40.0	29.1	53.0
	T3A* (Iwasawa & Matsuo, 2021)	64.1	62.3	33.4	65.0	75.4	63.2	66.7	57.4	63.0	72.7	32.8	54.4	67.7	39.1	18.3	55.7
	SHOT (Liang *et al.*, 2020)	76.9	71.2	21.4	63.6	79.0	71.2	73.3	75.0	79.6	76.6	55.7	71.8	71.2	46.1	44.7	65.1
	MATE* (Mirza *et al.*, 2023)	75.0	71.1	27.5	67.5	78.7	69.5	72.0	79.1	84.5	75.4	44.4	73.6	72.9	39.7	34.2	64.3
	PL (Lee *et al.*, 2013)	81.9	78.1	25.4	69.3	77.0	77.4	78.4	83.7	86.8	81.4	63.0	82.3	78.1	52.3	49.5	71.0
	MEMO (Zhang *et al.*, 2022a)	83.8	81.0	30.8	68.7	83.6	75.3	77.3	83.0	85.3	74.6	58.0	74.9	67.6	50.3	47.3	69.4
	TENT (Wang *et al.*, 2021a)	82.0	78.6	26.7	71.0	78.2	78.4	79.9	84.6	87.0	82.7	65.4	83.5	79.6	52.3	51.2	72.1
	Ours	**85.0**	**83.9**	**33.0**	**74.6**	**87.0**	**80.9**	**82.3**	**85.1**	**88.0**	**82.7**	**66.9**	**84.0**	**80.5**	**56.2**	**55.3**	**75.0**
PointNet	Source-Only	43.2	82.8	4.1	41.6	43.8	44.1	44.6	84.3	86.2	33.6	24.0	83.6	37.9	22.6	21.8	46.5
	LAME (Boudiaf, Mueller, Ben Ayed & Bertinetto, 2022)	23.4	15.3	4.0	4.5	32.2	6.6	8.8	40.1	65.2	3.5	2.5	31.48	7.5	4.0	4.0	16.8
	SHOT (Liang *et al.*, 2020)	71.8	83.5	11.9	66.4	71.2	61.3	61.6	82.7	83.7	41.2	29.1	80.5	49.0	28.7	28.7	56.8
	DUA (Mirza *et al.*, 2022)	67.5	84.7	8.3	61.8	67.1	60.2	60.7	84.6	86.3	43.0	31.1	84.2	51.5	32.9	35.6	57.3
	PL (Lee *et al.*, 2013)	72.2	83.5	12.8	67.8	73.3	64.0	66.5	84.2	85.8	46.5	35.7	84.1	56.4	30.0	31.1	59.6
	BFTT3D † (Wang *et al.*, 2024b)	85.5	81.7	19.3	68.1	85.2	71.0	72.8	87.2	89.5	60.3	31.4	85.3	66.0	45.9	44.1	66.2
	MEMO (Zhang *et al.*, 2022a)	72.5	83.4	11.0	66.2	72.4	63.2	65.5	83.6	85.4	47.0	35.2	80.4	52.3	33.0	26.6	58.5
	TENT (Wang *et al.*, 2021a)	72.4	83.5	**13.1**	68.4	**74.1**	65.0	68.3	84.1	86.3	48.1	37.6	**84.5**	57.8	30.4	32.9	60.4
	Ours	**73.1**	**84.0**	12.1	**69.0**	73.5	**66.1**	**68.6**	**84.5**	**86.1**	**49.2**	**40.8**	84.0	**59.7**	**34.8**	**34.2**	**61.3**
DGCNN	Source-Only	68.1	74.9	13.6	55.1	74.7	74.7	75.3	51.5	82.2	76.7	57.2	57.6	76.4	32.2	12.0	58.8
	SHOT (Liang *et al.*, 2020)	75.3	78.2	29.1	66.6	76.0	69.9	68.9	50.4	68.8	59.4	44.7	42.1	47.1	12.5	7.4	53.1
	DUA (Mirza *et al.*, 2022)	81.6	83.1	39.4	74.0	83.4	81.2	82.1	71.4	85.5	81.2	72.3	74.3	80.5	39.0	25.5	70.3
	PL (Lee *et al.*, 2013)	79.6	81.9	35.4	73.6	79.1	80.0	80.8	78.9	86.8	81.1	72.1	80.0	79.6	38.2	33.8	70.7
	BFTT3D † (Wang *et al.*, 2024b)	80.5	80.0	41.5	77.7	75.4	78.0	79.3	76.0	83.5	81.2	68.5	78.4	78.6	43.7	37.8	70.7
	MEMO (Zhang *et al.*, 2022a)	79.0	80.7	50.1	67.9	78.7	74.1	74.6	75.7	77.8	66.0	57.9	62.6	56.3	30.7	20.5	63.5
	TENT (Wang *et al.*, 2021a)	**80.4**	**82.0**	38.6	**75.2**	80.1	**80.8**	**81.1**	78.7	86.0	**81.4**	74.3	80.8	**80.5**	39.2	34.9	71.6
	Ours	80.1	81.4	**56.8**	73.2	**83.1**	79.1	80.6	**81.9**	**86.6**	79.3	**74.5**	**81.6**	79.0	**51.9**	**43.2**	**74.2**

ShapeNet-C. ShapeNetCore-v2 (Chang *et al.*, 2015) is a large-scale dataset used for point cloud classification, consisting of 51,127 shapes from 55 categories. It is divided into training (70%), validation (10%), and test (20%) sets. To assess model robustness under real-world conditions, (Mirza *et al.*, 2023) applies 15 different types of corruptions to the test set, similar to those in ModelNet-40-C. These corruptions were generated using an open-source implementation provided by (Sun *et al.*, 2022). This modified version of the dataset is referred to as ShapeNet-C.

ScanObjectNN-C. ScanObjectNN (Uy *et al.*, 2019b) is a real-world point cloud classification dataset consisting of 15 categories. It includes 2,309 samples for training and 581 samples for testing. To evaluate model robustness, (Mirza *et al.*, 2023) introduces 15 distinct corruptions to the test set, following the methodology outlined in (Sun *et al.*, 2022). This modified version of this dataset is referred to as ScanObjectNN-C.

Table 4.2 Top-1 Classification Accuracy (%) for all distribution shifts in the ShapeNet-C dataset. $*$ is explained in Section 4.5.3

	Method	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	ocksion	lidar	Mean
Point-MAE	Source-Only	77.4	71.8	8.6	54.4	77.9	75.5	76.0	85.3	76.5	80.5	57.1	85.1	76.0	11.0	7.1	61.3
	DUA* (Mirza *et al.*, 2022)	76.1	70.1	14.3	60.9	76.2	71.6	72.9	80.0	83.8	77.1	57.5	75.0	72.1	11.9	12.1	60.8
	TTT-Rot* (Sun *et al.*, 2020)	74.6	72.4	23.1	59.9	74.9	73.8	75.0	81.4	82.0	69.2	49.1	79.9	72.7	14.0	12.0	60.9
	T3A* (Iwasawa & Matsuo, 2021)	70.0	60.5	6.5	40.7	67.8	67.2	68.5	79.5	79.9	72.7	42.9	79.1	66.8	7.7	5.6	54.4
	SHOT (Liang *et al.*, 2020)	78.6	74.2	10.4	62.3	73.9	68.3	64.9	68.1	53.0	52.5	31.4	54.2	41.2	2.3	1.7	49.1
	MATE* (Mirza *et al.*, 2023)	77.8	74.7	4.3	66.2	78.6	76.3	75.3	86.1	86.6	79.2	56.1	84.1	76.1	12.3	13.1	63.1
	PL (Lee *et al.*, 2013)	80.8	78.4	16.4	71.4	81.2	79.5	79.8	85.3	83.2	81.6	69.4	84.9	79.5	10.7	10.3	66.2
	MEMO (Zhang *et al.*, 2022a)	81.2	74.5	17.5	60.3	63.8	58.9	54.0	50.2	40.4	33.8	25.6	23.4	18.4	16.6	16.4	42.3
	TENT (Wang *et al.*, 2021a)	81.1	78.7	14.3	**71.4**	81.6	79.4	79.7	**85.9**	82.6	**81.8**	70.0	**85.0**	**79.6**	9.9	10.4	66.1
	Ours	**82.6**	**81.0**	**21.2**	71.2	**82.5**	**79.8**	**80.1**	85.2	**84.8**	81.1	**70.4**	83.9	78.9	**11.2**	**11.5**	**67.0**
PointNet	Source-Only	59.9	76.1	9.3	52.8	60.3	55.3	55.0	83.1	82.6	42.9	26.3	83.0	47.8	6.3	5.5	49.8
	SHOT (Liang *et al.*, 2020)	65.5	77.7	6.4	39.2	37.9	27.6	25.6	51.2	39.0	9.0	7.1	39.3	11.8	1.8	2.1	29.4
	DUA (Mirza *et al.*, 2022)	66.7	78.6	12.5	57.1	66.4	59.6	60.9	83.1	82.4	46.1	34.1	83.0	52.7	8.2	9.6	53.4
	PL (Lee *et al.*, 2013)	67.0	78.7	11.8	57.4	67.8	61.5	62.2	83.2	82.8	48.5	37.6	83.0	54.9	8.5	9.0	54.3
	MEMO (Zhang *et al.*, 2022a)	65.9	77.0	**13.9**	42.9	36.4	27.9	21.9	17.8	16.7	17.3	16.6	16.4	16.5	**17.1**	**16.0**	28.0
	TENT (Wang *et al.*, 2021a)	67.2	79.1	12.8	59.2	67.8	62.9	63.6	**83.4**	**82.8**	51.6	40.9	**83.0**	58.2	9.6	9.5	55.5
	Ours	**68.3**	**79.3**	12.4	**61.2**	**70.7**	**65.8**	**66.8**	82.2	82.3	**56.6**	**46.0**	80.9	**60.4**	10.0	10.1	**56.9**
DGCNN	Source-Only	74.7	74.1	30.5	55.9	75.0	77.3	78.1	85.1	81.6	79.2	66.2	84.7	77.4	8.0	6.5	63.6
	SHOT (Liang *et al.*, 2020)	73.1	68.2	14.6	52.2	48.9	43.0	36.3	43.8	27.8	22.4	15.4	25.0	14.6	1.9	1.8	32.6
	DUA (Mirza *et al.*, 2022)	78.1	77.4	24.5	72.8	78.1	79.9	80.4	85.3	82.5	81	73.3	84.6	79.1	9.3	10.5	66.4
	PL (Lee *et al.*, 2013)	78.3	78.2	24.1	73.4	79.3	80.4	81.1	85.5	82.7	81.5	74.5	84.7	80.0	10.1	11.6	67.0
	MEMO (Zhang *et al.*, 2022a)	77.5	69.0	34.0	40.0	27.3	23.9	20.6	18.1	17.2	17.0	16.9	16.7	16.7	**18.1**	**16.8**	28.7
	TENT (Wang *et al.*, 2021a)	78.7	78.7	27.7	**73.5**	**78.6**	**79.8**	**80.2**	**84.4**	**81.8**	**79.9**	**74.6**	**83.7**	**79.2**	10.6	14.2	**67.0**
	Ours	**80.0**	**80.0**	**59.0**	69.5	77.0	74.9	74.7	80.0	78.9	74.9	70.5	77.9	72.4	9.1	12.0	66.0

4.5.3 Main Results

In all result tables, *source only* refers to testing the pretrained model directly on the corrupted dataset without any adaptation. While BFTT3D (Wang *et al.*, 2024b) is included in the tables, its results are not directly comparable to ours because since, as mentioned in the Introduction, this method relies on source data during TTA. Except for the ones marked with $*$, all results are reproduced. Moreover, results marked with † indicate dependence on the source data during TTA.

ModelNet-40C. We evaluated the effectiveness of our method on the ModelNet40-C dataset using three different backbones, Point-MAE, PointNet, and DGCNN, under various corruptions. As reported in Table 4.1, our method consistently outperforms prior approaches across different backbones and corruptions. For the Point-MAE backbone, our method improves performance across all corruption types, with the average accuracy increasing from 72.1% (TENT) and 69.4%

Table 4.3 Top-1 Classification Accuracy (%) for all distribution shifts in the ScanObjectNN-C dataset. $*$ and † are explained in Section 4.5.3

	Method	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Mean
Point-MAE	Source-Only	20.8	32.5	16.7	16.0	22.4	32.2	35.3	68.8	64.9	35.8	28.6	69.7	33.9	9.3	9.1	33.1
	DUA* (Mirza *et al.*, 2022)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	46.0
	TTT-Rot* (Sun *et al.*, 2020)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	46.1
	T3A* (Iwasawa & Matsuo, 2021)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	40.3
	SHOT (Liang *et al.*, 2020)	38.7	53.7	17.0	31.4	37.9	47.3	46.5	71.7	70.7	50.0	45.5	70.3	49.0	9.4	8.8	43.2
	PL (Lee *et al.*, 2013)	38.1	56.0	17.4	31.4	39.6	51.0	53.7	74.6	74.4	54.7	49.0	75.4	56.0	9.0	7.4	45.8
	MATE* (Mirza *et al.*, 2023)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	47.0
	MEMO (Zhang *et al.*, 2022a)	40.4	55.5	18.6	**34.0**	**41.4**	51.8	53.3	73.6	74.6	54.5	46.7	73.0	55.7	8.8	**9.2**	46.1
	TENT (Wang *et al.*, 2021a)	38.1	55.8	16.6	32.4	39.6	51.0	**54.7**	**74.2**	**75.0**	54.5	**50.0**	**74.2**	**56.8**	**8.8**	7.6	45.9
	Ours	**41.0**	**58.8**	**18.7**	33.6	39.6	**51.9**	52.5	72.1	74.2	**55.7**	48.2	74.0	54.3	8.2	9.0	**46.1**
PointNet	Source-Only	20.6	36.1	10.3	18.4	20.6	25.6	28.4	63.2	64.5	27.9	23.1	62.8	27.0	6.9	10.0	29.7
	SHOT (Liang *et al.*, 2020)	42.2	62.3	15.0	33.8	41.6	35.9	35.3	66.0	66.4	36.3	32.8	65.4	40.4	9.0	8.4	39.4
	DUA (Mirza *et al.*, 2022)	28.1	43.3	13.9	21.7	31.2	29.1	34.8	62.9	65.6	30.1	26.0	63.3	31.2	7.6	9.2	33.2
	PL (Lee *et al.*, 2013)	40.6	61.5	16.2	36.5	40.4	35.5	36.1	64.8	65.4	36.3	35.1	66.8	39.6	8.0	9.4	39.5
	BFTT3D † (Wang *et al.*, 2024b)	42.9	60.1	30.0	34.8	44.1	45.4	47.0	78.7	78.5	41.3	31.2	75.9	44.1	15.0	14.5	45.5
	MEMO (Zhang *et al.*, 2022a)	41.6	**62.5**	**17.6**	**36.1**	**41.2**	**37.5**	**37.7**	**67.6**	65.4	39.1	31.8	65.0	39.8	10.4	9.4	40.2
	TENT (Wang *et al.*, 2021a)	40.2	61.5	16.2	35.7	40.8	35.7	37.1	65.4	66.4	35.9	**35.1**	**67.0**	39.6	8.8	**10.1**	39.7
	Ours	**42.0**	61.9	15.8	35.9	41.0	37.1	37.3	67.4	**68.2**	**40.6**	35.0	66.8	**40.2**	**10.5**	9.2	**40.6**
DGCNN	Source-Only	26.7	39.9	27.0	22.5	29.6	41.5	41.3	70.6	60.2	43.4	34.4	70.2	43.0	8.7	9.3	37.9
	SHOT (Liang *et al.*, 2020)	36.8	48.9	28.1	45.8	33.8	50.7	49.6	70.8	65.1	53.3	47.4	67.5	51.7	8.7	8.8	44.5
	DUA (Mirza *et al.*, 2022)	38.4	51.9	30.2	48.4	39.2	54.0	55.2	73.6	70.7	57.8	50.3	74.3	56.9	9.2	10.8	48.1
	PL (Lee *et al.*, 2013)	38.4	53.6	27.6	52.1	39.1	55.0	58.0	73.6	68.2	57.5	53.3	71.3	57.3	9.2	8.1	48.1
	BFTT3D † (Wang *et al.*, 2024b)	42.7	58.2	47.7	56.6	46.6	63.2	65.9	77.5	82.3	67.0	61.5	77.8	65.4	14.5	14.6	56.1
	MEMO (Zhang *et al.*, 2022a)	37.7	52.6	**28.0**	**52.4**	39.2	51.2	54.2	67.2	62.0	48.4	47.9	59.5	50.3	**11.3**	**13.0**	45.0
	TENT (Wang *et al.*, 2021a)	39.1	**55.2**	25.3	50.5	39.7	54.0	59.2	**73.4**	68.4	58.8	54.2	71.9	56.2	10.1	8.3	48.3
	Ours	**39.2**	54.2	27.8	51.6	**43.0**	**58.3**	**59.9**	70.5	**71.2**	**61.4**	**55.4**	**74.1**	**60.6**	10.2	10.6	**49.9**

(MEMO) to 75.0%. This reflects the robustness of our approach in handling distribution shifts by leveraging the proposed sampling variation and weight averaging technique, as discussed in Section 4.4. In the PointNet backbone, our method achieves an overall improvement in mean accuracy from 60.4% (TENT) 58.5 (MEMO) to 61.3%. This improvement is driven by the better handling of difficult corruptions such as *rotation*, *distortion*, and *occlusion*, where our method consistently performs better than TENT. For the DGCNN backbone, our method performs exceptionally well on specific corruptions such as *occlusion* and *lidar*, achieving improvements of 12.7% and 8.3%, respectively, over TENT, and 21.2% and 22.7%, respectively, over MEMO. This leads to a higher overall average, increasing from 71.6% (TENT) and 63.5% (MEMO) to 74.2%. These results highlight the effectiveness of our method in improving robustness under various corruption scenarios, particularly when adapting to real-world data shifts during test-time adaptation.

ShapeNet-C. In Table 4.2, we evaluate our method on the ShapeNet-C dataset across the same three backbones. For the Point-MAE backbone, our method outperforms TENT and MEMO, achieving a mean accuracy of 67.0% compared to TENT's 66.1% and MEMO's 42.3%. Particularly large gains in performance are obtained by our method for the *background* and *dens-inc* corruptions. We observe similar improvements with the PointNet backbone, as our method achieves a mean accuracy of 56.9%, surpassing TENT's score of 55.5% and MEMO's score of 28.0%. Our model shows significant performance gains in difficult corruptions like *upsampling*, *rbf*, *rotation*, *shear*, and *rbf-inv*. These results demonstrate the effectiveness of our method in improving robustness across various corruption types, ensuring the model generalizes better to corrupted data during test-time adaptation. For the DGCNN backbone, our method outperforms MEMO, although its performance is somewhat comparable to the TENT method.

ScanObjectNN-C. As a final evaluation, we assessed our method on the ScanObjectNN-C dataset, using three backbones. As presented in Table 4.3, our method achieved notable improvements in performance across multiple corruptions and backbones. For the Point-MAE backbone, our approach improves the mean accuracy to 46.1%, surpassing TENT's 45.9%, with notable gains in corruptions such as *Gaussian noise* (58.8%) and *uniform noise* (41.0%). On this dataset with this backbone, our performance is equal to MEMO. Likewise, our method improved the mean accuracy to 40.6% for the PointNet backbone, outperforming TENT's 39.7%. This improvement is particularly evident in challenging corruptions such as *occlusion* (10.5%) and *shear* (40.6%). On this dataset and with this backbone, MEMO performs comparably to our method. In the DGCNN backbone, our approach yielded a mean accuracy of 49.9%, outperforming TENT's 48.3% and MEMO's 45.0%. Our method showed strong improvements like in *upsample* (43.0%), *rbf* (58.3%), and *shear* (61.4%), further demonstrating its robustness in handling distribution shifts across the dataset.

4.5.4 Ablation Study

In this section, we conduct a comprehensive ablation study on the ModelNet40-C dataset using the Point-MAE backbone to examine the impact of various factors on our model's performance

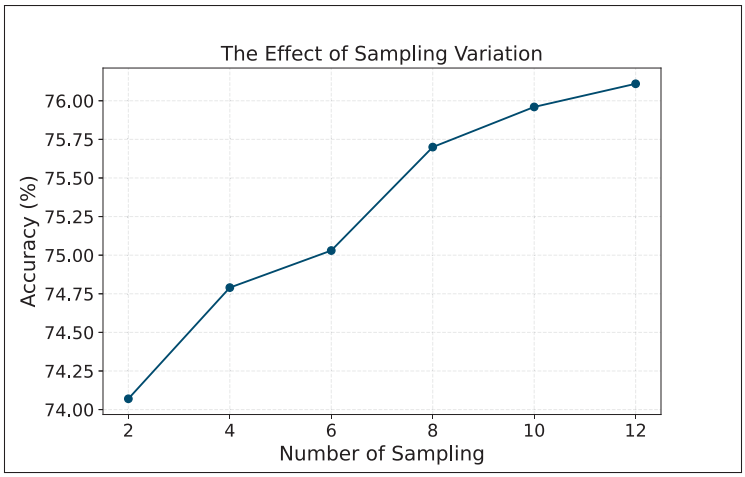

Figure 4.2 Impact of different N^V on model accuracy

in TTA. Specifically, we evaluate four aspects: *Sampling Variation*, *Number of Iterations*, *Batch Size*, and *Types of Augmentation*. For consistency, all experiments were conducted with a learning rate of 0.001.

Sampling Variation. We first explore the effect of increasing the number of sampling variations N^V. Figure 4.2 shows the steady rise in model accuracy as we increase the number of sampling variations. This validates our idea of combining weight averaging and sampling variation to enhance the model's robustness. As N^V increases from 2 to 12, the accuracy improves approximately from 74.0% to 76.0%. This demonstrates that leveraging more diverse sampling during TTA enables better adaptation to distribution shifts. Based on these results, we selected $N^V = 6$ for all subsequent experiments to balance computational efficiency and performance improvement. For this experiment, the batch size and iteration are set to 128 and 1, respectively.

Number of Iterations. As shown in Figure 4.4, the accuracy improves as the number of iterations increases. Our method and TENT show performance gains with more iterations, but our method consistently outperforms TENT at every stage of the iteration process. Starting with a gap at the first iteration, our approach maintains a steady improvement, surpassing TENT at each level. Based on these results, we opted to use just one iteration in subsequent experiments

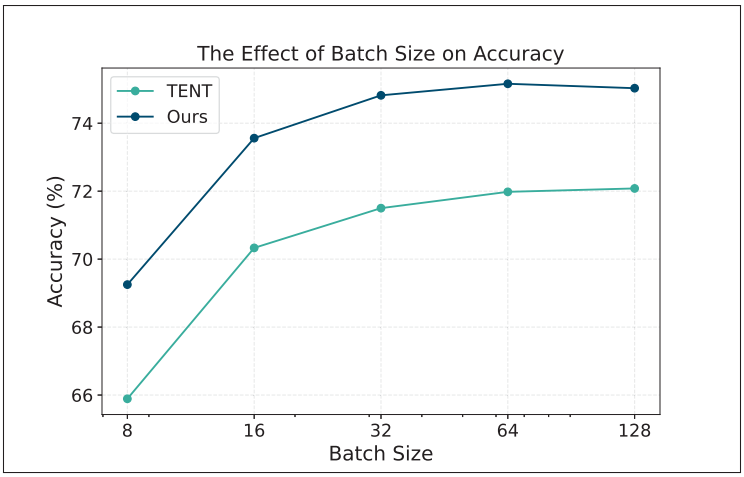

Figure 4.3 Impact of batch size on accuracy for two methods

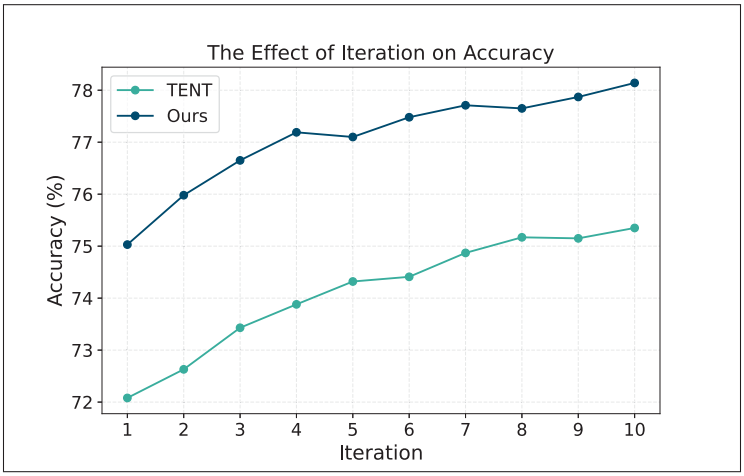

Figure 4.4 Impact of iteration on accuracy for two methods

to prioritize faster model adaptation while still achieving a competitive accuracy boost. For this experiment, the batch size is set to 128.

Batch Size. As illustrated in Figure 4.3, our method consistently outperforms TENT across all batch sizes. Notably, the improvement remains steady even at lower batch sizes, such as 8 and 16, where our method achieves approximately 3% higher accuracy. This demonstrates that our approach is robust across different batch settings, making it effective even in scenarios with

limited data. Based on these results, we selected a batch size of 128. For this experiment, the number of iterations is set to 1.

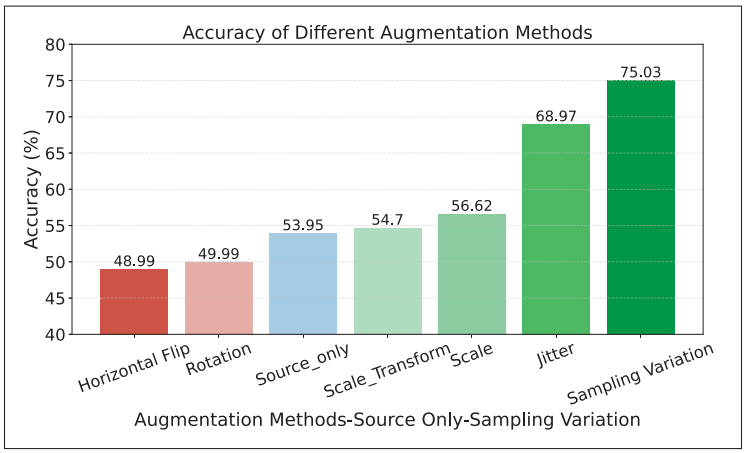

Figure 4.5 Comparison between sampling variation and different augmentations

Types of Augmentation. It can be argued that the effect of sampling variation is akin to applying various augmentations to the data samples. Hence, we have assessed the integration of different augmentations into our weight averaging process. As shown in Figure 4.5, we applied augmentations such as *Horizontal Flip*, *Rotation*, *Scale*, *Scale Transform*, *Jitter*, and *Sampling Variation* to investigate their impact.

Suprisingly, certain augmentations like *Horizontal Flip* and *Rotation* worsened the model's performance, reducing accuracy even below the *source-only* model. However, augmentations like *Jitter* and especially *Sampling Variation* resulted in improvements, with *Sampling Variation* outperforming all other strategies. This highlights that our approach, which leverages sampling variation combined with weight averaging, is highly effective for boosting performance compared to data augmentations. For this experiment, the batch size and iteration are set to 128 and 1, respectively.

4.6 Conclusion

In this paper, we introduced a novel TTA framework combining weight averaging with sampling variation to enhance model robustness against distribution shifts in 3D point cloud data. Evaluated on multiple backbones and datasets, our method outperforms existing approaches such as TENT, particularly under challenging corruptions. Through ablation studies, we demonstrated the effectiveness of our approach across different batch sizes, iterations, and sampling variations. Our method offers a robust, efficient solution for improving generalization in 3D point cloud classification.

CHAPTER 5

SMART-PC: SKELETAL MODEL ADAPTATION FOR ROBUST TEST-TIME TRAINING IN POINT CLOUDS

Ali Bahri[a] , Moslem Yazdanpanah[a] , Sahar Dastani[a] , Mehrdad Noori[a] , Gustavo A.Vargas Hakim[a] , David Osowiechi[a] , Farzad Beizaee[a] , Ismail Ben Ayed[b] , Christian Desrosiers[a]

[a] Department of Information Technologies Engineering, École de Technologie Supérieure
[b] Department of Systems Engineering, École de Technologie Supérieure
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

Paper published in *International Conference on Machine Learning (ICML)*, May 2025

5.1 Abstract

Test-Time Training (TTT) has emerged as a promising solution to address distribution shifts in 3D point cloud classification. However, existing methods often rely on computationally expensive backpropagation during adaptation, limiting their applicability in real-world, time-sensitive scenarios. In this paper, we introduce SMART-PC, a skeleton-based framework that enhances resilience to corruptions by leveraging the geometric structure of 3D point clouds. During pre-training, our method predicts skeletal representations, enabling the model to extract robust and meaningful geometric features that are less sensitive to corruptions, thereby improving adaptability to test-time distribution shifts. Unlike prior approaches, SMART-PC achieves real-time adaptation by eliminating backpropagation and updating only BatchNorm statistics, resulting in a lightweight and efficient framework capable of achieving high frame-per-second rates while maintaining superior classification performance. Extensive experiments on benchmark datasets, including ModelNet40-C, ShapeNet-C, and ScanObjectNN-C, demonstrate that SMART-PC achieves state-of-the-art results, outperforming existing methods such as MATE in terms of both accuracy and computational efficiency. The implementation is available at: https://github.com/AliBahri94/SMART-PC.

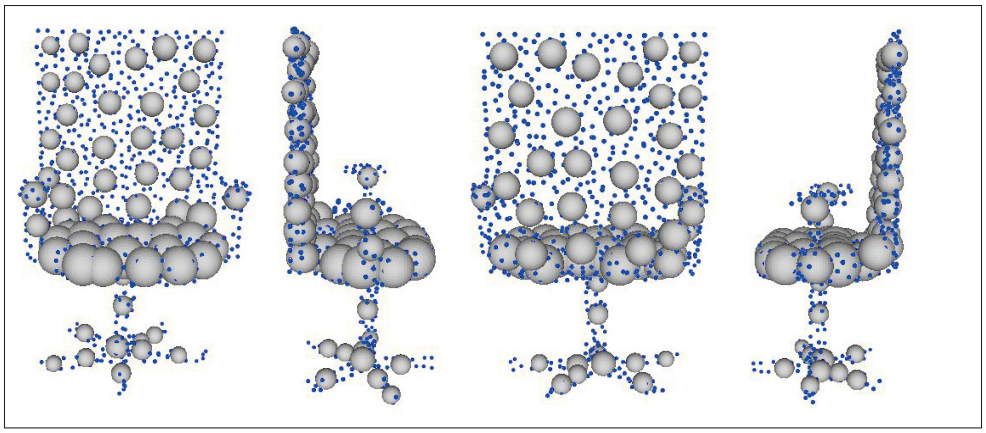

Figure 5.1 The blue points represent the sampled points on the surface of spheres created using the skeletal points as centers and their corresponding radii. Each sphere illustrates the local geometric structure defined by the skeletal representation

5.2 Introduction

Recent advancements in deep learning have significantly improved the classification of 3D point clouds (Pang *et al.*, 2022; Zhang *et al.*, 2022b; Bahri *et al.*, 2025c; Liang *et al.*, 2024; Bahri *et al.*, 2024c). However, these models often assume that the test data distribution matches the training distribution, an assumption that rarely holds in real-world scenarios. Distribution shifts, caused by factors such as environmental conditions or sensor inaccuracies in LiDAR data, can introduce distortions that degrade model performance (Ren, Pan & Liu, 2022a; Sun *et al.*, 2022). This vulnerability is particularly challenging in critical domains like autonomous navigation and robotics. As pre-training for all potential scenarios is impractical, robust methods for unsupervised test-time adaptation are essential to address these shifts effectively.

In TTT for classification, an additional technique is employed during pre-training with the source dataset, which can be leveraged during the adaptation phase. During adaptation, the network utilizes unlabeled target data to dynamically adjust to shifts in data distributions at test time. Recent TTT approaches in the 2D image domain have explored various strategies, such as entropy-based regularization, updating BatchNorm statistics, and self-supervised tasks (Liang *et al.*, 2020; Wang *et al.*, 2021a; Mirza *et al.*, 2022; Sun *et al.*, 2020). However, these methods

struggle when directly applied to 3D point clouds, highlighting the need for TTT techniques specifically designed for 3D data. In the 3D domain, a notable work is the MATE framework (Mirza *et al.*, 2023), which addresses the problem of TTT for 3D point cloud classification. MATE introduces a 3D-specific approach that leverages the self-supervised paradigm, where a network is adapted to out-of-distribution (OOD) target data by solving a self-supervised task. MATE utilizes a masked autoencoder (MAE), where the network reconstructs a point cloud from its partially removed data. This reconstruction process enables adaptation, enhancing robustness to distribution shifts during test-time.

While MATE represents a significant advancement in TTT for 3D point clouds, it has critical limitations. MATE focuses on reconstructing points on the surface of objects, making it highly sensitive to surface-level corruptions, such as noise, which frequently occur in real-world scenarios. This focus on surface points makes the features learned by MATE highly sensitive to corruptions, resulting in reduced resilience to distribution shifts. Additionally, MATE heavily depends on updating all network parameters during adaptation. This approach reduces the model's speed, making it less practical for real-time applications due to its lower Frames/Second.

In contrast, our method overcomes these limitations by introducing a skeleton-based framework for TTT. By capturing the essential geometric structure of 3D point clouds, skeletons enable the model to learn powerful and meaningful features that are inherently less sensitive to corruptions. This is achieved through the abstraction property of skeletal representations (Lin *et al.*, 2021), which simplifies the shape into a compact form, filtering out high-frequency noise and local distortions. As a result, the model focuses on the underlying geometric structure rather than surface-level variations, enhancing robustness to distribution shifts. By leveraging these robust features, our approach eliminates the need for backpropagation during adaptation, significantly improving speed. Instead, following prior studies (Nado *et al.*, 2020; Li, Wang, Shi, Hou & Liu, 2018b), we update only the BatchNorm statistics, which effectively adapts the model to distribution shifts without modifying its learned weights. This efficiency demonstrates that skeleton-based pre-training inherently equips the model with features that are more resilient to corruption compared to MATE, providing a lightweight and effective solution for TTT. As

shown in Figure 5.1, the skeletal representation captures the geometric structure of point clouds by abstracting them into compact and meaningful features. Our method not only overcomes the limitations of MATE but also introduces a paradigm shift in TTT by emphasizing the importance of meaningful feature extraction during pre-training, ensuring robustness and adaptability with minimal computational effort. Our main contributions are:

- We propose SMART-PC, a skeleton-based framework for TTT that enhances robustness by extracting geometric features that are less sensitive to distribution shifts.
- Our method achieves test-time adaptation without backpropagation, relying solely on updating BatchNorm statistics, making it highly efficient for real-time applications.
- We achieve state-of-the-art results on benchmark datasets, demonstrating the effectiveness of our approach compared to existing methods, including MATE.

5.3 Related Works

Test-Time Training. TTT enhances model adaptability by leveraging unseen target data during inference, unlike domain generalization or adaptation methods, which operate solely during training. TTT methods can be broadly categorized into regularization-based and self-supervised approaches. Regularization-based methods include TENT (Wang *et al.*, 2021a), which minimizes prediction entropy by adapting BatchNorm parameters, and SHOT (Liang *et al.*, 2020), which combines entropy minimization with diversity regularization to develop robust feature extraction. MEMO (Zhang *et al.*, 2022a) applies data augmentations to test inputs and minimizes the entropy of averaged outputs, improving robustness. DUA (Mirza *et al.*, 2022) updates BatchNorm statistics to address distribution shifts with minimal overhead, while T3A (Iwasawa & Matsuo, 2021) refines the linear classifier using pseudo-prototypes, achieving backpropagation-free adaptation. These methods highlight the diversity of TTT strategies for addressing distribution shifts.

Among self-supervised approaches, TTT++ (Liu *et al.*, 2021b) introduces an additional self-supervised branch that leverages contrastive learning within the source model to facilitate adaptation to the target domain. Another method, TTT-MAE (Gandelsman *et al.*, 2022),

employs a MAE to address the one-sample learning problem effectively, demonstrating improved performance across various visual benchmarks. A recent method, MATE (Mirza *et al.*, 2023), is the first TTT framework specifically tailored for 3D point cloud data, enhancing the resilience of deep networks to distribution shifts during testing. It leverages a MAE objective, masking a significant portion of each target point cloud and tasking the network with reconstructing the complete structure before classification. In addition to these approaches, several works on test-time adaptation have explored updating model parameters during inference to handle distribution shifts effectively (Wang *et al.*, 2024b; Bahri *et al.*, 2024a).

Skeletal Representation. Skeletal representations are a compact and structural abstraction of 3D point clouds, capturing the underlying geometric and topological properties of objects. These representations simplify complex point cloud data by reducing it to a set of key skeletal points that capture the core of the geometry (fig. 5.1), which are particularly useful for tasks such as shape analysis, object recognition, and reconstruction. Several methods have been proposed to generate skeletal representations from 3D data. Early approaches often relied on medial axis transformations (MAT) (Choi, Choi & Moon, 1997) or Voronoi-based (Ogniewicz & Ilg, 1992) methods to extract the central skeleton of an object. While effective in capturing global geometry, these methods are typically computationally intensive and sensitive to noise (Li *et al.*, 2015; Sun, Choi, Yu & Wang, 2015; Yan, Letscher & Ju, 2018). More recent learning-based approaches, such as Point2Skeleton (Lin *et al.*, 2021) and Learnable Skeleton-Aware 3D Point Cloud Sampling (Wen, Yu & Tao, 2023), have shifted towards using neural networks to predict skeletal points directly from input point clouds. These methods leverage local geometric features and end-to-end optimization to produce accurate and robust skeletal representations. By capturing the core geometric structure of 3D point clouds, skeletal representations provide a compact and meaningful abstraction that is less sensitive to noise and distortions compared to raw point cloud data. This robustness makes them particularly suitable for dynamic adaptation scenarios, where models need to generalize effectively to unseen conditions. In this paper, we leverage skeletal representations with technical enhancements that enable the network to learn robust geometric features, thereby eliminating the need for extensive parameter updates during

TTA. Instead, our method relies solely on lightweight adjustments, such as updating BatchNorm statistics. This approach ensures both efficient and effective TTT while fully utilizing the inherent strengths of skeletal representations.

5.4 Method

5.4.1 Preliminaries

A point cloud is represented as $P \in \mathbb{R}^{N\times 3}$, where N is the total number of points, and each point is defined by its 3D spatial coordinates (x, y, z). This representation captures the geometric structure of 3D objects, but due to the irregular and unordered nature of point clouds, processing all N points can be computationally expensive and redundant. To reduce redundancy and focus on the most representative points, we apply FPS. This technique selects a subset of M points from the original point cloud as *centers*, denoted as:

$$C = FPS(P) \in \mathbb{R}^{M\times 3}, \tag{5.1}$$

where $M \ll N$. The FPS algorithm ensures that the selected centers are evenly distributed across the point cloud, preserving the overall geometric structure. For each center point $c_i \in C$, we construct its local neighborhood by selecting its K nearest neighbors from the original point cloud P using the KNN algorithm. This results in a neighborhood tensor:

$$P_{local} = kNN(C, P) \in \mathbb{R}^{M\times K\times 3}, \tag{5.2}$$

where K is the number of neighbors for each center point. Each neighborhood $P_{local}[i] \in \mathbb{R}^{K\times 3}$, corresponding to a center c_i, captures the local geometric features around c_i. These neighborhoods provide localized context, allowing the network to efficiently process the point cloud by focusing on both global structure (via the centers) and local geometric details (via the neighbors). This hierarchical tokenization forms the basis for subsequent feature extraction and skeletal prediction.

With the point cloud tokenized into centers and their local neighborhoods, the next step is to define the skeletal representation. The skeleton of a 3D shape provides a compact representation of its intrinsic geometric structure. It abstracts the shape by representing it as a collection of skeletal points and their corresponding radii, which together form a *skeletal mesh*. This concept was introduced by (Lin *et al.*, 2021) for learning-based approaches. A skeletal mesh is defined as a discrete set of skeletal spheres. Each sphere is represented as:

$$s = \big(c_s, r(c_s)\big) \in \mathbb{R}^4, \tag{5.3}$$

where $c_s \in \mathbb{R}^3$ is the center of the skeletal sphere, referred to as the *skeletal point*, and $r(c_s) \in \mathbb{R}$ is its associated radius, representing the distance from the center c_s to the surface of the original 3D shape. The skeletal mesh is constructed by connecting the skeletal spheres using two main elements. First, *edges*, denoted as $e_{ij} = (c_i, c_j)$, connect two skeletal points c_i and c_j, forming the basic linear structure of the skeleton. Second, *faces*, denoted as $f_{ijk} = (c_{s_i}, c_{s_j}, c_{s_k})$, are triangular connections between three skeletal points that capture 2D non-manifold surfaces within the mesh.

The skeletal representation possesses two key properties. The first is **recoverability**, where the skeletal mesh acts as a complete descriptor of the shape. This means the original 3D shape can be reconstructed by interpolating the skeletal spheres. For a skeletal sphere $s = (c_s, r(c_s))$, any point p on its surface can be expressed as:

$$p = c_s + r(c_s) \cdot v, \tag{5.4}$$

where v is a unit vector specifying the direction. The second property is **abstraction**, by which the skeleton simplifies the representation of the shape by capturing its fundamental structure, significantly reducing the complexity of the original point cloud while retaining its essential geometric information. This discrete skeletal representation approximates the Medial Axis Transform (MAT), which is a continuous formulation of the skeleton. Unlike the MAT, the skeletal mesh is more robust to boundary noise and perturbations, making it

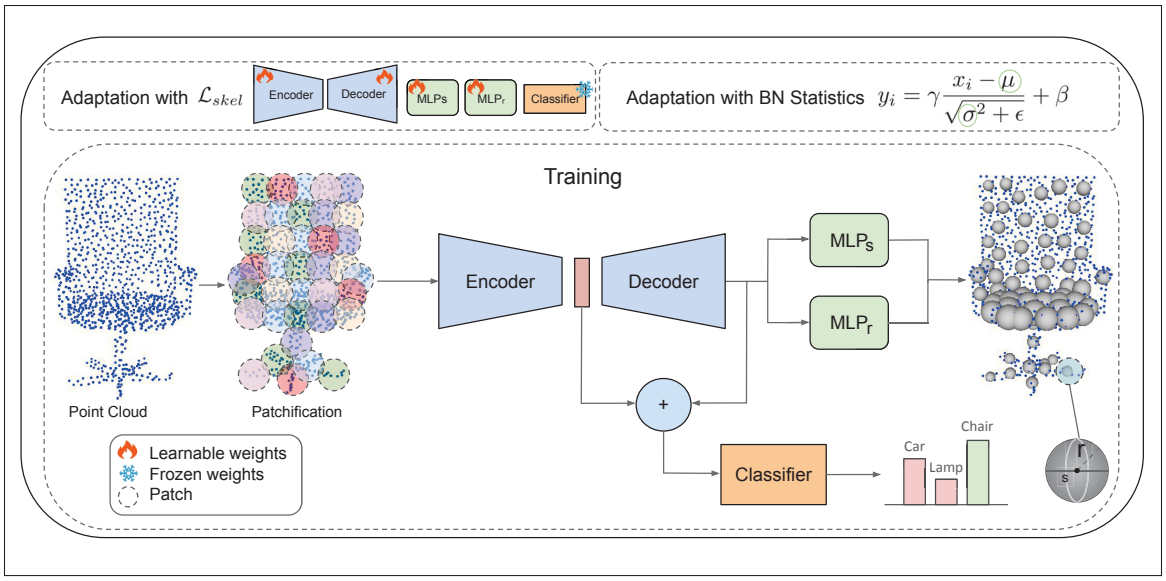

Figure 5.2 An overview of the SMART-PC framework. The framework integrates skeletal prediction and classification tasks, leveraging skeletal representations to extract robust geometric features. During online adaptation, two strategies are employed: adaptation with the skeletal loss $\mathcal{L}_{skel}$ and backpropagation, and a lightweight, backpropagation-free approach that updates only BatchNorm statistics

suitable for learning-based approaches. By predicting skeletal points and their radii, the skeletal representation provides a robust and efficient way to analyze and process 3D shapes, even under noisy or corrupted conditions.

5.4.2 Overview

In this paper, we propose a novel skeleton-based framework for Test-Time Training (TTT) designed to improve the robustness and adaptability of 3D point cloud classification under distribution shifts. Our method consists of three components: *Framework*, *Training*, and *Test-Time Adaptation* (TTA).

In the *Framework* component, we define the structure of the model, which includes components for both classification and skeletal point prediction. This involves investigating the architectural details and ensuring the network is capable of handling both tasks effectively. During the *Training* phase, the model is trained on source data to perform two tasks: skeletal prediction and

classification. The skeletal-based pre-training enables the model to learn robust and meaningful geometric features that enhance resistance to corruptions and generalize effectively across varying distributions. In the TTA phase, our method eliminates the need for backpropagation when the adaptation strategy is online, meaning the model is not reset after each epoch. Instead, it relies solely on lightweight updates to the BatchNorm statistics, enabling the model to adapt dynamically to target data while achieving a higher frame rate for real-time applications. The overall network architecture is illustrated in Figure 5.2.

5.4.3 Framework

Our framework is designed to perform two primary tasks: skeletal point and radius prediction, and point cloud classification. These tasks share a unified encoder while leveraging separate network components for their specific objectives. The framework is divided into two key branches: the *skeletal branch* for predicting skeletal representations and the *classification branch* for predicting class labels.

Skeletal Branch. The skeletal branch predicts skeletal points $S \in \mathbb{R}^{M\times 3}$ and their corresponding radii $R \in \mathbb{R}^{M\times 1}$ from the input point cloud. The encoder, denoted as E, processes the tokenized point cloud $P_{local} \in \mathbb{R}^{M\times K\times 3}$ to extract global and local geometric features:

$$F_{enc} = E(P_{local}) \in \mathbb{R}^{M\times d}, \tag{5.5}$$

where d is the dimensionality of the feature space. The encoder captures both local geometric details and global structural information.

The decoder, denoted as D, refines the encoded features to produce a contextually enriched representation:

$$F_{dec} = D(F_{enc}) \in \mathbb{R}^{M\times d}. \tag{5.6}$$

To predict skeletal points and radii, two separate Multi-Layer Perceptrons (MLP_s and MLP_r) are applied to the refined features F_{dec}. The skeletal points are predicted as:

$$c_s = MLP_s(F_{dec}) \in \mathbb{R}^{M\times 3}, \tag{5.7}$$

and the radii are predicted as:

$$r = MLP_r(F_{dec}) \in \mathbb{R}^{M\times 1}. \tag{5.8}$$

The skeletal branch leverages these predictions to model the underlying geometric structure of the input point cloud. Unlike existing skeletal prediction methods (Lin *et al.*, 2021; Wen *et al.*, 2023), which rely on convex combinations of input points to generate skeletal points, our approach directly predicts the skeletal points and radii from the refined features F_{dec}. Convex combination methods inherently inject raw point cloud data into the prediction process, making the task easier for the network. However, this dependency on the raw point cloud can hinder the encoder's ability to learn robust and meaningful geometric features, especially in the presence of noise or distribution shifts.

Classification Branch. The classification branch uses the shared encoder to extract features for class prediction. To enhance the classification performance, the features from the encoder F_{enc} and decoder F_{dec} are combined by a sum:

$$F_{combined} = F_{enc} + F_{dec}. \tag{5.9}$$

This combination allows the classification head to leverage high-level features from the decoder that are closely related to the skeletal points, alongside the global context from the encoder. By incorporating these skeletal-related features, the model enhances its ability to classify point clouds, particularly by utilizing structural information that aids in distinguishing complex or similar classes. The combined features are passed through the classification head, which consists

of multiple MLP layers, normalization, and dropout layers, to predict class probabilities:

$$p = Softmax(MLP_{cls}(F_{combined})) \in \mathbb{R}^K, \tag{5.10}$$

where K is the number of classes.

5.4.4 Training

In our framework, the skeletal branch is trained in a self-supervised manner since no labels are available for skeletal points, while the classification branch is trained in a supervised manner using labeled source data. To optimize these branches, we adopt loss functions inspired by prior works in skeletal representation (Lin *et al.*, 2021; Wen *et al.*, 2023). Below, we detail the loss functions used for training the skeletal and classification branches.

Skeletal Losses. The skeletal branch optimizes three complementary loss functions to ensure accurate prediction of skeletal points and their corresponding radii. Unlike existing skeletal methods that operate at the point level, our approach predicts skeletal points and radii at the patch level. For each patch, the model predicts the center of a skeletal sphere and its radius, which together abstract the local structure of the point cloud.

1) Point-to-Sphere Loss. This loss ensures that input points lie on the surface of their corresponding skeletal spheres and that skeletal spheres align closely with their associated input points P:

$$\begin{aligned}\mathcal{L}_{p2s} = &\sum_{p \in P} \left(\min_{s \in S} \|p - c_s\|_2 - r(c_s) \right) \\ &+ \sum_{s \in S} \left(\min_{p \in P} \|c_s - p\|_2 - r(c_s) \right),\end{aligned} \tag{5.11}$$

where c_s and $r(c_s)$ again denote the center and radius of skeletal sphere s.

2) Sampling Loss. To further ensure alignment between the skeletal spheres and the input point cloud, the Chamfer Distance between sampled points on the surface of skeletal spheres and the input points is calculated:

$$\mathcal{L}_{sampling} = \sum_{p \in P} \min_{t \in T} \|p - t\|_2 + \sum_{t \in T} \min_{p \in P} \|t - p\|_2, \tag{5.12}$$

where T represents points sampled uniformly on the surfaces of the skeletal spheres. This loss aggregates geometric information from the neighborhood of input points, effectively reducing high-frequency noise. As each skeletal point represents the local center of a region in P, the influence of individual noisy points is averaged, resulting in s being less sensitive to corruption.

3) Radius Regularization Loss. To avoid the instability caused by overly small radii due to noise, a regularization term that encourages larger radii is also used:

$$\mathcal{L}_{radius} = -\sum_{s \in S} r(c_s). \tag{5.13}$$

Furthermore, the radii r, predicted alongside c_s, capture the extent of each skeletal point's influence. This loss encourages larger radii, reducing sensitivity to localized noise by ensuring that each skeletal sphere represents a broader region of the point cloud. This abstraction minimizes the effect of outliers, further enhancing robustness.

The total skeletal loss is a weighted combination of these three terms:

$$\mathcal{L}_{skel} = \mathcal{L}_{p2s} + \lambda_1 \mathcal{L}_{sampling} + \lambda_2 \mathcal{L}_{radius}, \tag{5.14}$$

where $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are hyperparameters balancing the contributions of each loss.

Classification Loss. For the classification branch, we train the network using labeled source data. The features from the encoder and decoder are combined as described earlier and passed to the classification head. The classification loss is defined as the cross-entropy between the

predicted probabilities and the ground truth labels:

$$\mathcal{L}_{cls} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{k=1}^{K} y_{ik}\log(\hat{y}_{ik}), \tag{5.15}$$

where B is the batch size, K is the number of classes, y_{ik} is the ground truth label for the i-th sample in class k, and $\hat{y}_{ik}$ is the predicted probability for the same class.

The overall loss for training the network is a combination of the skeletal and classification losses:

$$\mathcal{L}_{total} = \mathcal{L}_{skel} + \mathcal{L}_{cls}. \tag{5.16}$$

This joint optimization allows the model to learn robust skeletal representations in a self-supervised manner while simultaneously leveraging labeled source data for classification.

5.4.5 Test-Time Adaptation (TTA)

In this work, we explore two modes of test-time adaptation, inspired by prior works such as MATE (Mirza *et al.*, 2023): *online adaptation* and *standard adaptation*. Both modes aim to adapt the model to corrupted target data during test time but differ in how the model state is managed across epochs and corruptions.

Online Adaptation. In the online mode, the model is not reset after each epoch, allowing the accumulated information from previous batches to influence the adaptation process. However, the model is reset at the beginning of adaptation for each new corruption type. Initially, we perform test-time adaptation in a backpropagation-free manner, updating only the statistical parameters of the Batch Normalization (BatchNorm) layers (i.e., the running mean and running variance). This approach leverages the observation that the features extracted by the model during pre-training on source data are highly robust and less sensitive to corruption. By updating only the BatchNorm statistics, the model dynamically adjusts to the target data, achieving higher frame rates for real-time applications while maintaining high performance.

To further analyze the effectiveness of the extracted features, we conducted an additional experiment where all model parameters, along with the BatchNorm statistical parameters, were updated during test-time adaptation. Specifically, we optimized the skeletal loss functions $\mathcal{L}_{p2s}$, $\mathcal{L}_{sampling}$, and $\mathcal{L}_{radius}$ (as described in section 5.4.4) to adapt the skeletal representation to the target distribution.

Standard Adaptation. In the standard mode, the model is reset after every batch, meaning each test batch adapts independently without accumulating information from previous batches. In this case, updating only the BatchNorm statistics has limited effectiveness since the adaptation does not retain context across batches. Consequently, for standard adaptation, we update all the parameters of the model using the skeletal loss functions $\mathcal{L}_{p2s}$, $\mathcal{L}_{sampling}$, and $\mathcal{L}_{radius}$. This allows the model to adjust more comprehensively to each batch of corrupted data.

5.5 Experiments

In this section, we present a comprehensive evaluation of our proposed method using multiple 3D point cloud datasets. To thoroughly examine its robustness and generalization capabilities, we conduct experiments on three benchmark datasets: ModelNet40-C (Sun *et al.*, 2022), ShapeNet-C (Chang *et al.*, 2015), and ScanObjectNN-C (Uy *et al.*, 2019b). These datasets encompass a variety of real-world challenges, including different levels of corruption and noise, enabling us to showcase the effectiveness of our approach in diverse and complex scenarios. Furthermore, we evaluate the performance of our method in both online and standard modes. For the online mode, we compare the backpropagation-free strategy with backpropagation-based adaptation to highlight the efficiency and robustness of our approach. A detailed description of the dataset, additional experiments, expanded visualization results related to skeletons, and a comprehensive evaluation of our method across all datasets, including accuracy metrics for each corruption type, are provided in the Supplementary Materials.

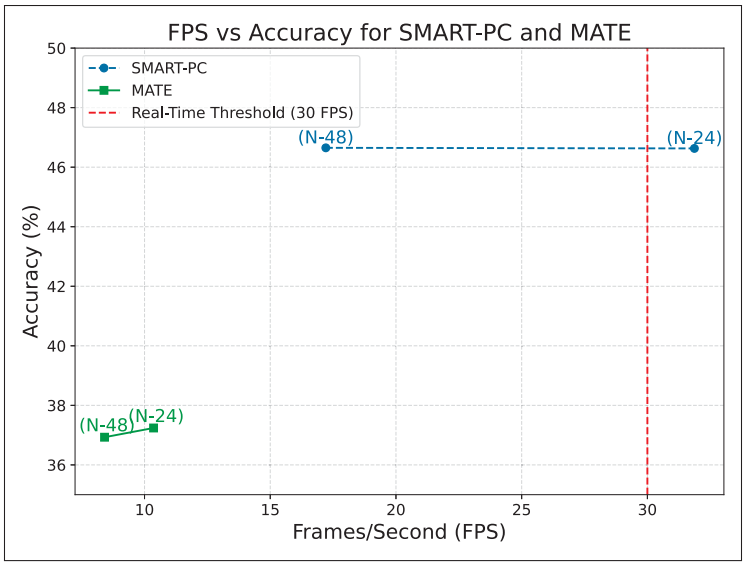

Figure 5.3 Frames/Second vs. Accuracy for SMART-PC and MATE during online adaptation on the ScanObjectNN dataset. SMART-PC achieves higher frame rates and accuracy, surpassing the real-time threshold (30 Frames/Second) in the (N-24) setting

5.5.1 Implementation Details

For both training and adaptation, we employed the Point-MAE backbone, following the settings outlined in the MATE paper (Mirza *et al.*, 2023) to ensure a fair and consistent comparison. The batch size is set to 1 for both adaptation modes, with 1 iteration for the online mode and 20 iterations for the standard mode, identical to the MATE paper for fair comparison once again. The optimizer and learning rate are identical to those used in the MATE paper. For augmentation, *scale-transfer* is used during pre-training, similar to MATE. During adaptation, however, MATE employs random masking, whereas we use random rotation in all experiments. All experiments were conducted using a single NVIDIA A6000 GPU, ensuring consistency across all tested configurations. Additionally, we evaluate the frames per second rate of our method and compare it with MATE.

Real-Time Adaptation. Figure 5.3 illustrates the adaptation performance of SMART-PC and MATE in terms of Frames/Second on the ScanObjectNN-C dataset. The adaptation strategy employed is online, with a batch size of 1 and a single iteration per adaptation step. The terms

"N-48" and "N-24" represent the number of created batches, which are generated differently for the two methods: MATE uses random masking, while SMART-PC employs random rotations.

A significant advantage of SMART-PC is its ability to perform adaptation in online mode without requiring backpropagation. By updating only the statistics of BatchNorm layers, SMART-PC achieves significantly higher Frames/Second compared to MATE, which relies on computationally expensive backpropagation during adaptation. As shown in fig. 5.3, reducing the number of batches from N-48 to N-24 does not affect the accuracy of either method. However, MATE cannot increase its Frames/Second rate due to the inherent limitations of backpropagation. In contrast, SMART-PC demonstrates a substantial increase in Frames/Second, surpassing the real-time threshold of 30 Frames/Second when fewer batches are used. This highlights the efficiency and scalability of SMART-PC, making it a practical solution for real-world applications requiring real-time adaptation. For additional comparisons with other methods, including diffusion-based approaches, please refer to the Supplementary Materials.

5.5.2 Main Results

In all results tables, "Org-SO" refers to the evaluation of a naive pretrained model, without any additional pre-training branch, on the corrupted dataset without adaptation. "MATE-SO" represents a model pretrained with a reconstruction branch, evaluated on the corrupted dataset without adaptation. "SMART-PC-SO" denotes our model pretrained with a skeleton branch on clean data, evaluated on the corrupted dataset without adaptation. Results marked with * indicate reproduced outcomes, and "†" refers to adaptation using BatchNorm statistical parameters without backpropagation.

ModelNet-40C. Table 5.1 presents the top-1 classification accuracy on the ModelNet40-C dataset under various corruption types. In the source-only setting, SMART-PC-SO achieves an average accuracy of 61.7%, significantly outperforming both MATE-SO (53.7%) and Org-SO (54.0%). This demonstrates the robustness of our skeleton-based pre-training in handling distribution shifts without adaptation.

Table 5.1 Top-1 Classification Accuracy (%) for the ModelNet-40C, ScanObjectNN-C and ShapeNet-C datasets. Differences between SMART-PC variants and MATE counterparts are indicated as (±X%). * denotes reproduced results, † denotes adaptation without backpropagation, and ‡ denotes diffusion-based test-time adaptation methods

	ModelNet-40C	ScanObjectNN-C	ShapeNet-C
Org-SO*	54.0	37.0	61.3
MATE-SO*	54.4	34.5	56.5
SMART-PC-SO	**61.7** (+7.3)	**38.7** (+4.2)	**64.5** (+8.0)
DUA (Mirza *et al.*, 2022)	54.7	38.7	60.8
TTT-Rot (Sun *et al.*, 2020)	53.0	–	60.9
SHOT (Liang *et al.*, 2020)	26.6	29.6	36.2
T3A (Iwasawa & Matsuo, 2021)	55.7	–	54.4
TENT (Wang *et al.*, 2021a)	26.5	27.7	37.4
CloudFixer-Standard‡ (Shim, Kim & Yang, 2024)	68.0	-	-
CloudFixer-Online‡ (Shim *et al.*, 2024)	77.2	-	-
DDA-Standard‡ (Gao *et al.*, 2023)	68.1	-	-
SVWA-Standard* (Bahri *et al.*, 2024a)	57.1	37.4	50.5
MATE-Standard* (Mirza *et al.*, 2023)	63.0	36.9	63.1
SMART-PC-Standard	**63.1** (+0.1)	**39.6** (+2.7)	**64.4** (+1.3)
BFTT3D* (Wang *et al.*, 2024b)	57.2	33.0	60.7
MATE-Online* (Mirza *et al.*, 2023)	70.6	36.9	**69.1**
SMART-PC-Online†	70.8	46.7	65.9
SMART-PC-Online	**72.9** (+2.3)	**47.4** (+10.5)	67.1 (-2.0%)

For adaptation strategies, in the standard mode, SMART-PC-Standard achieves an average accuracy of 63.1%, higher than MATE-Standard (63.0%). In the online mode, SMART-PC-Online attains the highest average accuracy of 72.9%, compared to 69.6% for MATE-Online. Notably, SMART-PC-Online† leverages a backpropagation-free strategy, which enables efficient adaptation while maintaining superior performance. These results highlight the effectiveness and scalability of SMART-PC in both source-only (without adaptation) and adaptive settings under challenging conditions.

ShapeNet-C. Table 5.1 summarizes the top-1 classification accuracy on the ShapeNet-C dataset under various corruption types. In the source-only setting, SMART-PC-SO achieves an average accuracy of 64.5%, outperforming both MATE-SO (56.5%) and Org-SO (61.3%). This result highlights the effectiveness of our skeleton-based pre-training in generalizing to unseen corruptions without adaptation.

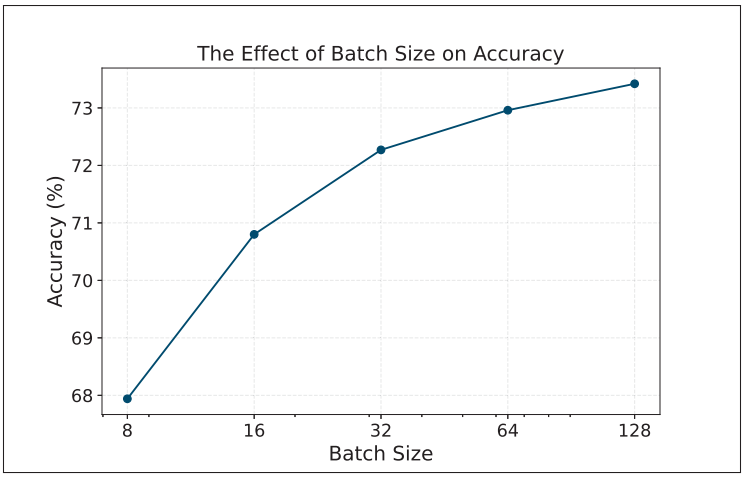

Figure 5.4 Impact of batch size on mean accuracy for the ModelNet-C dataset in standard mode

ScanObjectNN-C. Table 5.1 presents the top-1 classification accuracy across 15 corruption types on the ScanObjectNN-C dataset. In the source-only setting, SMART-PC-SO achieves an average accuracy of 38.7%, significantly outperforming both MATE-SO (34.5%) and Org-SO (37.0%).

In the standard adaptation mode, SMART-PC-Standard achieves an average accuracy of 39.6%, slightly higher than MATE-Standard (36.9%). This improvement highlights the effectiveness of leveraging skeletal representations for enhancing robustness to distribution shifts.

In the online mode, SMART-PC-Online achieves the highest average accuracy of 47.4%, significantly outperforming MATE-Online (36.9%). Furthermore, SMART-PC†, which utilizes a backpropagation-free adaptation strategy by updating only the BatchNorm statistics, achieves a competitive accuracy of 46.7%. The efficiency of the backpropagation-free strategy highlights its practicality for real-world applications requiring high Frames per Second.

Overall, these results demonstrate that SMART-PC consistently achieves state-of-the-art performance across all adaptation modes, with notable gains under challenging real-world corruptions, highlighting its robustness and generalization capability.

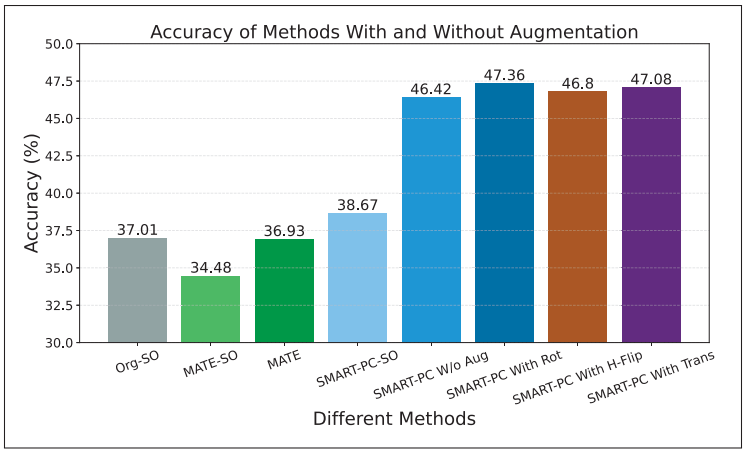

Figure 5.5 Effect of augmentation during adaptation in our method, and comparison with other methods on the ScanObjectNN dataset in online mode

5.5.3 Ablation Study

Batch Size. We evaluate the effect of batch size on our method's performance using the ModelNet40-C dataset in standard adaptation mode, where all parameters are updated, with 20 iterations as in the MATE paper. As shown in Figure 5.4, increasing the batch size improves accuracy, from 67.94% at batch size 8 to 73.42% at batch size 128. This improvement showcases the effectiveness of larger batch sizes for achieving higher accuracy. To ensure fairness in comparison with MATE and other methods and to avoid increasing computational costs, we used a batch size of 1 for all main results in both standard and online modes.

Augmentation. Figure 5.5 illustrates the impact of different augmentations and the absence of augmentation on our method during adaptation. This experiment was conducted on the ScanObjectNN dataset in online mode with batch size 1 and iteration 1. Consistent with the MATE paper, we created 48 batches using random rotations to introduce diversity among the data. Even without augmentation (*SMART-PC W/O Aug*), our method achieves a 46.42% accuracy, outperforming *MATE* (36.93%) and demonstrating the robustness of our skeleton-based framework. Applying random rotations (*SMART-PC With Rot*) increases accuracy to 47.36%. Other augmentations, including horizontal flipping (*SMART-PC With H-Flip*) and translations

(*SMART-PC With Trans*), achieve comparable improvements, with accuracies of 46.8% and 47.08%, respectively.

Table 5.2 Ablation study of skeleton loss coefficients on ModelNet40 and ModelNet40-C (online adaptation). The best configuration corresponds to the original coefficients from the Point2Skeleton paper

Pt2Sphere	Sampling	RadiusReg	Source Acc(%)	Corrupted Acc(%)
1.0	1.0	0.0	91.3	67.82
0.0	1.0	1.0	91.6	67.79
1.0	0.0	1.0	91.6	67.80
1.0	1.0	1.0	91.2	72.84
0.3	**1.0**	**0.4**	**91.3**	**72.95**

Feature Summation. We conduct an ablation study on the impact of summing the features from the shared encoder, used for both skeletal and classification tasks, with the decoder features. This summation improves SMART-PC's accuracy from 62.4% to 63.1% on the ModelNet-C dataset in standard mode, compared to using only the encoder features for classification.

Skeleton Loss Coefficients. To further evaluate the contribution of each loss component in our skeleton-based pretraining, we conducted an ablation study using different coefficients for the Skeletal loss terms: point-to-sphere loss, sampling loss, and radius regularization. The experiments were performed on the ModelNet40 and ModelNet40-C datasets under the online adaptation setting.

As shown in Table 5.2, the best performance is obtained with the coefficient set $(0.3, 1.0, 0.4)$, which corresponds to the original settings in the Point2Skeleton paper (Lin *et al.*, 2021). This configuration achieves the highest accuracy of 72.95% on corrupted data (Corrupted Acc), confirming that each skeletal loss term contributes meaningfully to the learning of robust features under corruption. Additionally, as described in the main paper, the Radius Regularization Loss (Equation 6) is designed to avoid instability caused by overly small radii, especially under noisy conditions. This loss encourages the model to learn larger and more stable radii, which improves the robustness of the skeletal abstraction. Although we do not observe excessively large radii, the Point-to-Sphere and Sampling losses (Equations 11 and 12) implicitly constrain radius size

by preserving geometric consistency. As shown in Table 5.2, removing the regularization term leads to a drop in performance, confirming its importance.

Table 5.3 Mean accuracy (%) of BFTT3D using different pretrained models under the backpropagation-free setting. SMART-PC-SO achieves the best results across all datasets

Dataset	Org-SO	MATE-SO	SMART-PC-SO
ScanObjectNN-C	33.00	33.22	**35.90**
ModelNet40-C	57.16	54.71	**65.25**
ShapeNet-C	60.73	53.07	**62.24**

Evaluating Pretraining Strategies in Backpropagation-Free Adaptation. Our method supports two adaptation modes: one with backpropagation and one that is backpropagation-free. The goal of the backpropagation-free mode is to show that pretraining with a skeleton-based decoder encourages the model to learn robust and meaningful geometric features. These features are resilient enough that, during test-time, simply updating the BatchNorm statistical parameters (i.e., running mean and variance) is sufficient to improve performance—without performing gradient-based updates.

To validate this effect, we conducted additional experiments using the BFTT3D (Wang *et al.*, 2024b) adaptation method across three different pretraining strategies (Org-SO, MATE-SO, and SMART-PC-SO). Each pretrained model was evaluated using the same BFTT3D adaptation strategy under the backpropagation-free setting. As shown in Table 5.3, our SMART-PC-SO model consistently outperforms both Org-SO and MATE-SO across all three datasets. This provides strong evidence that the skeletal decoder encourages the model to extract more structure-aware and corruption-resilient features, which support effective test-time adaptation without updating model weights.

5.6 Conclusion

In this paper, we proposed SMART-PC, a skeleton-based framework for robust and efficient test-time training of 3D point cloud models under distribution shifts. By leveraging skeletal representations, SMART-PC enhances robustness to corruptions while enabling high Frames

per Second during online adaptation through a backpropagation-free strategy that updates only BatchNorm statistics. Experiments on benchmark datasets, including ModelNet40-C, ShapeNet-C, and ScanObjectNN-C, demonstrate state-of-the-art performance in both source-only and adaptation settings. Overall, SMART-PC provides a scalable and practical solution for real-world applications requiring robust 3D point cloud classification under challenging conditions. Future work may explore extending this framework to other 3D tasks, such as semantic segmentation, instance segmentation, and object detection.

CONCLUSION AND RECOMMENDATIONS

This thesis investigated how to build 3D deep learning systems that remain reliable across the full lifecycle of perception: from label-efficient pretraining, to efficient geometric modeling, to robust deployment under real-world distribution shift. Despite recent advances in supervised 3D vision, existing models continue to face two fundamental limitations: their dependence on resource-intensive annotation and computation, and their brittleness when exposed to unseen sensor conditions or environmental noise. The work developed in this thesis addresses these challenges by proposing new methods in both self-supervised learning and test-time learning, unified by the goal of extracting stable geometric structure from raw point clouds.

Chapter 2 focused on learning meaningful and transferable geometric representations without relying on manual annotations. **GeoMask3D** introduced a principled geometry-aware masking strategy that replaces purely random patch selection with a structured, complexity-driven approach grounded in intrinsic point-cloud geometry. Instead of treating all patches as equally informative, GeoMask3D computes local geometric complexity to selectively mask or preserve regions according to their structural importance. This forces the masked autoencoder to reconstruct geometrically challenging areas rather than overfitting to flat, redundant surface patches, ultimately yielding richer and more discriminative latent representations. To ensure consistency and robustness, the method also incorporates a feature-level distillation mechanism that stabilizes complexity estimation across epochs and prevents drift in geometric predictions during pretraining. Together, these components encourage the encoder to learn geometry-aware features that generalize well beyond reconstruction. Extensive evaluations demonstrate that GeoMask3D consistently improves downstream performance on classification, segmentation, and reconstruction benchmarks, highlighting the value of integrating explicit geometric priors into masked point-cloud pretraining.

Chapter 3 presented **Spectral-Informed Mamba**, which addressed a fundamental architectural limitation in adapting state-space models to 3D point clouds: the absence of a stable and geometry-aware token ordering. Since Mamba requires a sequential input, applying it directly to unordered point sets leads to ordering ambiguity, sensitivity to rotations, and loss of geometric coherence. Spectral-Informed Mamba resolves this by leveraging the Laplacian spectrum of the point-cloud graph to construct an *isometry-invariant* traversal that respects surface smoothness and structural continuity. Unlike heuristic or grid-based orderings, the proposed spectral traversal emerges directly from intrinsic geometry, ensuring consistent ordering. Building on this ordering, Spectral-Informed Mamba introduces hierarchical local grouping to preserve fine-scale spatial relations and a token realignment mechanism that aligns Mamba dynamics with the irregular geometry of 3D neighborhoods. This combination enables state-space models to operate with linear complexity while capturing long-range dependencies without sacrificing geometric fidelity. The resulting framework closes a long-standing gap between computational efficiency and geometric robustness, demonstrating that Mamba architectures—when equipped with principled geometric traversal—can outperform transformer-based masked pretraining methods on large-scale 3D representation learning benchmarks.

Chapter 4 addressed robustness under distribution shift, a critical requirement for real-world 3D deployment. **Sampling-Variation Weight Averaging (SVWA)** introduced the first comprehensive TTA strategy tailored to point clouds. The method generates multiple geometric variations of each test sample—reflecting realistic changes in sampling, and local neighborhoods—and adapts the model independently to each variant. By harnessing this sampling diversity rather than treating it as noise, SVWA turns inherent randomness into a stabilizing signal. Averaging the adapted model weights nudges the solution toward a flatter optimum, leading to significantly improved generalization under corruption, sensor noise, and occlusion.

Finally, Chapter 5 introduced **SMART-PC**, a skeleton-based test-time training framework that leverages structural 3D priors to enable fast and stable adaptation. SMART-PC learns robust skeletal abstractions during pretraining—compact representations that capture object topology and remain consistent under viewpoint, density, and corruption variations. At test time, these skeletal features guide adaptation through a lightweight batch-statistics update rather than full gradient-based optimization, avoiding the latency and instability that characterize conventional TTT methods. This design allows SMART-PC to operate in real time, making it applicable to robotics, navigation, and embedded perception systems. By aligning skeletal geometry with model predictions during deployment, SMART-PC achieves reliable adaptation even under severe distribution shifts, demonstrating the value of high-level structural cues for resilient 3D test-time learning.

Taken together, these contributions form a unified pipeline for robust and generalizable 3D point-cloud perception. The thesis demonstrates that (i) geometric priors are essential for meaningful self-supervised pretraining, (ii) spectral traversal provides an effective mechanism for enabling efficient 3D state-space modeling, and (iii) sampling diversity and structured geometric abstractions can be leveraged to achieve effective test-time learning without labels. By integrating these insights, this work advances toward 3D perception systems that are efficient, interpretable, robust to corruption, and suitable for deployment in real-world environments.

While the proposed methods significantly advance the robustness and efficiency of 3D models, several open directions remain. Extending spectral traversal to large-scale point clouds, streaming sensors, or LiDAR sequences could broaden its applicability in robotics and mapping. Another open direction is incorporating skeletal abstractions directly into self-supervised pretraining objectives, enabling models to learn richer geometric features without relying on test-time training. Using the stability and topology encoded by skeletons during pretraining may yield representations that generalize more effectively across tasks and domains. For test-time

adaptation, exploring continual or lifelong TTA—where models continuously refine themselves over long deployment periods—remains a promising direction, especially when combined with memory-efficient mechanisms that avoid catastrophic forgetting.

In summary, this thesis contributes new geometric, spectral, and adaptive principles for 3D deep learning, laying the groundwork for perception systems that can learn efficiently, operate robustly, and generalize reliably under diverse real-world conditions.

APPENDIX I

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED GEOMASK3D: GEOMETRICALLY INFORMED MASK SELECTION FOR SELF-SUPERVISED POINT CLOUD LEARNING IN 3D

1. Implementation Pipeline

We used PyTorch to implement the core functionalities of our approach. The codebase is structured into two primary parts: *main-pretrain* and *main-finetune*.

Pretraining Phase (*main-pretrain*). In the *main-pretrain* section, we focus on the initial training phase of our models. In this phase, which is crucial to establish a robust foundation for our models, we first load the ShapeNet dataset and then apply our methods to train the models. algorithm 1 summarizes the training steps for GM3D and includes the pseudocode for the reconstruction of point clouds, generation of geometric complexity, and the distillation of knowledge within the feature space.

Finetuning Phase (*main-finetune*). Once the pretraining is complete, we proceed to the *main-finetune* part. In this stage, only the encoder of student $\mathcal{E}^s$ from the pretrained models is carried forward. The output of this phase is directly responsible for the experimental results presented in our paper.

To ensure complete transparency and reproducibility of our results, we have made all relevant materials publicly available. This includes:

- The full source code for both *main-pretrain* and *main-finetune* phases.
- All log files containing the detailed results of our experiments
- Pretrained models for both pretraining and finetuning stages.

All these resources can be accessed through our GitHub repository. This repository includes everything needed to understand our code, covering all aspects of the implementations and the reproduction of the results. Moreover, the specifics of our network's hyperparameters for

Algorithm 1 Pseudo-Code of GM3D in a PyTorch-like Style

```
# teacher inference
-, GC^t = GM3D^t(X)
# curriculum patch selection
M = Mask-Generation(GC^t, N^sel)
# student forward to compute objectives
X^rec, f^rec_s, GC^s = GM3D^s(X^v)
# knowledge teacher (frozen graph)
f^rec_f = E^f(X)
# compute losses
# feature-space
L^rec_f = MSE(f^rec_s[M], f^rec_f[M])
# point-space
L^rec_p = Chamfer(X^rec[M], X[M])
# both spaces
L^rec = L^rec_p + L^rec_f
# geometric complexity
L^GC = DRC(L^rec, GC^s, M)
# final loss
L = L^rec + L^GC
return L
```

the pretraining and fine-tuning phases are comprehensively detailed in Table I-1. Additionally, the pre-trained model for the Knowledge Teacher is selected based on the baseline methods, Point-MAE and Point-M2AE.

2. Additional Visualization

Geometric Complexity. Figure I-1 illustrates the GC analysis of randomly selected point clouds from the ShapeNet dataset. This illustration highlights the model's capability to assess GC at the patch level, where the red points denote areas of high GC, while the blue points indicate areas of low GC.

Reconstructed Points. To elucidate the capabilities of Masked Autoencoders (MAEs) in processing point cloud data, Figure I-2 provides a visual sequence involving the original input, the intermediate masking phase, and the reconstructed output. The first column, titled "Input

Table-A I-1 Hyperparameter configuration

Config	Value
Optimizer	AdamW
Base learning rate	1e-3
Weight decay	0.05
Momentum	$\beta_1, \beta_2 = 0.9, 0.95$
α	1.0
β (After epoch 15)	1000.0
γ (After epoch 15)	10.0
Batch size (Point-MAE+GM3D)	256
Batch size (Point-M2AE+GM3D)	128
Learning rate schedule	cosine decay
Pre-training epochs	400
Fine-tuning epochs	500
Augmentation (pretraining)	random scaling and translation
Augmentation (finetuning)	random rotation

Point Cloud", displays the entirety of the point cloud data, illustrating the initial condition before any processing. The subsequent column, "Masked Point Cloud", reveals only the points that remain visible after a portion of the data has been masked. The final column, "Reconstructed Point Cloud", demonstrates the model's ability to infer and restore the masked parts of the point cloud. The visual comparison in Figure I-2 distinctly highlights the high accuracy of

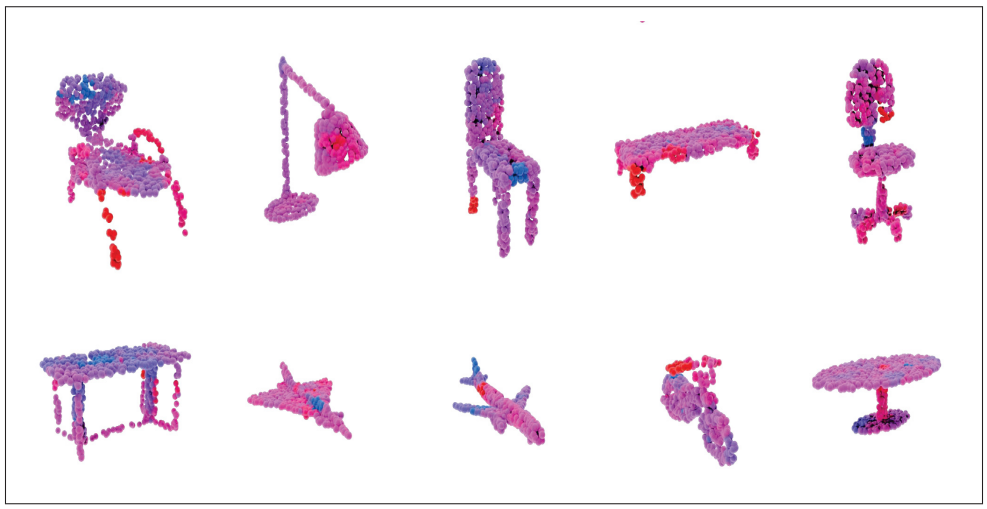

Figure-A I-1 Visualization of GC values on diverse point clouds from the ShapeNet dataset

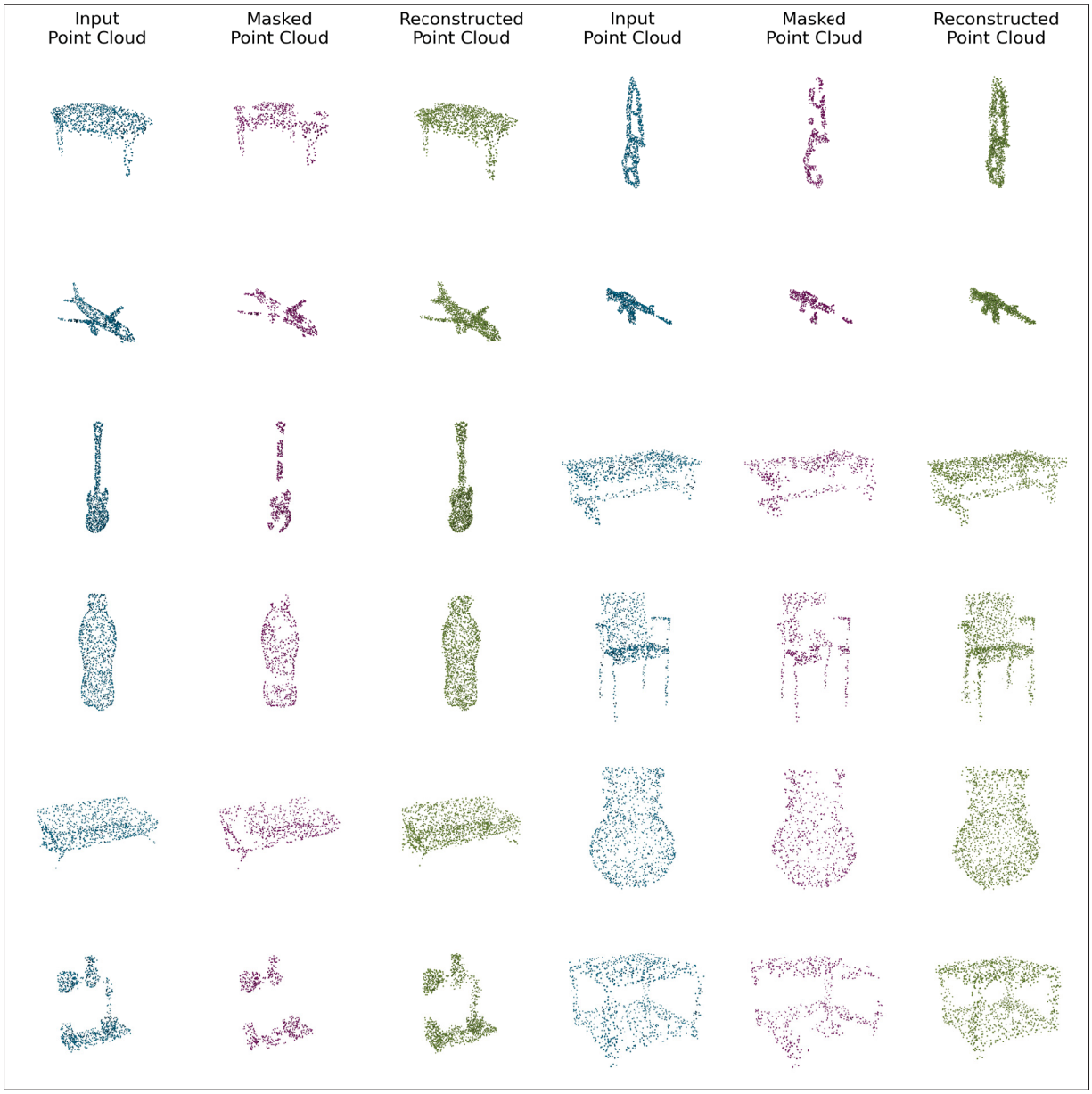

Figure-A I-2 Reconstruction results on the ShapeNet dataset

the reconstructed points, underscoring the efficacy of our proposed method. It is noteworthy that these visual results were obtained using Point-MAE+GM3D. For a fair and consistent comparison, the mask ratio used here is like Point-MAE (60%).

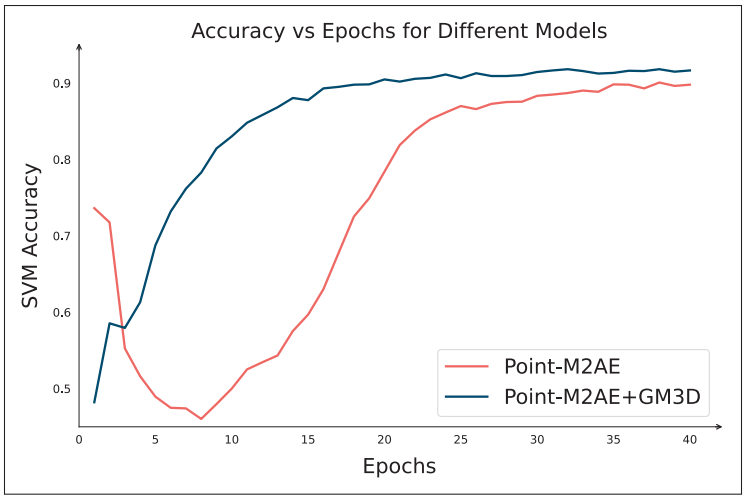

Figure-A I-3 Comparison of convergence speed during the training phase (Point-M2AE)

3. Additional Analyses

Pretraining Phase. In Figure I-3, we illustrate the progression of SVM Accuracy throughout the pretraining phase on the ModelNet40 dataset. The red curve represents the Point-M2AE model, while the blue curve denotes the Point-M2AE enhanced with our GM3D method. It is evident from the graph that the incorporation of GM3D leads to a substantial improvement in SVM accuracy, reflecting the model's enhanced classification performance. A key observation is the accelerated convergence rate of the GM3D-augmented model; it achieves a rapid increase in accuracy within the initial epochs, demonstrating not only the efficacy of GM3D in facilitating faster learning but also indicating an enhanced ability to generalize from the training data.

Fine-tuning Phase. Figure I-4 displays fine-tuning accuracy on the OBONLY dataset, revealing that the integration of our GM3D module with Point-M2AE leads to the highest accuracy. Compared to the baseline Point-M2AE and the model trained from scratch, Point-M2AE+GM3D demonstrates a more significant improvement and exhibits less variability.

Integration of GeoMask3D into Point-FEMAE. In this section, we present an experimental evaluation of integrating our GeoMask3D approach into the Point-FEMAE (Zha *et al.*, 2024) framework. Point-FEMAE employs two types of masking strategies: global masking and local

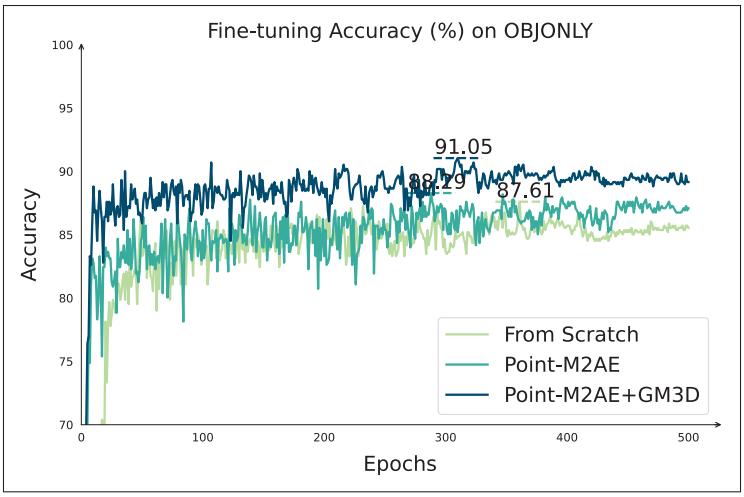

Figure-A I-4 Fine-tuning vs. Training from scratch on ScanObjectNN (Point-M2AE)

masking. To enhance its capability, we modified the network by incorporating the geometric complexity decoder $\mathcal{D}^{s}_{GC}$ and replacing the original random global masking strategy with our geometrically guided masking technique.

While Point-FEMAE also utilizes local masking, where Euclidean distances guide the selective masking of tokens related to meaningful parts of the object, this strategy already contributes to capturing geometric structures effectively. However, this local masking approach may overlap with the objectives of our GeoMask3D, potentially reducing its distinct impact. Despite this, integrating these two methods provides valuable insights into the effectiveness of geometrically guided masking.

To assess the effectiveness of our approach, we pre-trained Point-FEMAE+GM3D on the ShapeNet dataset and evaluated it on the OBJ-ONLY dataset. The results, presented in Table I-2, demonstrate that our modification improves performance. For comparison, we reproduced the baseline results of Point-FEMAE on this dataset using the original code and the pre-trained model available in the official repository.

Table-A I-2 Point-FEMAE with and without GeoMask3D on the OBJ-ONLY dataset

Method	OBJ-ONLY
Point-FEMAE	92.08
Point-FEMAE + GM3D	**92.77**

Impact of Geometric-Guided Masking. To further analyze the contribution of geometric-guided masking to the performance improvements observed in our method, we conducted an ablation study to isolate its effect from the GC prediction task.

To investigate the isolated impact of GC prediction, our method is pre-trained using the Point-MAE backbone with GC prediction but without geometric-guided masking. Instead of our masking strategy, we employed random masking. The objective of this experiment was to determine whether GC prediction alone contributes to the performance gains or if the synergy with geometric-guided masking is essential. We evaluated the pretrained model on the OBJ-ONLY dataset, as presented in Table I-3.

As shown in the results, the model trained with both GC prediction and geometric-guided masking outperforms the one using GC prediction alone. This highlights the necessity of integrating geometric-guided masking with GC prediction to achieve optimal performance.

4. Time Analysis

In this section, we provide a detailed analysis of the computational resources required by our method compared to the baseline Point-MAE backbone. All experiments were conducted under identical hardware settings using an NVIDIA A6000 GPU with a batch size of 128.

Table-A I-3 Ablation study on the impact of Geometric-Guided Masking

Method	OBJ-ONLY
Point-MAE + GC Prediction + Random Masking	89.67
Point-MAE + GC Prediction + Geometric-Guided Masking	**90.36**

4.1 Pre-Training

To evaluate the computational overhead introduced by our approach, we measured the pre-training time per epoch. The results indicate that:

- Point-MAE requires approximately 2.8 minutes per epoch.
- Our method requires 4.3 minutes per epoch.

The additional 1.5 minutes per epoch in our method is not solely due to the inclusion of the knowledge teacher model (which remains frozen during training), but also results from the computational procedures involved in predicting geometric complexity.

4.2 Downstream Tasks

As highlighted in the main paper, in downstream tasks, only the encoder of the student model is utilized. Since this encoder is identical to the one used in the baseline methods, our approach maintains computational efficiency during downstream inference. Thus, our method incurs no additional computational burden compared to baseline methods in downstream tasks.

APPENDIX II

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED SPECTRAL INFORMED MAMBA FOR ROBUST POINT CLOUD PROCESSING

1. Computational Efficiency, Runtime, and Memory Usage

In this section, we conducted a comprehensive analysis regarding the computational efficiency, runtime, and memory usage of our SAST approach. The focus of these experiments is to assess the overhead introduced by our SAST in comparison to the Point-Mamba backbone.

Our SAST strategy employs SciPy's sparse eigen-solver (function `eigs` implementing the implicitly restarted Arnoldi algorithm) to compute the first eigenvectors of the graph Laplacian. As highlighted in the main paper, this operation is not expensive since the size of this matrix depends on the number of patches which is much less than the original number of points (128 vs. 2048). Additionally, even when increasing the number of patches (tokens), the computational overhead of this step is limited due to three reasons: *i*) the Laplacian matrix is very sparse as we only consider the K nearest neighbors of each patch ($K \approx 20$), *ii*) we only compute the first k eigenvectors ($k \approx 5$ in our experiments), and *iii*) this computation is done **only once** for each point cloud in a **pre-processing step**.

Memory Usage: As shown in Figure II-1, the memory usage of our SAST strategy (black line) is significantly lower than the memory usage of the Point-Mamba backbone (red line). When increasing the number of patches along the x-axis, our strategy based on a sparse eigen-solver does not require substantially more memory compared to the backbone. The star in this figure shows the used number of tokens in downstream tasks.

Runtime: Our SAST strategy, which can be implemented in the data loader and ran in parallel on CPU, is also fast. As can be seen in Figure II-2, the runtime of SAST scales well when increasing the number of patches (tokens), and only a small amount of runtime is added in training or inference for the token length of 128 used in our main experiments (yellow star).

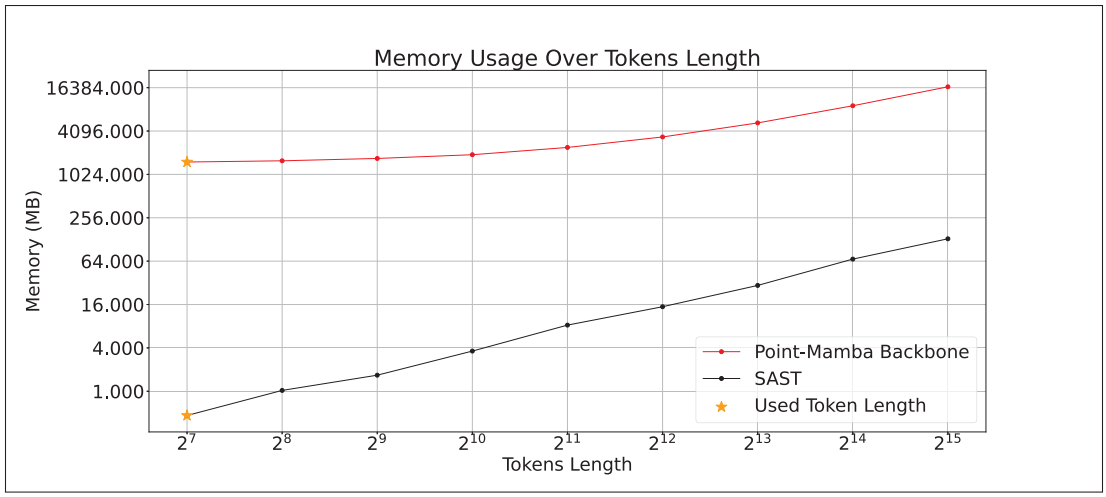

Figure-A II-1 Memory Usage Over Tokens Length. Both axes are scaled by $\log_2$ for better visualization

FLOPS: Figure II-3 presents the relationship between FLOPS and token length for both the Point-Mamba backbone and our SAST method. Compared to running the Point-Mamba back, our SAST demonstrates a more gradual increase in FLOPS due to the use of sparse computations and the low number of eigenvectors involved. Once again, this shows the limited overhead of incorporating SAST, even as token length increases.

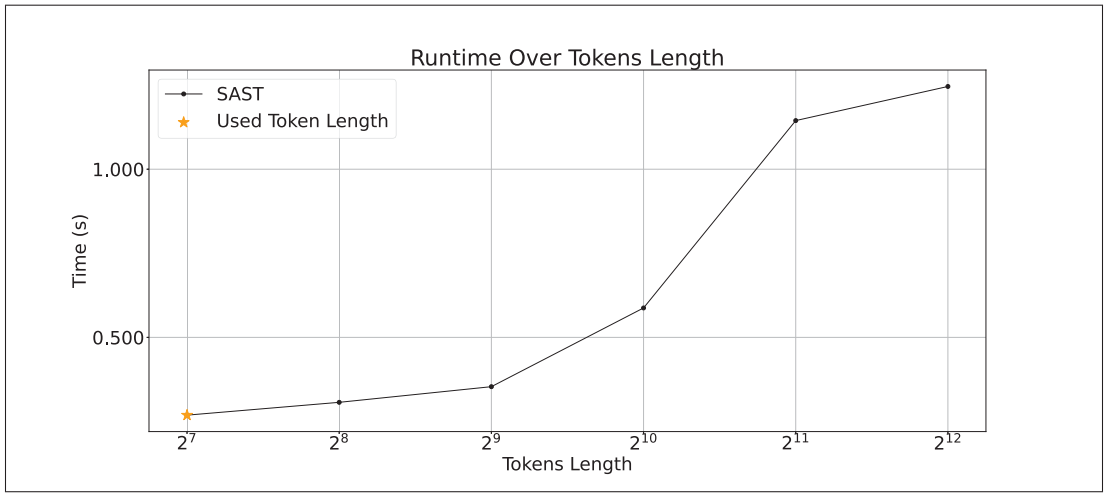

Figure-A II-2 Runtime Over Tokens Length. Both axes are scaled by $\log_2$ for better visualization

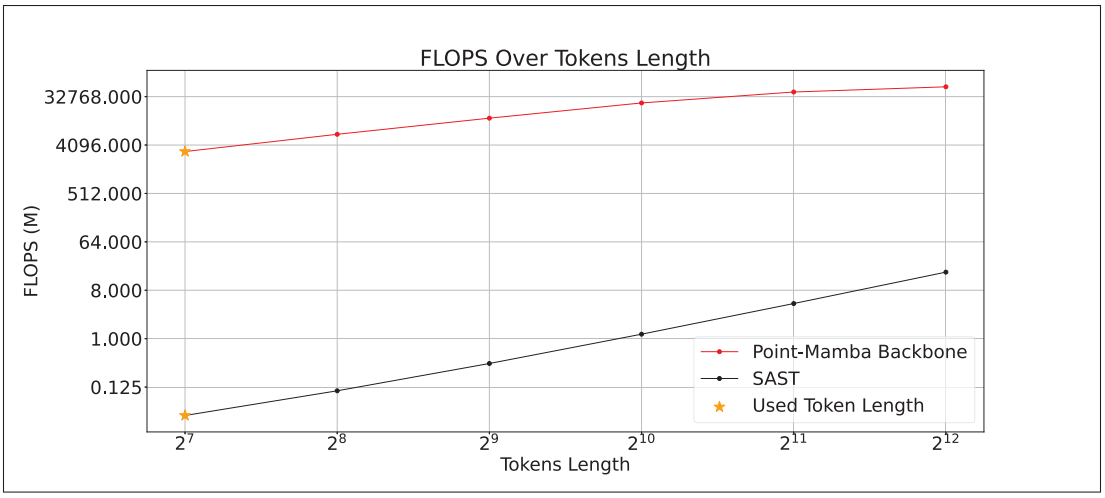

Figure-A II-3 FLOPS Over Tokens Length. The horizontal axis is scaled by $\log_2$ for better visualization

2. Additional Ablation Study

The Effect of HLT on Classification: In this section, we investigate the effect of the HLT strategy on the classification task. The results for HLT on the ObjectNN dataset are shown in Table II-1. As observed, the HLT strategy underperforms compared to SAST across all three settings (OBJ-BG, OBJ-ONLY, and PB-T50-RS), regardless of whether the model is trained from scratch or pretrained.

This performance gap highlights the limitations of the HLT strategy in tasks requiring global understanding, such as classification. Specifically, HLT processes high-level information from all eigenvectors simultaneously in a **single traversal order** (forward and backward), which is effective for segmentation tasks but may lead to insufficiently distinct feature representations for global classification. In contrast, SAST processes information from different eigenvectors in **separate traversals**, enabling better representation of high-level structures critical for classification tasks.

Number of Eigenvectors in HLT: Table II-2 evaluates the impact of varying the number of eigenvectors in our proposed HLT strategy on part segmentation performance using the

Table-A II-1 Object classification on ScanObjectNN. Accuracy (%) is reported

Methods	Backbone	OBJ-BG	OBJ-ONLY	PB-T50-RS
Training from scratch				
Ours (HLT)	Mamba	90.87	90.53	86.22
Ours (SAST)	Mamba	**92.25**	**91.39**	**87.30**
Training from pretrained				
Ours (HLT)	Mamba	91.80	91.42	87.52
Ours (SAST)	Mamba	**94.32**	**91.91**	**89.10**

ShapeNetPart dataset, considering both *training from scratch* and *training from pretrained weights*.

In both scenarios, the segmentation accuracy, measured by the mean Intersection over Union (mIoU), demonstrates that the number of eigenvectors directly influences performance. The accuracy peaks at four eigenvectors, achieving 85.9% (scratch) and 86.1% (pretrained), as this setting provides an optimal balance for spatial encoding.

When the number of eigenvectors is low, the model fails to partition points accurately, resulting in unrelated points being grouped into the same segment. Conversely, when the number of eigenvectors is high, the performance decreases slightly due to redundancy and noise. Higher-order eigenvectors encode finer details or localized variations, which may not align with meaningful segmentation. This can lead to overfitting or confusion between closely related parts.

Table-A II-2 Part segmentation on ShapeNetPart

Methods	Param. (M)	Eigenvectors	mIoU
Training from scratch			
Ours (HLT)	12.3	3	85.6
Ours (HLT)	12.3	**4**	**85.9**
Ours (HLT)	12.3	5	85.9
Ours (HLT)	12.3	6	85.8
Training from pretrained			
Ours (HLT)	12.3	3	85.8
Ours (HLT)	12.3	**4**	**86.1**
Ours (HLT)	12.3	5	86.0
Ours (HLT)	12.3	6	85.8

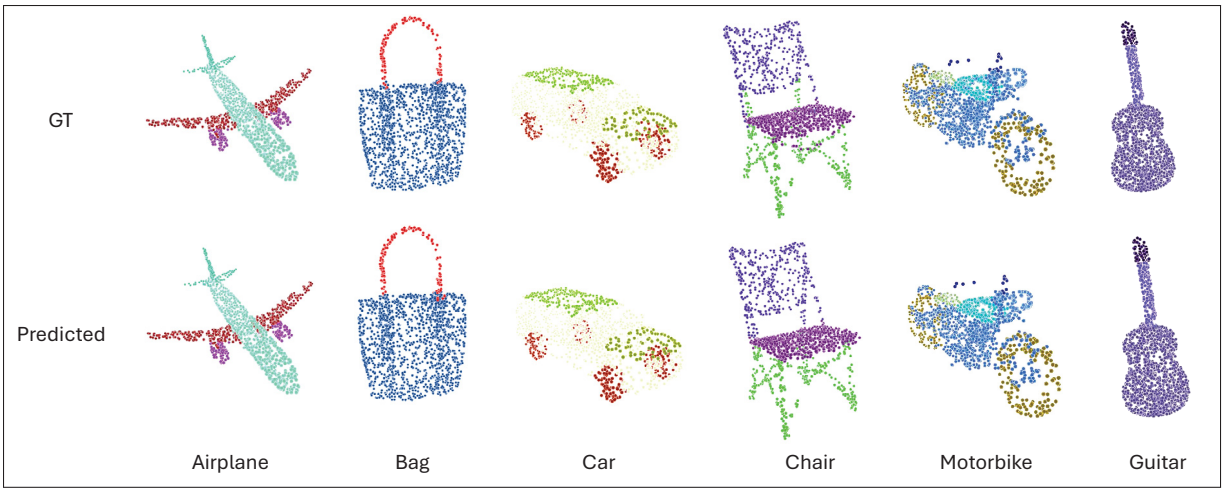

Figure-A II-4 The qualitative results of part segmentation of our HLT method on ShapeNetPart dataset

3. Additional Visualization

Segmentation Results. Figure II-4 provides results for six object categories ("Airplane," "Bag," "Car," "Chair," "Motorbike," and "Guitar) obtained by our HLT method. In this figure, each point is color-coded based on its class label. The comparison between the ground truth (GT) and the predicted segmentation demonstrates the outstanding performance of our method, as well as its ability to capture fine-grained details.

Dataset Challenges and Ground Truth Anomalies in ShapeNetPart. The ShapeNetPart dataset is widely recognized as a challenging benchmark for 3D point cloud segmentation. Upon further investigation of the dataset, we observed certain inconsistencies and inaccuracies in the provided GT annotations. As illustrated in Figure II-5, our method exhibits a visually better segmentation than the ground truth.

Such discrepancies in the ground truth highlight potential limitations in the dataset itself, which poses a challenge for both training and evaluation. This phenomenon also provides an explanation for the observation that recent state-of-the-art methods (as listed in Table 3.3 of the main paper) achieve similar mIOU performance on this dataset, with only marginal

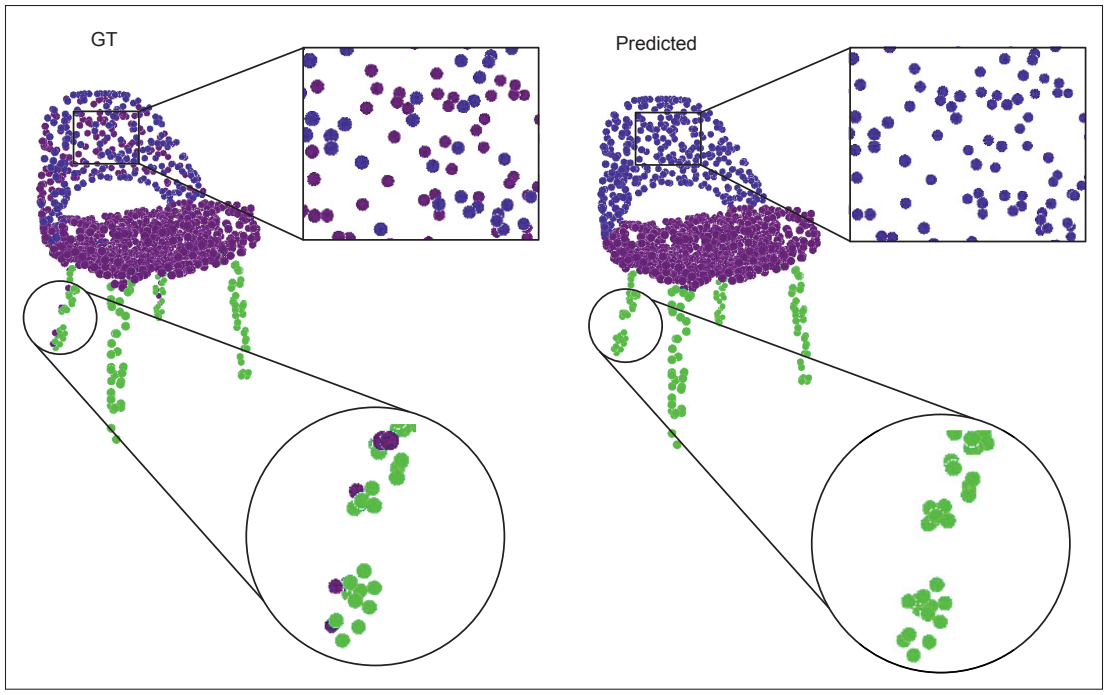

Figure-A II-5 An example illustrating inconsistencies in the ground truth (GT) annotations. The predicted segmentation (right) is geometrically more accurate and consistent compared to the GT (left)

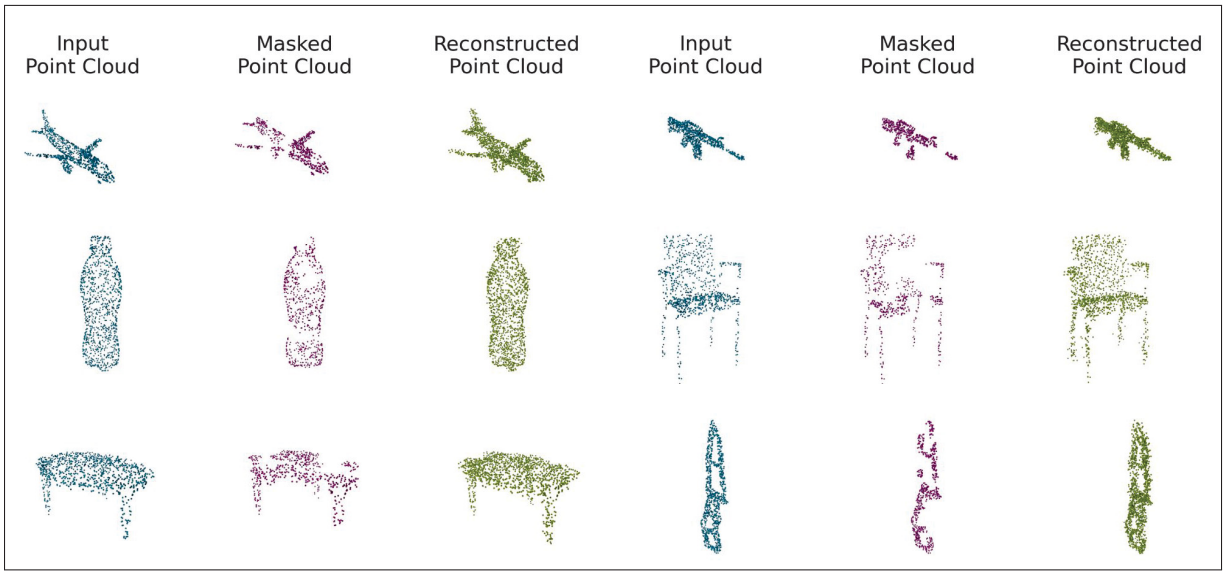

Figure-A II-6 Reconstruction results on the ShapeNet dataset

improvements. The inherent noise and errors in the ground truth annotations make it difficult for methods to demonstrate significant gains in segmentation quality.

Despite these challenges, our method still achieves competitive performance while maintaining robust predictions that align closely with the underlying geometric features of the objects.

Reconstruction Results. Figure II-6 showcases the reconstruction capability of Masked Autoencoders (MAEs) on point cloud data using the ShapeNet dataset. The figure consists of three columns for each sample, illustrating the progression from the input point cloud to the final reconstructed result.

The first column, “Input Point Cloud”, represents the original point cloud data, providing a complete view of the object before any masking or processing. This serves as a reference for evaluating the quality of the reconstruction. The second column, displays the same point cloud after a portion of the data has been masked out. The visible points indicate the sparse information available to the model during the reconstruction phase.The third column, “Reconstructed Point Cloud”, demonstrates the MAE’s ability to predict and restore the masked regions, resulting in a nearly complete reconstruction that aligns closely with the original structure.

These results underline the effectiveness of MAEs in capturing and reconstructing geometric details from incomplete data, making them well-suited for extracting meaningful features.

APPENDIX III

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED TEST-TIME ADAPTATION IN POINT CLOUDS: LEVERAGING SAMPLING VARIATION WITH WEIGHT AVERAGING

1. Implementation

We used PyTorch to implement the core functionalities of our approach. The codebase is structured into two main parts: *Pretrain* and *adaptation*.

Pretrain. We begin by focusing on the initial pretraining phase of the base models (Point-MAE, PointNet, DGCNN, and CurveNet). During this phase, we pretrain the backbones in a fully supervised manner, following the standard definition of Test-Time Adaptation (TTA). The pretraining is conducted on clean datasets such as ModelNet, ShapeNet, and ScanObjectNN. This phase ensures that the models are adequately prepared for the subsequent adaptation steps.

Adaptation. After completing the pretraining phase, we transition to the adaptation stage. In this phase, we only update the Normalization Layers of the models using our method, which is built upon the TENT algorithm. By selectively adapting the normalization layers, we efficiently adjust the models to handle corrupted data without requiring full retraining. This targeted approach not only reduces computational costs but also enhances the model's ability to generalize to different data distributions. The results of this adaptation phase are directly reflected in the experimental findings presented in this paper.

To ensure complete transparency and reproducibility of our results, we have made all relevant materials publicly available. This includes:

- The full source code for both *Pretrain* and *Adaptation* phases;
- All log files containing the detailed results of our experiments;
- Pretrained the base models for all backbones.

All these resources can be accessed through our *code*. This repository includes everything needed to understand our code, covering all aspects of the implementations and the reproduction of the results. Moreover, the specific hyperparameters used for all backbones are comprehensively outlined in Table III-1.

2. Resource Overhead

Time. Our method builds on the TENT algorithm but extends it by introducing multiple sampling variations $\mathcal{P}_v$ during TTA. While there may be concerns about potential resource overhead, particularly regarding execution time, our method is designed to run in parallel for all $\mathcal{P}_v$. This parallelization allows the model to adapt independently for each variation, significantly reducing time costs compared to a sequential approach. The comparison between parallel and sequential adaptations is detailed in the Supplementary Material Section 3. To quantify the computational cost, we evaluated our method on the PointNet backbone, comparing it directly with TENT. Using $N^V = 6$, the average adaptation time for TENT is approximately **21 ms**, whereas our method required around **26 ms**. This marginal difference indicates that the parallelization ensures minimal resource overhead, making our approach highly efficient even with multiple sampling variations.

Memory. Given that our method adapts only the learnable parameters of the normalization layers, keeping the other weights frozen and shared, it involves a limited number of parameters

Table-A III-1 Hyperparameters

Backbone	Config	Value
All	Optimizer	AdamW
All	learning rate	1e-3
All	Weight decay	0.0
All	Momentum	$\beta = 0.9$
All	Iteration	1
All	FPS	512, 1024
PointMAE-PointNet	Batch size	128
DGCNN-CurveNet	Batch size	16, 64

in the adaptation process. For instance, in the PointNet backbone, there are approximately 3,500,000 parameters, and we adapt only around 12,000 parameters, which constitutes **0.3%** of all parameters. Consequently, when using $N^V = 6$, the memory resource overhead is approximately **1.8%** of the whole backbone, which is negligible.

3. Additional Experiments

Parallel vs Sequential WA. We investigated two different strategies to handle model adaptation across multiple variations:

- **Parallel Mode**: After adapting the model using each variation $\mathcal{P}_V$, the model is reset to its initial state before the next adaptation begins. The weights adapted from each variation θ_v are stored individually. The final model weights θ_{avg} are then calculated by averaging all the adapted weights across the variations. This approach enables the model to process each variation independently, offering faster adaptation.
- **Sequential Mode**: In this method, the model does not reset after each adaptation. Instead, the adapted model from one variation serves as the starting point for the next variation. This results in iterative adaptation, where the model progressively refines its parameters after each variation $\mathcal{P}_V$, creating a cumulative adaptation process. The final model weights θ_{avg} are then calculated by averaging all the adapted weights across the variations.

As shown in Figure III-1, both modes offer similar performance as the number of variations N^V increases. However, since speed and efficiency are critical for TTA, we select the parallel mode, as it allows for faster processing by adapting the model simultaneously across all variations. This experiment was conducted using the Point-MAE backbone on the ModelNet-40C dataset.

Integration of Jitter and Sampling Variation. In this experiment, we investigate the effect of combining jitter augmentation with the sampling variation as a new strategy $\mathcal{P}$ to generate model diversity in our method. As seen in Figure III-2, jitter is selected for this combination because it shows the best performance among other augmentation techniques (as noted in Figure 3 of the main paper). However, while combining jitter with sampling variation yields better

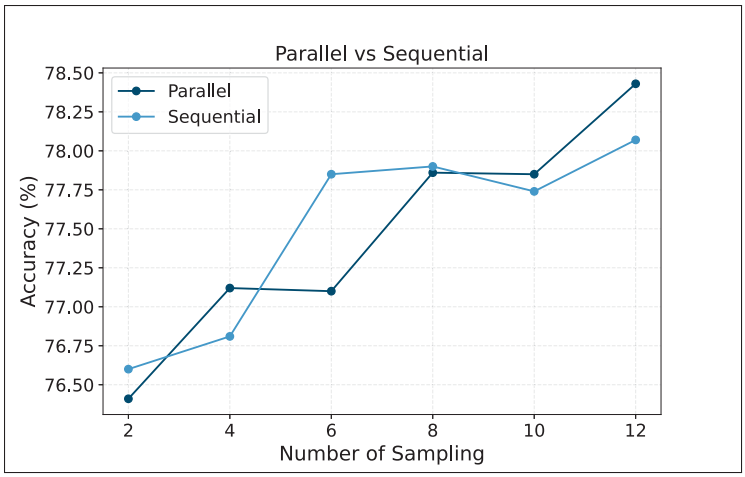

Figure-A III-1 Impact of Parallel vs Sequential on Accuracy

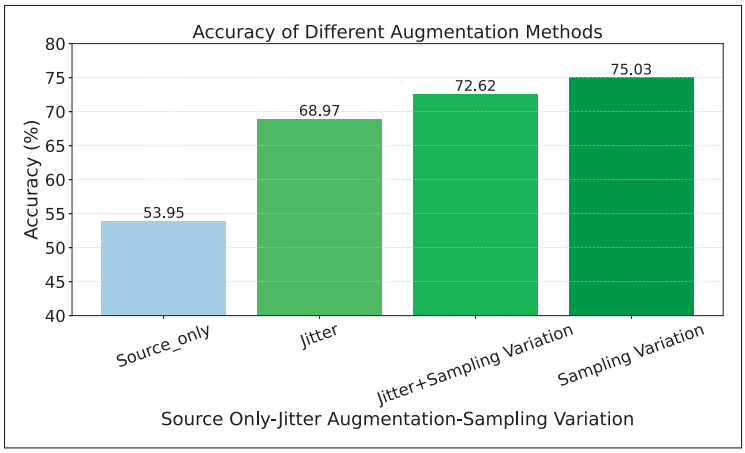

Figure-A III-2 Comparison between Sampling Variation and Different Augmentations

results compared to using jitter alone, it does not surpass the performance of our method when using sampling variation exclusively.

Impact of Batch Normalization and Layer Normalization. In Table III-2, we investigate the effect of updating Batch Normalization (BN) layers only versus updating both Batch Normalization (BN) and Layer Normalization (LN) layers during test-time adaptation. The results demonstrate that updating only BN layers significantly improves performance over the

Table-A III-2 Top-1 Classification Accuracy (%) for all distribution shifts in the ModelNet-40C dataset

	Method	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Mean
Point-MAE	Source-Only	66.6	59.1	7.2	31.8	74.6	67.7	69.8	59.3	75.1	74.4	38.0	53.7	70.0	38.6	23.4	53.9
	Ours (BN)	85.4	84.7	29.9	74.8	87.1	80.9	82.3	85.1	88.4	82.4	67.9	83.9	80.7	55.7	54.8	74.9
	Ours (BN & LN)	85.0	83.9	33.0	74.6	87.0	80.9	82.3	85.1	88.0	82.7	66.9	84.0	80.5	56.2	55.3	75.0

Source-Only baseline. Furthermore, updating both BN and LN layers leads to a slight but consistent improvement across most corruptions, resulting in a higher mean accuracy (75.0%) compared to updating BN layers alone (74.9%). The experiment was conducted with a batch size of 128 and 5 iterations, using PointMAE as the backbone. The dataset used was ModelNet40-C, and weight averaging was performed in parallel mode.

Evaluation on the CurveNet Backbone. In order to further assess the robustness and generalizability of our method, we conducted additional experiments using a different backbone architecture, CurveNet, on the ModelNet-40C dataset. The results are summarized in Table III-3. As can be seen, our method demonstrates consistent improvements over baseline approaches, achieving a mean accuracy of **76.2%**, which is notably higher than TENT's accuracy of **75.3%**. The improvements are particularly significant in corruptions like *occlusion* (56.3%), and *lidar* (56.4%), where our method consistently outperforms the other approaches.

Table-A III-3 Top-1 Classification Accuracy (%) for all distribution shifts in the ModelNet-40C dataset

	Method	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Mean
CurveNet	Source-Only	67.3	77.1	7.6	47.6	70.1	78.6	80.6	79.2	88.1	77.0	68.8	78.6	77.6	35.5	26.5	64.0
	SHOT (Liang *et al.*, 2020)	75.5	78.3	22.4	61.1	68.7	72.9	69.1	62.3	64.7	39.2	31.0	30.6	27.1	10.7	8.0	48.1
	DUA (Mirza *et al.*, 2022)	81.5	84.3	27.5	71.1	81.3	82.6	84.5	85.5	89.0	82.1	76.9	85.2	81.7	46.6	45.8	73.7
	PL (Lee *et al.*, 2013)	79.5	84.0	29.5	72.6	82.7	82.0	83.1	85.9	88.7	81.2	78.9	85.3	81.6	52.8	52.5	74.7
	TENT (Wang *et al.*, 2021a)	80.9	84.9	29.0	73.9	83.8	83.1	**85.5**	85.2	**89.3**	**83.0**	79.8	**85.8**	**83.6**	50.2	51.0	75.3
	Ours	**80.9**	**85.6**	**30.0**	**74.7**	**83.9**	**83.2**	84.3	**86.1**	88.7	82.8	**81.2**	85.7	82.7	**56.3**	**56.4**	**76.2**

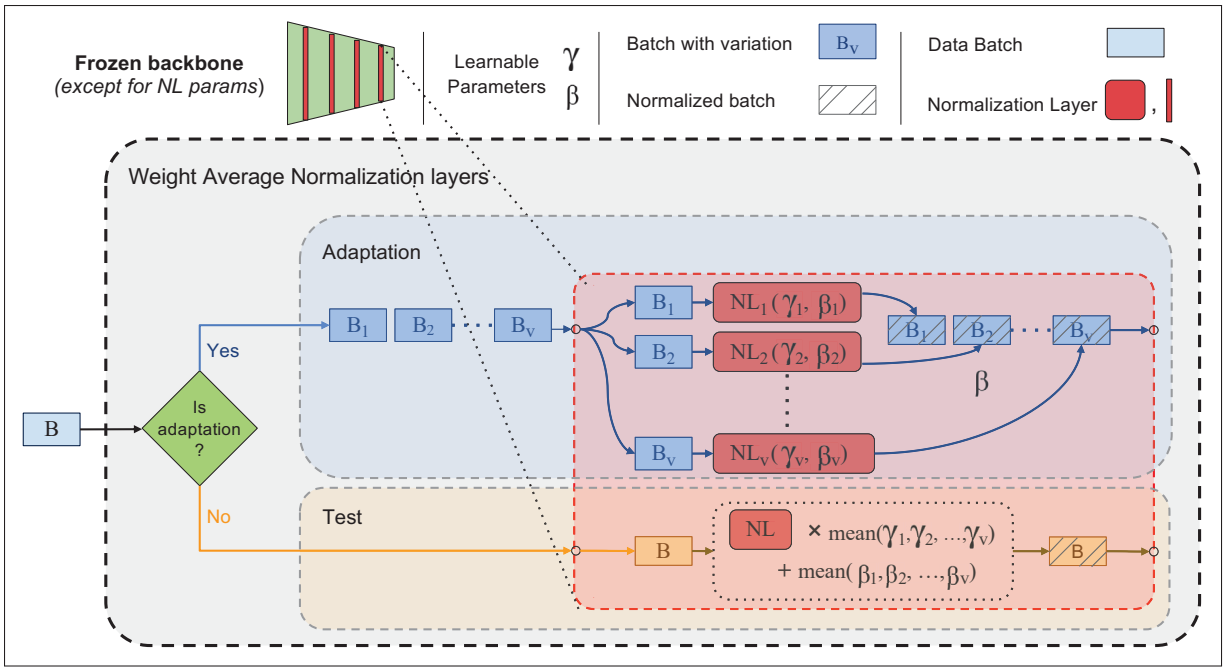

Figure-A III-3 Detailed diagram of our method's Parallel mode

Efficient Parallel Implementation. Figure III-3 illustrates the detailed implementation of our method in parallel mode. When adapting only the normalization layers, we handle N^V variations $\mathcal{P}_v$ in parallel. For each sampling variation $\mathcal{P}_v$, our method adapts the corresponding normalization layers independently. This means that the weights of the rest of the network (the majority) are shared across variations, reducing the memory overhead significantly. As shown in Figure III-3, we construct a "Weight Average Normalization Layer," which comprises the N^V individual normalization layers.

During adaptation, all the variations are processed through their respective normalization layers. After adaptation, the normalization layer parameters γ and β are then averaged to produce the final set of normalized parameters. With this technique, we avoid saving or reloading the backbone weights for each variation, which leads to memory efficiency. For example, in the PointNet backbone, the normalization layers constitute only 0.3% of the total network parameters. Hence, by adapting only these layers, we reduce the memory resource overhead to a mere 1.8% when using $N^V = 6$, compared to the 500% memory overhead of the naive implementation.

APPENDIX IV

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED SMART-PC: SKELETAL MODEL ADAPTATION FOR ROBUST TEST-TIME TRAINING IN POINT CLOUDS

1. Implementation

Our approach was implemented using PyTorch, with the codebase organized into two main components: *Pretrain* and *Adaptation*.

Pretrain. We start with the initial pretraining phase of the base models (Point-MAE). In this phase, the backbone is pretrained with a classification branch in a fully supervised manner and a skeleton branch in a self-supervised manner. The pretraining is performed on clean datasets such as ModelNet, ShapeNet, and ScanObjectNN, ensuring the models are well-prepared for the subsequent adaptation steps.

Adaptation. After completing the pretraining phase, we transition to the adaptation stage, which consists of two modes: **online** and **standard**. In the **standard adaptation mode**, all model parameters are updated using the skeleton loss, allowing the model to comprehensively adjust to the target data. In the **online adaptation mode**, we employ two distinct strategies. The first strategy involves adapting only the statistical parameters of the BatchNorm layers (e.g., running mean and variance) without backpropagation. This approach significantly reduces computational costs, enabling our method to achieve higher Frames per Second (FPS) and making it suitable for real-time applications. The second strategy involves updating all model parameters using the skeleton loss, similar to the standard mode. These flexible adaptation strategies highlight the efficiency and scalability of our method, catering to both real-time and high-accuracy requirements. The code will be published upon the acceptance of the paper.

2. Datasets

ModelNet-40C. ModelNet-40C (Sun *et al.*, 2022) serves as a robustness benchmark for point cloud classification, designed to evaluate the ability of architectures to handle real-world distribution shifts. It extends the original ModelNet-40 test set by introducing 15 common corruption types, grouped into three categories: transformations, noise, and density variations. These corruptions simulate practical challenges like sensor errors and LiDAR noise, offering a realistic assessment of model performance under diverse and challenging conditions.

ShapeNet-C. ShapeNetCore-v2 (Chang *et al.*, 2015) is a widely used dataset for point cloud classification, containing 51,127 3D shapes spanning 55 categories. It is partitioned into three subsets: 70% for training, 10% for validation, and 20% for testing. To evaluate the robustness of models under real-world conditions, (Mirza *et al.*, 2023) augmented the test set with 15 types of corruptions, mirroring those in ModelNet-40C. These corruptions, created using the open-source implementation from (Sun *et al.*, 2022), resulted in a modified version of the dataset known as ShapeNet-C.

ScanObjectNN-C. ScanObjectNN (Uy *et al.*, 2019b) is a real-world dataset for point cloud classification, comprising 2,309 training samples and 581 testing samples across 15 categories. To assess robustness, (Mirza *et al.*, 2023) applied 15 unique corruption types to the test set, using the approach described in (Sun *et al.*, 2022). The resulting modified dataset is referred to as ScanObjectNN-C.

3. Detailed Reuslts

In this section, the performance of our method is presented in Table IV-1, Table IV-2, and table IV-3 showcasing detailed results for each corruption across all datasets. Our method consistently outperforms previous approaches in source-only (without adaptation), standard adaptation, and online adaptation modes. In most corruption scenarios, SMART-PC achieves higher accuracy compared to prior methods, demonstrating its robustness and effectiveness in

Table-A IV-1 Top-1 Classification Accuracy (%) for all distribution shifts in ModelNet40-C. All results use a Point-MAE backbone trained on clean data and adapted to OOD test samples with batch size 1. "*" indicates reproduced results; "†" indicates adaptation using BN statistics without backpropagation

Corruptions:	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Avg.
Org-SO	66.6	59.2	7.2	31.8	74.6	67.7	69.8	59.3	75.1	74.4	38.0	53.7	70.0	**38.6**	23.4	54.0
MATE-SO	59.7	51.3	**28.2**	55.3	71.5	57.4	60.7	**65.2**	**77.4**	67.1	30.2	62.3	61.9	37.2	19.9	53.7
SMART-PC-SO	**81.8**	**79.5**	13.6	**65.4**	**84.3**	**75.7**	**77.8**	62.0	65.9	**73.4**	**42.7**	**69.1**	**73.8**	36.3	**24.4**	**61.7**
DUA	65.0	58.5	14.7	48.5	68.8	62.8	63.2	62.1	66.2	68.8	46.2	53.8	64.7	41.2	36.5	54.7
TTT-Rot	61.3	58.3	34.5	48.9	66.7	63.6	63.9	59.8	68.6	55.2	27.3	54.6	64.0	40.0	29.1	53.0
SHOT	29.6	28.2	9.8	25.4	32.7	30.3	30.1	30.9	31.2	32.1	22.8	27.3	29.4	20.8	18.6	26.6
T3A	64.1	62.3	33.4	65.0	75.4	63.2	66.7	57.4	63.0	72.7	32.8	54.4	67.7	39.1	18.3	55.7
TENT	29.2	28.7	10.1	25.1	33.1	30.3	29.1	30.4	31.5	31.8	22.7	27.0	28.6	20.7	19.0	26.5
MATE-Standard*	69.8	61.8	**18.9**	63.9	72.5	64.0	66.0	**74.0**	**80.8**	71.0	36.7	69.2	66.3	**38.4**	**29.9**	58.9
SMART-PC-Standard	**82.4**	**80.1**	12.0	**67.1**	**84.5**	**76.0**	**78.6**	67.3	72.9	**73.3**	**43.9**	**72.6**	**73.5**	37.4	24.8	**63.1**
MATE-Online*	80.6	79.5	20.7	71.5	82.6	78.1	80.7	78.1	**86.6**	**79.6**	54.9	78.4	77.4	**45.4**	**49.6**	69.6
SMART-PC-Online†	85.0	83.2	31.6	77.6	**85.9**	79.2	80.8	77.8	79.3	77.8	60.3	80.6	**76.8**	45.2	40.4	**70.8**
SMART-PC-Online	**85.4**	**84.0**	**49.4**	**79.7**	85.7	**80.1**	**81.3**	**81.7**	82.7	78.3	**60.5**	**82.6**	76.7	44.4	41.8	**72.9**

Table-A IV-2 Top-1 Classification Accuracy (%) for all distribution shifts in the ShapeNet-C dataset. All results are based on the PointMAE backbone trained on a clean training set and adapted to the OOD test set with a batch size of 1. Results marked with * indicate reproduced outcomes, while "†" denotes adaptation using BatchNorm statistical parameters without backpropagation

Corruptions:	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Avg.
Org-SO	77.4	71.8	8.6	54.4	77.9	75.5	76.0	85.3	76.5	80.5	57.1	85.1	76.0	11.0	7.1	61.3
MATE-SO	69.7	63.3	2.1	50.6	71.1	70.2	72.1	**85.9**	**77.8**	75.6	44.0	**85.4**	70.3	7.0	3.1	56.5
SMART-PC-SO	**80.6**	**78.5**	**11.4**	**61.3**	**81.6**	**81.1**	**81.5**	84.9	74.4	**81.1**	**64.1**	85.0	**79.9**	**11.8**	**10.0**	**64.5**
DUA	76.1	70.1	14.3	60.9	76.2	71.6	72.9	80.0	83.8	77.1	57.5	75.0	72.1	11.9	12.1	60.8
TTT-Rot	74.6	72.4	23.1	59.9	74.9	73.8	75.0	81.4	82.0	69.2	49.1	79.9	72.7	14.0	12.0	60.9
SHOT	44.8	42.5	12.1	37.6	45.0	43.7	44.2	48.4	49.4	45.0	32.6	46.3	39.1	6.2	5.9	36.2
T3A	70.0	60.5	6.5	40.7	67.8	67.2	68.5	79.5	79.9	72.7	42.9	79.1	66.8	7.7	5.6	54.4
TENT	44.5	42.9	12.4	38.0	44.6	43.3	44.3	48.7	49.4	45.7	34.8	48.6	43.0	10.0	10.9	37.4
MATE-Standard	77.8	74.7	4.3	**66.2**	78.6	76.3	75.3	**86.1**	**86.6**	79.2	56.1	84.1	76.1	**12.3**	**13.1**	63.1
SMART-PC-Standard	**80.8**	**78.9**	**8.9**	60.4	**81.8**	**81.1**	**81.7**	84.8	78.4	**80.8**	**63.7**	**84.9**	**79.8**	11.5	8.8	**64.4**
MATE-Online*	**81.5**	78.6	**40.9**	**75.9**	**81.6**	79.7	80.1	**84.9**	**85.9**	**81.8**	70.8	**85.1**	79.0	**14.2**	**16.6**	**69.1**
SMART-PC-Online†	80.4	78.7	21.0	72.7	80.9	80.9	**80.6**	82.5	78.3	80.9	70.1	82.5	79.0	10.5	9.7	65.9
SMART-PC-Online	81.2	**80.5**	28.9	74.3	81.2	**80.7**	80.5	83.1	81.0	80.4	**73.2**	82.8	**79.0**	10.0	10.2	67.1

handling distribution shifts across ModelNet40-C, ShapeNet-C, and ScanObjectNN-C. The improvements are particularly significant in challenging corruption types, highlighting the advantages of leveraging skeletal representations for test-time training.

4. More Visualizations

In Figure IV-1, we present several 3D objects showcasing the original point clouds (blue dots) and their corresponding skeletal spheres. Each skeletal sphere is defined by a skeleton point (the

Table-A IV-3 Top-1 Classification Accuracy (%) for all distribution shifts in the ScanObjectNN-C dataset. All results are based on the PointMAE backbone trained on a clean training set and adapted to the OOD test set with a batch size of 1. Results marked with * indicate reproduced outcomes, while "†" denotes adaptation using BatchNorm statistical parameters without backpropagation

Corruptions:	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Avg.
Org-SO	21.7	18.8	16.9	18.4	22.2	**46.0**	47.0	**72.1**	**69.4**	**48.9**	35.6	**73.0**	**49.4**	6.7	9.3	37.0
MATE-SO	20.3	32.2	**18.9**	21.2	20.5	35.6	36.7	69.9	66.6	38.9	28.7	70.4	39.4	8.3	9.8	34.5
SMART-PC-SO	**26.7**	**37.7**	16.9	**21.3**	**27.2**	44.2	**48.9**	69.5	56.3	48.5	**43.2**	72.3	48.0	**8.4**	**11.0**	**38.7**
DUA*	30.5	40.1	10.2	23.6	29.9	43.7	46.1	68.3	66.3	48.5	38.9	68.7	48.4	8.6	8.1	38.7
SHOT*	30.2	34.1	16.2	22.6	22.6	32.4	32.1	45.5	45.0	34.5	29.3	47.8	36.2	7.1	8.1	29.6
TENT*	29.5	31.6	17.6	24.8	27.2	31.0	32.4	40.7	35.0	30.2	26.6	36.6	29.3	10.5	12.4	27.7
MATE-Standard*	27.5	29.4	14.3	**22.2**	25.6	40.8	42.0	**73.7**	**63.2**	45.1	35.3	**73.3**	45.3	7.1	9.3	36.9
SMART-PC-Standard	**27.5**	**39.1**	**19.3**	21.5	**29.8**	**44.2**	**48.9**	68.3	60.2	**49.4**	**45.4**	70.1	**49.1**	**8.4**	**12.2**	**39.6**
MATE-Online*	33.0	44.1	13.3	25.3	29.1	36.8	37.7	73.3	65.2	37.2	31.3	72.6	40.6	7.2	7.4	36.9
SMART-PC-Online†	39.4	**54.6**	19.6	**40.1**	40.3	54.4	55.9	**73.3**	**69.5**	55.1	50.1	**74.9**	57.8	6.9	7.9	**46.7**
SMART-PC-Online	**42.3**	53.7	**23.8**	37.5	**41.6**	**57.0**	57.0	71.1	68.8	**57.3**	**53.4**	72.1	**58.2**	**8.4**	**8.4**	**47.4**

center of the sphere) and its associated radius, which represents the local geometric structure around the skeleton point. The spheres collectively capture the essential geometric and structural features of the objects.

This visualization demonstrates the capability of our model to learn meaningful and compact representations of 3D shapes through skeletal abstraction. The skeleton effectively captures the underlying structure while filtering out high-frequency noise, enabling the model to focus on the fundamental geometry of the objects. The diversity of shapes in this figure—ranging from guitars and airplanes to furniture—highlights the robustness of our approach in generalizing across different object classes. These visualizations also provide insight into how the skeletal abstraction simplifies complex point cloud data into manageable representations, facilitating better performance in downstream tasks such as classification and adaptation under challenging conditions. This compact representation ensures that the geometric structure is preserved, even when the original point clouds are corrupted or noisy.

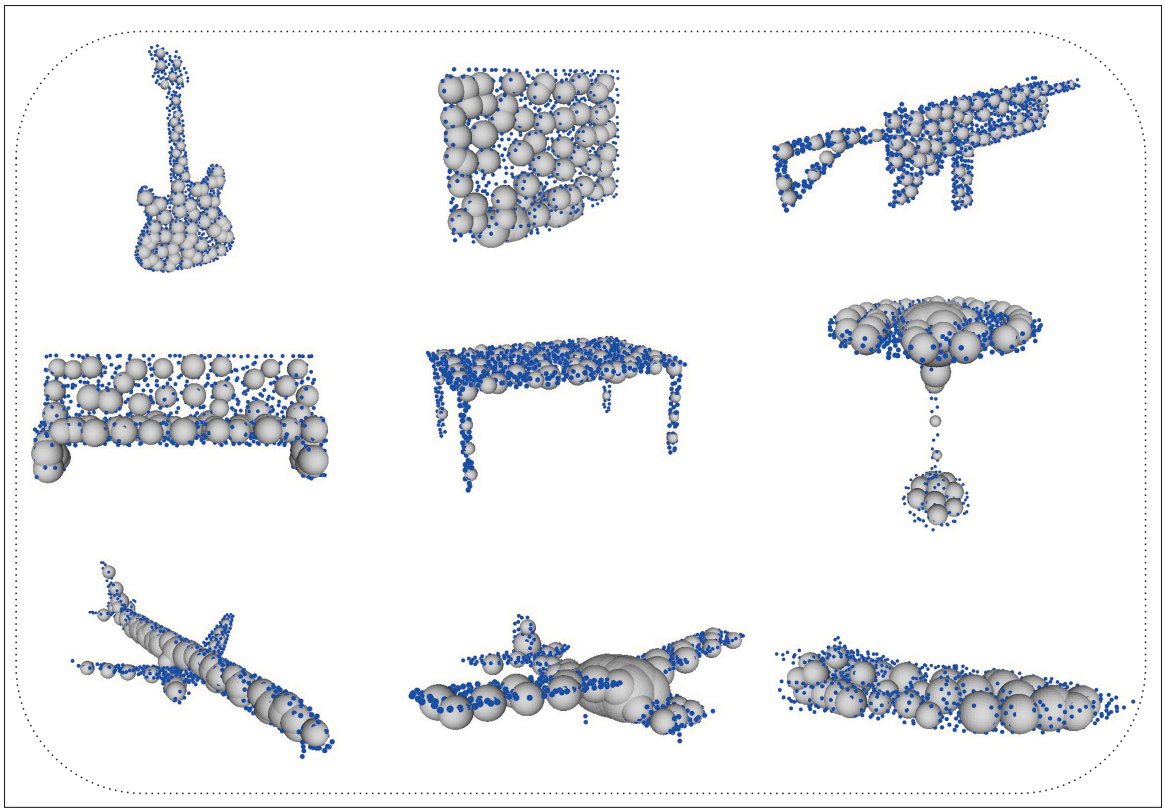

Figure-A IV-1 Visualization of 3D objects with original point clouds (blue dots) and their corresponding skeletal spheres

BIBLIOGRAPHY

Afham, M., Dissanayake, I., Dissanayake, D., Dharmasiri, A., Thilakarathna, K. & Rodrigo, R. (2022). CrossPoint: Self-supervised cross-modal contrastive learning for 3D point cloud understanding. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9902–9912.

Atzmon, M., Maron, H. & Lipman, Y. (2018). Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*.

Bahri, A., Desrosiers, C. & Swamy, M. N. (2024a). Test-Time Prompt Tuning for Vision Transformers. *arXiv preprint arXiv:2403.18944*.

Bahri, A., Yazdanpanah, M., Noori, M., Cheraghalikhani, M., Hakim, G. A. V., Osowiechi, D., Beizaee, F., Ayed, I. B. & Desrosiers, C. (2024b). GeoMask3D: Geometrically Informed Mask Selection for Self-Supervised Point Cloud Learning in 3D. *arXiv preprint arXiv:2405.12419*.

Bahri, A., Yazdanpanah, M., Noori, M., Cheraghalikhani, M., Hakim, G. A. V., Osowiechi, D., Beizaee, F., Ayed, I. B. & Desrosiers, C. (2024c). GeoMask3D: Geometrically Informed Mask Selection for Self-Supervised Point Cloud Learning in 3D. *arXiv preprint arXiv:2405.12419*.

Bahri, A., Yazdanpanah, M., Dastani, S., Noori, M., Hakim, G. A. V., Osowiechi, D., Beizaee, F., Ayed, I. B. & Desrosiers, C. (2025a). SMART-PC: Skeletal model adaptation for robust test-time training in point clouds. *arXiv preprint arXiv:2505.19546*.

Bahri, A., Yazdanpanah, M., Noori, M., Dastani, S., Cheraghalikhani, M., Hakim, G. A. V., Osowiechi, D., Beizaee, F., Ben Ayed, I. & Desrosiers, C. (2025b). Spectral informed mamba for robust point cloud processing. *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 11799–11809.

Bahri, A., Yazdanpanah, M., Noori, M., Dastani, S., Cheraghalikhani, M., Osowiechi, D., Hakim, G. A. V., Beizaee, F., Ayed, I. B. & Desrosiers, C. (2025c). Spectral Informed Mamba for Robust Point Cloud Processing. *arXiv preprint arXiv:2503.04953*.

Bahri, A., Yazdanpanah, M., Noori, M., Oghani, S. D., Cheraghalikhani, M., Osowiechi, D., Beizaee, F., Hakim, G. A. V., Ben Ayed, I. & Desrosiers, C. (2025d). Test-time adaptation in point clouds: Leveraging sampling variation with weight averaging. *Proceedings of the Winter Conference on Applications of Computer Vision*, pp. 266–275.

Beltagy, I., Peters, M. E. & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Bengio, Y. & LeCun, Y. (2007). Scaling Learning Algorithms Towards AI. In *Large Scale Kernel Machines*. MIT Press.

Boudiaf, M., Mueller, R., Ben Ayed, I. & Bertinetto, L. (2022). Parameter-free online test-time adaptation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8344–8353.

Cha, S., Kim, D., Cho, M.-H. & Ceylan, D. (2021). SWAD: Stochastic Weight Averaging in Domain Generalization. *Advances in Neural Information Processing Systems (NeurIPS)*.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H. et al. (2015). ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*.

Chen, T. et al. (2024). PointGPT: Auto-regressively Generative Pre-training from Point Clouds. *arXiv preprint arXiv:2401.02416*.

Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *International conference on machine learning*, pp. 1597–1607.

Chen, X. & He, K. (2021). Exploring simple siamese representation learning. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758.

Cheng, S., Chen, X., He, X., Liu, Z. & Bai, X. (2021). PRA-Net: Point relation-aware network for 3d point cloud analysis. *IEEE Transactions on Image Processing*, 30, 4436–4448.

Child, R., Gray, S., Radford, A. & Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Choi, H. I., Choi, S. W. & Moon, H. P. (1997). Mathematical theory of medial axis transform. *pacific journal of mathematics*, 181(1), 57–88.

Chung, F. R. (1997). *Spectral graph theory*. American Mathematical Soc.

Courant, R. & Hilbert, D. (2008). *Methods of mathematical physics: partial differential equations*. John Wiley & Sons.

Dastani, S., Bahri, A., Hakim, G. A. V., Yazdanpanah, M., Noori, M., Osowiechi, D., Barbeau, S., Ayed, I. B., Lombaert, H. & Desrosiers, C. (2025). TRUST: Test-Time Refinement using Uncertainty-Guided SSM Traverses. *arXiv preprint arXiv:2509.22813*.

Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S. Y. et al. (2023). Objaverse-XL: A universe of 10M+ 3D objects. *arXiv preprint arXiv:2307.05663*.

Deng, X., Zhang, W., Ding, Q. & Zhang, X. (2023). Pointvector: a vector representation in point cloud analysis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9455–9465.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL*.

Dong, R., Qi, Z., Zhang, L., Zhang, J., Sun, J., Ge, Z., Yi, L. & Ma, K. (2022). Autoencoders as cross-modal teachers: Can pretrained 2d image transformers help 3d representation learning? *arXiv preprint arXiv:2212.08320*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Esser, P., Rombach, R. & Ommer, B. (2021). Taming transformers for high-resolution image synthesis. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883.

Fei, Y., Cao, Y., Wang, Y., Bai, X. & Liu, Y. (2023). Self-Supervised Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Gandelsman, Y., Sun, Y., Chen, Z., Li, X., Cisse, M., Darrell, T., Pathak, D. & Efros, A. A. (2022). Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*.

Gao, J., Zhang, J., Liu, X., Darrell, T., Shelhamer, E. & Wang, D. (2023). Back to the source: Diffusion-driven adaptation to test-time corruption. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11786–11796.

Golub, G. H. & Van Loan, C. F. (2013). *Matrix computations*. JHU press.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. (2016). *Deep learning*. MIT Press.

Gu, A. & Dao, T. (2024). Mamba: Linear-time sequence modeling with selective state spaces. *First conference on language modeling*.

Gu, A., Dao, T., Ermon, S., Rudra, A. & Ré, C. (2020). HiPPO: Recurrent Memory with Optimal Polynomial Projections. *Advances in Neural Information Processing Systems*, 33, 1474–1487.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A. & Ré, C. (2021). Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34, 572–585.

Gu, A., Goel, K. & Ré, C. (2022a). Efficiently Modeling Long Sequences with Structured State Spaces. *International Conference on Learning Representations*.

Gu, A., Goel, K. & Ré, C. (2022b). Parameterization of Multi-Input State-Space Models for Long-Range Sequence Modeling. *International Conference on Machine Learning*, pp. 7790–7802.

Gu, A., Dao, T., Erichson, N. B., Rudin, C. & Darrell, T. (2023). Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752*.

Guo, M.-H., Cai, Z.-N., Liu, T.-J., Mu, T.-J., Martin, R. R. & Hu, S.-M. (2021). PCT: Point Cloud Transformer. *International Conference on Computer Vision (ICCV)*.

Guo, Q., Liu, W., Zhao, Q., Ning, Z., Cheng, S. & Hu, J. (2024). Self-Supervised Learning of 3D Point Clouds: A Survey. *2024 36th Chinese Control and Decision Conference (CCDC)*, pp. 4464–4471.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L. & Bennamoun, M. (2020). Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12), 4338–4364.

Guo, Z., Zhang, R., Qiu, L., Li, X. & Heng, P.-A. (2023). Joint-mae: 2d-3d joint masked autoencoders for 3d point cloud pre-training. *arXiv preprint arXiv:2302.14007*.

Gupta, R., Faqir-Rhazoui, Y., Dao, T. & Gu, A. (2022). Simplifying Structured State Space Models for Sequence Modeling. *arXiv preprint arXiv:2208.04933*.

Hamdi, A., Giancola, S. & Ghanem, B. (2021a). MVTN: Multi-view transformation network for 3D shape recognition. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1–11.

Hamdi, A., Giancola, S. & Ghanem, B. (2021b). Voint cloud: Multi-view point cloud representation for 3D understanding. *arXiv preprint arXiv:2111.15363*.

Han, Z., Shang, M., Liu, Y.-S. & Zwicker, M. (2019a). View inter-prediction gan: Unsupervised representation learning for 3D shapes by learning global shape memories to support local view predictions. *Proceedings of the AAAI conference on artificial intelligence*, 33(01), 8376–8384.

Han, Z., Wang, X., Liu, Y.-S. & Zwicker, M. (2019b). Multi-angle point cloud-VAE: Unsupervised feature learning for 3D point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10441–10450.

Hasani, R., Lechner, M., Amini, A., Hsieh, J., Bruck, J. & Rus, D. (2022). Liquid Time-Constant Networks. *Nature Machine Intelligence*, 4, 852–863.

Hatem, A., Qian, Y. & Wang, Y. (2023a). Test-time adaptation for point cloud upsampling using meta-learning. *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1284–1291.

Hatem, M., Taleb, A., Xiao, Z. & Xu, Y. (2023b). Point-TTA: Test-Time Adaptation for 3D Point Cloud Registration. *IEEE/CVF International Conference on Computer Vision (ICCV)*.

He, K., Fan, H., Wu, Y., Xie, S. & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738.

He, K., Chen, X., Xie, S. et al. (2022). Masked Autoencoders Are Scalable Vision Learners. *CVPR*.

Hinton, G. E., Osindero, S. & Teh, Y. W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18, 1527–1554.

Hong, C.-Y., Chou, Y.-Y. & Liu, T.-L. (2023). Attention discriminant sampling for point clouds. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14429–14440.

Huang, S., Xie, Y., Zhu, S.-C. & Zhu, Y. (2021). Spatio-temporal self-supervised representation learning for 3D point clouds. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6535–6545.

Iwasawa, Y. & Matsuo, Y. (2021). Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, pp. 1–13.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D. & Wilson, A. G. (2018). Averaging Weights Leads to Wider Optima and Better Generalization. *Uncertainty in Artificial Intelligence (UAI)*.

Jaritz, M., Gu, J., Sun, H., He, S., Konolige, K. & Icardi, L. (2019). Multi-View PointNet for 3D Scene Understanding. *ICRA*.

Jiang, J., Lu, X., Zhao, L., Dazeley, R. & Wang, M. (2023). Masked autoencoders in 3d point cloud representation learning. *IEEE Transactions on Multimedia*, 27, 820–831.

Kanezaki, A., Matsushita, Y. & Nishida, Y. (2018). RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5010–5019.

Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X. & Jia, J. (2022). Stratified transformer for 3d point cloud segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8500–8509.

Landrieu, L. & Simonovsky, M. (2018). Large-scale point cloud semantic segmentation with superpoint graphs. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4558–4567.

Lee, D.-H. et al. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *Workshop on challenges in representation learning, ICML*, 3(2), 896.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S. & Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. *International conference on machine learning*, pp. 3744–3753.

Levi, D. & Avidan, S. (2023). EPiC: Benchmarking the Robustness of 3D Point Cloud Models. *CVPR*.

Li, J., Chen, B. M. & Lee, G. H. (2018a). SO-net: Self-organizing network for point cloud analysis. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9397–9406.

Li, P., Wang, B., Sun, F., Guo, X., Zhang, C. & Wang, W. (2015). Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics (TOG)*, 35(1), 1–16.

Li, Y., Wang, N., Shi, J., Hou, X. & Liu, J. (2018b). Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80, 109–117.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X. & Chen, B. (2018c). PointCNN: Convolution on x-transformed points. *Advances in neural information processing systems*, 31.

Li, Y., Cai, T., Zhang, Y., Chen, D. & Dey, D. (2022). What makes convolutional models great on long sequence modeling? *arXiv preprint arXiv:2210.09298*.

Liang, J., Hu, D. & Feng, J. (2020). Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation. *International Conference on Machine Learning (ICML)*, pp. 6028–6039.

Liang, T., Li, W., Xu, Y., Wang, Y. & Han, B. (2024). Point-Mamba: Point Cloud Understanding via State Space Models. *European Conference on Computer Vision*.

Lim, H., Kim, B., Choo, J. & Choi, S. (2023). TTN: A domain-shift aware batch normalization in test-time adaptation. *arXiv preprint arXiv:2302.05155*.

Lin, C., Li, C., Liu, Y., Chen, N., Choi, Y.-K. & Wang, W. (2021). Point2skeleton: Learning skeletal representations from point clouds. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4277–4286.

Liu, E., Shapira, L. et al. (2021a). PointGuard: Provable 3D Point Cloud Defense via Input Randomization. *ICCV*.

Liu, H., Cai, M. & Lee, Y. J. (2022). Masked discrimination for self-supervised learning on point clouds. *European Conference on Computer Vision*, pp. 657–675.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L. & Shazeer, N. (2018). Generating Wikipedia by Summarizing Long Sequences. *International Conference on Learning Representations*.

Liu, S., Wang, D., Zhu, Z., Darrell, T. & Gonzalez, J. (2021b). TTT++: When Does Self-Supervised Test-Time Training Fail or Thrive? *Advances in Neural Information Processing Systems (NeurIPS)*.

Liu, W., Sun, J., Li, W., Hu, T. & Wang, P. (2019a). Deep learning on point clouds and its application: A survey. *Sensors*, 19(19), 4188.

Liu, Y., Chen, C., Wang, C., King, X. & Liu, M. (2023). Regress before construct: Regress autoencoder for point cloud self-supervised learning. *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 1738–1749.

Liu, Y., Fan, B., Xiang, S. & Pan, C. (2019b). Relation-shape convolutional neural network for point cloud analysis. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8895–8904.

Liu, Y., Cheng, A., Chen, H. & Gu, A. (2024). VMamba: Visual State Space Models. *arXiv preprint arXiv:2401.01800*.

Loshchilov, I. & Hutter, F. (2016). SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Loshchilov, I. & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ma, X., Qin, C., You, H., Ran, H. & Fu, Y. (2022a). Rethinking network design and local geometry in point cloud: A simple residual MLP framework. *arXiv preprint arXiv:2202.07123*.

Ma, X., Zhou, C., Kong, X., He, J., Gui, L., Neubig, G., May, J. & Zettlemoyer, L. (2022b). Mega: moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*.

Mehta, H., Gupta, A., Cutkosky, A. & Neyshabur, B. (2022). Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*.

Mirza, M. U., Choi, C., Choi, S. & Kim, J. (2023). MATE: Masked Autoencoders are Online 3D Test-Time Learners. *Advances in Neural Information Processing Systems (NeurIPS)*.

Mirza, M., Li, L. & Jabri, A. (2022). Norm-Preserved Test-Time Training. *European Conference on Computer Vision*.

Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B. & Snoek, J. (2020). Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*.

Ng, A., Jordan, M. & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Ogniewicz, R. L. & Ilg, M. (1992). Voronoi skeletons: theory and applications. *CVPR*, 92, 63–69.

Osowiechi, P., Rasouli, A., Creighton, O. & Lakshminarasimhan, V. (2024). TTTFlow: Unsupervised Normalizing Flows for Test-Time Training. *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.

Pang, Y., Wang, W., Bai, S. & Guibas, L. (2022). Masked Autoencoders for Point Cloud Self-Supervised Learning. *European Conference on Computer Vision (ECCV)*.

Phan, A. V., Le Nguyen, M., Nguyen, Y. L. H. & Bui, L. T. (2018). Dgcnn: A convolutional neural network over large-scale labeled graphs. *Neural Networks*, 108, 533–543.

Pioro, R., Dao, T. & Gu, A. (2024). MoE-Mamba: Mixture-of-Experts Mamba for Efficient Long-Sequence Modeling. *arXiv preprint arXiv:2402.11662*.

Qi, C. R., Su, H., Mo, K. & Guibas, L. (2017a). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, pp. 652–660.

Qi, C. R., Yi, L., Su, H. & Guibas, L. J. (2017b). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *NeurIPS*.

Qi, C. R., Zhou, Y., Najibi, M., Sun, P., Vo, K., Deng, B. & Anguelov, D. (2021). Offboard 3d object detection from point cloud sequences. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6134–6144.

Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M. & Ghanem, B. (2022). Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35, 23192–23204.

Qiu, S., Anwar, S. & Barnes, N. (2021). Geometric back-projection network for point cloud classification. *IEEE Transactions on Multimedia*, 24, 1943–1955.

Ran, H., Liu, J. & Wang, C. (2022). Surface representation for point clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18942–18952.

Ren, J., Pan, L. & Liu, Z. (2022a). Benchmarking and analyzing point cloud classification under corruptions. *International Conference on Machine Learning*, pp. 18559–18575.

Ren, J., Pan, L. & Liu, Z. (2022b). Benchmarking and analyzing point cloud classification under corruptions. *International Conference on Machine Learning*, pp. 18559–18575.

Reuter, M., Wolter, F.-E. & Peinecke, N. (2005). Laplace-spectra as fingerprints for shape matching. *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pp. 101–106.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695.

Rusu, R. B. & Cousins, S. (2011). 3D is Here: Point Cloud Library (PCL). *IEEE International Conference on Robotics and Automation (ICRA)*.

Sauder, J. & Sievers, B. (2019). Self-supervised deep learning on point clouds by reconstructing space. *Advances in Neural Information Processing Systems*, 32.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W. & Bethge, M. (2020). Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33, 11539–11551.

Sharma, C. & Kaul, M. (2020). Self-supervised few-shot learning on point clouds. *Advances in Neural Information Processing Systems*, 33, 7212–7221.

Shi, J. & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8), 888–905.

Shi, S., Wang, X. & Li, H. (2019). Pointrcnn: 3d object proposal generation and detection from point cloud. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 770–779.

Shim, H., Kim, C. & Yang, E. (2024). CloudFixer: Test-Time Adaptation for 3D Point Clouds via Diffusion-Guided Geometric Transformation. *European Conference on Computer Vision*, pp. 454–471.

Shin, W., Choi, S., Kim, J. & Choi, Y. (2022). MM-TTA: Test-Time Adaptation for Multi-Modal 3D Semantic Segmentation. *European Conference on Computer Vision (ECCV)*.

Slim, H., Li, X., Li, Y., Ahmed, M., Ayman, M., Upadhyay, U., Abdelreheem, A., Prajapati, A., Pothigara, S., Wonka, P. & Elhoseiny, M. (2023). 3DCoMPaT++: An improved Large-scale 3D Vision Dataset for Compositional Recognition.

Su, H., Maji, S., Kalogerakis, E. & Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3D shape recognition. *Proceedings of the IEEE international conference on computer vision*, pp. 945–953.

Sun, F., Choi, Y.-K., Yu, Y. & Wang, W. (2015). Medial meshes–a compact and accurate representation of medial axis transform. *IEEE transactions on visualization and computer graphics*, 22(3), 1278–1290.

Sun, J., Zhang, Q., Kailkhura, B., Yu, Z., Xiao, C. & Mao, Z. M. (2022). Benchmarking robustness of 3D point cloud recognition against common corruptions. *arXiv preprint arXiv:2201.12296*.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. & Hardt, M. (2020). Test-time training with self-supervision for generalization under distribution shifts. *International conference on machine learning*, pp. 9229–9248.

Tang, Y., Li, X., Xu, J., Yu, Q., Hu, L., Hao, Y. & Chen, M. (2023). Point-LGMask: Local and Global Contexts Embedding for Point Cloud Pre-training with Multi-Ratio Masking. *IEEE Transactions on Multimedia*.

Tay, Y., Bahri, D., Yang, L., Metzler, D. & Juan, D.-C. (2020). Sparse sinkhorn attention. *International Conference on Machine Learning*, pp. 9438–9447.

Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, D. T. & Yeung, S.-K. (2019a). Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. *International Conference on Computer Vision (ICCV)*.

Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, T. & Yeung, S.-K. (2019b). Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1588–1597.

Valsesia, D., Fracastoro, G. & Magli, E. (2020). Learning localized representations of point clouds with graph-convolutional generative adversarial networks. *IEEE Transactions on Multimedia*, 23, 402–414.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł. & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems (NeurIPS)*.

Wang, C., Xu, H., Chen, K., Wang, W., Qin, C., Li, X., Gao, Y., Li, Y. & Li, S. (2024a). Continual Test-Time Domain Adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B. A. & Darrell, T. (2021a). Tent: Fully Test-Time Adaptation by Entropy Minimization. *International Conference on Learning Representations (ICLR)*.

Wang, D., Sun, J., Tang, J. & Zhang, D. (2024b). Backpropagation-Free Test-Time Adaptation for 3D Point Clouds. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, H., Liu, Q., Yue, X., Lasenby, J. & Kusner, M. J. (2021b). Unsupervised point cloud pre-training via occlusion completion. *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9782–9792.

Wang, P.-S. (2023). Octformer: Octree-based transformers for 3d point clouds. *ACM Transactions on Graphics (TOG)*, 42(4), 1–11.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M. & Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5), 1–12.

Wang, Y., Qian, Y. et al. (2023). Hard-MAE: Hard Masking for Masked Point Modeling. *ICCV*.

Wei, X., Yu, R. & Sun, J. (2020). View-GCN: View-based graph convolutional network for 3D shape analysis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1850–1859.

Wen, C., Yu, B. & Tao, D. (2023). Learnable skeleton-aware 3d point cloud sampling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17671–17681.

Wu, C., Zheng, J., Pfrommer, J. & Beyerer, J. (2023a). Attention-based point cloud edge sampling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5333–5343.

Wu, J., Zhang, C., Xue, T., Freeman, B. & Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29.

Wu, W., Qi, Q. & Fuxin, L. (2019). PointConv: Deep Convolutional Networks on 3D Point Clouds. *CVPR*.

Wu, X., Lao, Y., Jiang, L., Liu, X. & Zhao, H. (2022). Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35, 33330–33342.

Wu, X., Jiang, L., Wang, P.-S., Liu, Z., Liu, X., Qiao, Y., Ouyang, W., He, T. & Zhao, H. (2023b). Point transformer v3: Simpler, faster, stronger. *arXiv preprint arXiv:2312.10035*.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920.

Xiao, C., Wang, Y., Liu, Y. et al. (2023). Unsupervised Learning of Geometry-Aware 3D Representations. *International Conference on Computer Vision (ICCV)*.

Xu, Y., Fan, T., Xu, M., Zeng, L. & Qiao, Y. (2018). SpiderCNN: Deep learning on point sets with parameterized convolutional filters. *Proceedings of the European conference on computer vision (ECCV)*, pp. 87–102.

Yan, S., Yang, Z., Li, H., Song, C., Guan, L., Kang, H., Hua, G. & Huang, Q. (2023). Implicit Autoencoder for Point-Cloud Self-Supervised Representation Learning. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14530–14542.

Yan, Y., Letscher, D. & Ju, T. (2018). Voxel cores: Efficient, robust, and provably good approximation of 3d medial axes. *ACM Transactions on Graphics (TOG)*, 37(4), 1–13.

Yang, Y., Feng, C., Shen, Y. & Tian, D. (2018). FoldingNet: Point cloud auto-encoder via deep grid deformation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 206–215.

Yeo, T., Kar, O. F., Sodagar, Z. & Zamir, A. (2023). Rapid network adaptation: Learning to adapt neural networks using test-time feedback. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4674–4687.

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A. & Guibas, L. (2016). A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (ToG)*, 35(6), 1–12.

Yu, L., Shen, J., Qu, T. et al. (2022). Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling. *CVPR*.

Zbontar, J., Jing, L., Misra, I., LeCun, Y. & Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. *International conference on machine learning*, pp. 12310–12320.

Zha, Y., Ji, H., Li, J., Li, R., Dai, T., Chen, B., Wang, Z. & Xia, S.-T. (2024). Towards compact 3d representations via point feature enhancement masked autoencoders. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(7), 6962–6970.

Zhang, M., Menon, A., Jiang, H., Andreassen, A. & Kumar, N. (2022a). MEMO: Test Time Robustness via Adaptation and Augmentation. *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang, R., Zhang, X., Jin, X. et al. (2022b). Point-M2AE: Multi-Scale Masked Autoencoders for 3D Point Cloud Understanding. *NeurIPS*.

Zhang, T., Yuan, H., Qi, L., Zhang, J., Zhou, Q., Ji, S., Yan, S. & Li, X. (2025). Point cloud mamba: Point cloud learning via state space model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(10), 10121–10130.

Zhang, Y., Li, T., Zhou, H., Liu, W. & Zhao, H. (2024). Lookaround: Improving Generalization via Diversified Weight Averaging. *arXiv preprint arXiv:2402.16802*.

Zhang, Y. et al. (2023). Learning 3D Representations from 2D Pre-trained Models via Image-to-Point Masked Autoencoders. *NeurIPS*.

Zhao, H., Jiang, L., Fu, C.-W. & Jia, J. (2019). Pointweb: Enhancing local neighborhood features for point cloud processing. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5565–5573.

Zhao, H., Jiang, L., Fu, C.-W., Jia, J. et al. (2021). Point Transformer. *International Conference on Computer Vision (ICCV)*.

Zhu, W., Gao, Z., Shen, S. & Li, H. (2024). Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Models. *arXiv preprint arXiv:2403.13033*.