Self-Supervised Representation Learning for CAD Models

Zur Erlangung des akademischen Grades einer

Doktorin der Ingenieurwissenschaften

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

genehmigte
DISSERTATION

von

Vencia Desirée Elisabeth Herzog

aus Aachen

Tag der mündlichen Prüfung:	4. Februar 2025
Erster Gutachter:	Prof. Dr.-Ing. Tamim Asfour
Zweiter Gutachter:	Prof. Dr. Justus Piater

Deutsche Zusammenfassung

Designentscheidungen und Verarbeitungsschritte in der Produktentwicklung sind eng mit der geometrischen Form des Produktes verknüpft. Eine umfassende Unterstützung im Produktentwicklungsprozess, etwa durch Assistenz bei Konstruktionsarbeiten, Bewertung von Produktkosten und -Fertigbarkeit oder Auffinden von Gleichteilen, erfordert ein grundlegendes Geometrieverständnis. Maschinelle Lernverfahren ermöglichen über ihre datengetriebene Funktionsweise die Modellierung dieser komplexen geometrischen und semantischen Zusammenhänge. Allerdings stellt die begrenzte Verfügbarkeit von annotierten Trainingsdaten in der Praxis eine wesentliche Herausforderung dar.

Im Bereich der Sprach- und Bilderkennung werden zunehmend Methoden des *self-supervised representation learning* eingesetzt. Diese Methoden erlauben es kompakte, vielseitig einsetzbare Repräsentationen aus einem breiten Spektrum ungelabelter Daten zu generieren. Diese Repräsentationen kodieren grundlegende Strukturen und Merkmale der Eingabedaten und können zur dateneffizienten Lösung von Anwendungsaufgaben eingesetzt werden.

Diese Dissertation beschäftigt sich mit der Entwicklung von Verfahren zum selbstüberwachten Erlernen von 3D Repräsentationen für CAD-Modelle, mit Anwendungsfokus in der Produktentwicklung. Ein Kernproblem hierbei ist die Formulierung selbstüberwachter Aufgaben, die das Erlernen einer aussagekräftigen Repräsentation fördern. Während sich bestehende Forschungsarbeiten primär mit reinen Punktwolkendaten beschäftigen, liegt der Fokus dieser Dissertation speziell auf der Verwendung von Computer-aided Design (CAD) Daten.

Die Hauptbeiträge der Arbeit lassen sich in die folgenden drei Bereiche einteilen:

Dateneffizientes Lernen auf CAD Daten

In der Produktentwicklung stehen zahlreiche Datenquellen mit CAD-Daten zur Verfügung. Häufig mangelt es jedoch an ausreichend problemspezifischen, annotierten Trainingsdaten, um tiefe neuronale Netze erfolgreich auf geometrischen Daten zu trainieren. Der erste Teil dieser Arbeit beschäftigt sich mit der Untersuchung von Wissenstransferstrategien, die darauf abzielen, die Dateneffizienz von Lernaufgaben in der Produktentwicklung zu steigern. Dies wird durch die effiziente Nutzung verwandter und ungelabelter Daten erreicht. Ein Schwerpunkt der Untersuchungen liegt auf dem Vergleich von überwachten und selbstüberwachten *transfer learning* Methoden zur Verbesserung der Klassifikation mechanischer Bauteile. Die experimentelle Studie analysiert sowohl den Einfluss manueller Annotationen auf die Effektivität des Vortrainings, als auch die Auswirkung des *domain gap* – der Diskrepanz zwischen Vortraining- und Anwendungsdatendomänen. Die experimentellen Ergebnisse zeigen die Effektivität selbstüberwachter Lernmethoden für das Vortraining von CAD-Modellen, insbesondere mit einer starken Generalisierungsfähigkeit bei großem domain gap.

Selbstüberwachte Aufgabenformulierung für CAD Modelle

CAD-Daten enthalten vielschichtige Informationen und ausgeprägte konstruktionsbedingte Merkmale. Die Nutzung dieser CAD-spezifischen Informationen und Eigenschaften ermöglicht die Formulierung neuartiger selbstüberwachter Aufgaben, die es erlauben, mit reinen Punktwolken unerfasste geometrische und semantische Zusammenhänge abzubilden. Diese Arbeit bietet einen systematischen Überblick über verschiedene Arten nutzbarer CAD-Informationen, unterteilt in (1) Informationen aus nativen CAD-Formaten, (2) Konstruktionsmerkmale, und (3) multimodale Informationen, die aus CAD-Daten extrahiert werden können. Ein wesentlicher Beitrag ist die Entwicklung einer multimodalen, selbstüberwachten Lernmethode, die Punktwolken- und Bilddaten kombiniert, um ein umfassendes Objektverständnis zu erzielen. Dabei werden die Punktwolken- und Bildmodalitäten in einen gemeinsamen Merkmalsraum projiziert und Korrespondenzen auf lokaler und globaler Objektebene herge-

stellt. Das entwickelte *cross-modal pretraining* Verfahren übertrifft state-of-the-art Vortrainingsmethoden in mehreren 3D Benchmark-Datensätzen.

Praxisorientierte Anwendung in der Bauteilsuche

Eingelernte Repräsentationen von CAD-Modellen eignen sich auch für den Einsatz als vektorbasierte Objektsignaturen. Sie stellen strukturierte, komprimierte Kodierungen der Eingabedaten dar, die eine effiziente Berechnung eines Ähnlichkeitsmaßes zwischen den CAD-Modellen ermöglichen. Eine praxisorientierte Anwendung dieser eingelernten Objektsignaturen ist die Bauteilsuche, bei der ähnliche Teilkomponenten in historischen Konstruktionsdaten identifiziert werden, um sie in neuen Projekten wiederzuverwenden. In diesem Zusammenhang wurde ein interaktives Suchsystem entwickelt, das es Konstruktionsingenieuren ermöglicht, gewichtete *Regions Of Interest* auf hierarchischen, vektorbasierten Objektsignaturen zu definieren, um eine subjektive, funktionsorientierte Ähnlichkeitsbewertung vornehmen zu können.

Contents

1 Introduction **1**
1.1 The Role of Geometric Intelligence in Product Engineering 1
1.2 Problem Statement 4
1.3 Contributions 5
1.4 Structure of the Thesis 7

2 Foundations & Related Work **9**
2.1 Representation of CAD Models 9
2.1.1 Solid Modeling 10
2.1.2 Surface Meshes 12
2.1.3 Shape Signatures 14
2.1.4 Summary 18
2.2 Geometric Deep Learning 19
2.2.1 Representing CAD Models for Deep Neural Networks . . 20
2.2.2 Deep Learning on Point Clouds 25
2.2.3 Summary 30
2.3 Self-Supervised Learning on Point Clouds 31
2.3.1 Generative Learning 32
2.3.2 Contrastive Learning 34
2.3.3 Multimodal Learning 37
2.3.4 Summary 39
2.4 Discussion 40

3 Self-Supervised Representation Learning on 3D Geometry 41
3.1 An Introduction to Self-Supervised Representation Learning . . . 42
3.1.1 Supervised, Unsupervised, and Self-Supervised Learning 42
3.1.2 The Representation Learning Framework 43
3.2 Data-Efficient Learning on CAD Data 45
3.2.1 Data Acquisition and Usage in Product Engineering 45
3.2.2 Transfer Learning Strategies 48
3.3 Experiments 51
3.3.1 Experimental Setup 51
3.3.2 Transfer Learning Results 52
3.4 Summary 57

4 Self-Supervised Tasks for CAD Models 59
4.1 Leveraging B-Rep Information 60
4.2 Leveraging CAD Design Traits 63
4.2.1 90° Rotation Invariance Learning 63
4.3 Multimodal Learning 66
4.3.1 Approaches to 2D–3D Cross-Modal Pretraining 66
4.3.2 Structural Point Cloud–Picture Correspondence 68
4.4 Experiments 72
4.4.1 90° Rotation Invariance Learning 72
4.4.2 Cross-Modal Learning 75
4.5 Summary 82

5 Content-based Shape Retrieval for Design Reuse 85
5.1 Partial Shape Retrieval with User Guidance 86
5.1.1 Partial Shape Retrieval Methods 86
5.1.2 User Guidance and Feedback Mechanisms 88
5.2 Hierarchical Shape Signatures 89
5.2.1 Global Shape Descriptor 89
5.2.2 Local Shape Descriptor 90
5.3 Index-based Similarity Search 91
5.3.1 Indexing Structures for k-NN Search 91
5.3.2 Search Process 92
5.4 Experiments 94
5.4.1 Experimental Setup 94
5.4.2 Results on ESB 95
5.4.3 Results on DMUNet 99
5.4.4 Results on ABC Dataset with Screws 102

5.4.5 Discussion 104
5.5 Summary 105

6 Conclusion **107**
6.1 Scientific Contributions 107
6.2 Industrial Applications 109
6.3 Outlook 111

List of Symbols **117**

List of Figures **120**

List of Tables **121**

List of Algorithms **123**

Bibliography **125**

List of Publications **147**

CHAPTER 1

Introduction

Electromechanical product engineering is an iterative process that requires the coordinated efforts of various teams, from designers and numerical analysts to test engineers and manufacturing experts. Central to this process is the geometric shape of the product, typically represented by Computer-Aided Design (CAD) models. The geometric information contained in these models has a direct impact on design decisions, analysis results, and manufacturing considerations.

With advancing technologies and increased global competition, the demand for shorter lead times, higher quality, and greater cost efficiency continues to grow. The ability to derive meaningful insights from a product's geometry, referred to as *geometric intelligence* in this context, is becoming increasingly relevant for making informed decisions throughout the product life cycle.

1.1 The Role of Geometric Intelligence in Product Engineering

With the digitization of the industry, computer-aided design has become an indispensable tool for product development. It provides digital tools to create precise three-dimensional representations of physical artifacts, known as CAD models. CAD models are the primary source of information communicated

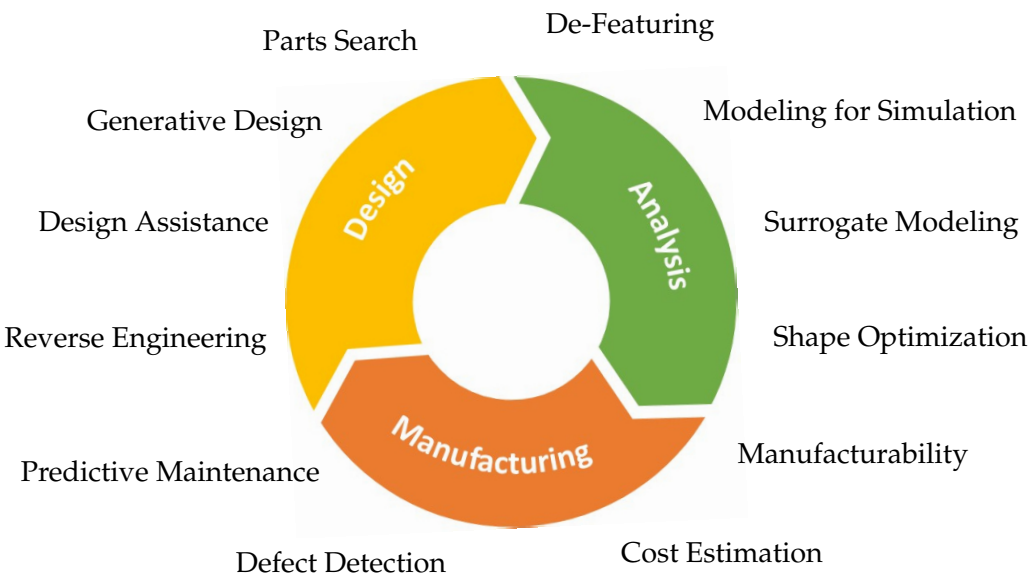


Figure 1.1: Examples of AI applications in product engineering.

throughout the product development process. The digital models are used to visualize the design concept, verify design feasibility, analyze product behavior, and configure manufacturing procedures.

The prospects for integrating Artificial Intelligence (AI) into model-based product engineering have been recognized since the late 1980s (Wright and Bourne, 1988; Carter, 1990). As computers became more prevalent in the industry, AI was seen as a logical next step to enhance system intelligence. By extracting knowledge from various sources throughout the product life cycle, it can provide valuable insights for design and development. It empowers designers and engineers to make informed business decisions and enables the automation of repetitive tasks, reducing time to market. Figure 1.1 gives examples of AI applications at various stages of product engineering, as outlined below.

AI in Design and Modeling The premise of intelligent *design assistance* is to provide comprehensive support for human design activities. This can range from automating basic modeling steps, such as identifying and modifying similar components like fillets, chamfers, and bolts, to providing guidance at a conceptual level. In *generative design*, AI facilitates the automatic generation of diverse designs defined by user criteria and constraints regarding performance, materials, and spatial requirements. This approach allows for rapid exploration and iteration within the design space. Integrated or standalone *parts search* capabilities reduce construction time and improve standardization by enabling efficient reuse of in-house or purchased parts. Lastly, in cases where the CAD

model for a finished product is unavailable or no longer aligns with the original geometry, the digital model can be reconstructed from measured data through a process known as *reverse engineering*.

AI in Simulation and Analysis Prior to simulation, CAD models undergo time-consuming preprocessing that is tailored to the type of simulation and solver being used. Engineers conduct a *de-featuring* of the CAD geometry to simplify the meshing process and reduce simulation run time. Other *modeling for simulation* includes, for example, marking potential collision surfaces in a crash simulation or detecting and replacing plastic connectors in a car chassis. Automating these routine tasks or providing prefiltered options can significantly cut manual labor in simulation preprocessing, allowing engineers to focus on tasks that require their expertise. A second strand of AI applications is concerned with substituting numerical simulations with *surrogate models*. A surrogate model learns from past simulations to approximate target variables such as stress or drag coefficient for variations of the given 3D design. Surrogate models provide fast run times and meshless geometry preparation. They facilitate interactive design exploration and allow *shape optimization* processes, such as aerodynamic airfoil optimization, to be encapsulated in a single, easy-to-use tool at design time.

AI in Manufacturing Decisions made in the early stages of product design have a significant impact on subsequent production costs. Early consideration of *manufacturability* and *cost estimation* can prevent costly mistakes down the line. These considerations are linked to the product's geometry and the associated machining features (e.g., holes, slots, grooves). In addition to integrating manufacturing information into the design phase, the wealth of sensor data collected during manufacturing can be used for real-time in-situ quality assessment. *Defect detection* methods use RGB(D) sensor data to detect surface or internal defects such as cracks and pores in manufactured parts. The imaging data can also be correlated with the original CAD model to identify any deviations from the 3D design, such as variations in 3D printed parts due to a combination of material and printing parameters. *Predictive maintenance* monitors the condition of in-service equipment to estimate maintenance schedules and predict time to failure.

A major obstacle for AI applications in product engineering is data scarcity. The lack of problem-specific, expertly labeled datasets limits the broader adoption

of machine learning methods in this area. In Natural Language Processing (NLP) and computer vision, *self-supervised representation learning* techniques are proving effective in building general knowledge from diverse, unlabeled datasets, which helps solve subsequent tasks with less training data. Similarly, fostering a fundamental geometric understanding from CAD models has the potential to improve data efficiency in subsequent engineering tasks.

Beyond improving data efficiency, integrating knowledge from the complex systems and tools used in product development has the potential to create a more complete understanding of the designed product. A growing trend in product development is to reduce iterations by providing a holistic view of the product's performance, manufacturability, durability, and cost early in the design phase. Research shows that early design decisions determine up to 80% of total product cost, while accounting for only a small fraction of that cost (Ullman, 2003). Achieving this holistic view requires modeling and encapsulating complex engineering processes into accessible tools at design time.

1.2 Problem Statement

This thesis is concerned with building fundamental geometric understanding from CAD models to improve application tasks in product engineering. The main objective is to create versatile, intermediate feature representations from a wide range of unlabeled CAD data that capture essential geometric and contextual relationships relevant to digital 3D designs. This is pursued through self-supervised representation learning, whereby Deep Neural Networks (DNNs) are trained to extract high-level, reusable features from raw shape inputs.

To address the practical limitations of manual annotation at a large scale, self-supervised learning techniques derive labels directly from the input data itself. A central aspect of this thesis is the design of self-supervised pretraining tasks, referred to as pretext tasks, that promote a high informative value of the learned representations. While existing studies mainly focus on raw point clouds, this thesis specifically targets the development of pretext tasks for CAD models, addressing key research questions on how to effectively exploit the distinctive information and characteristics of CAD models in the design of pretext tasks.

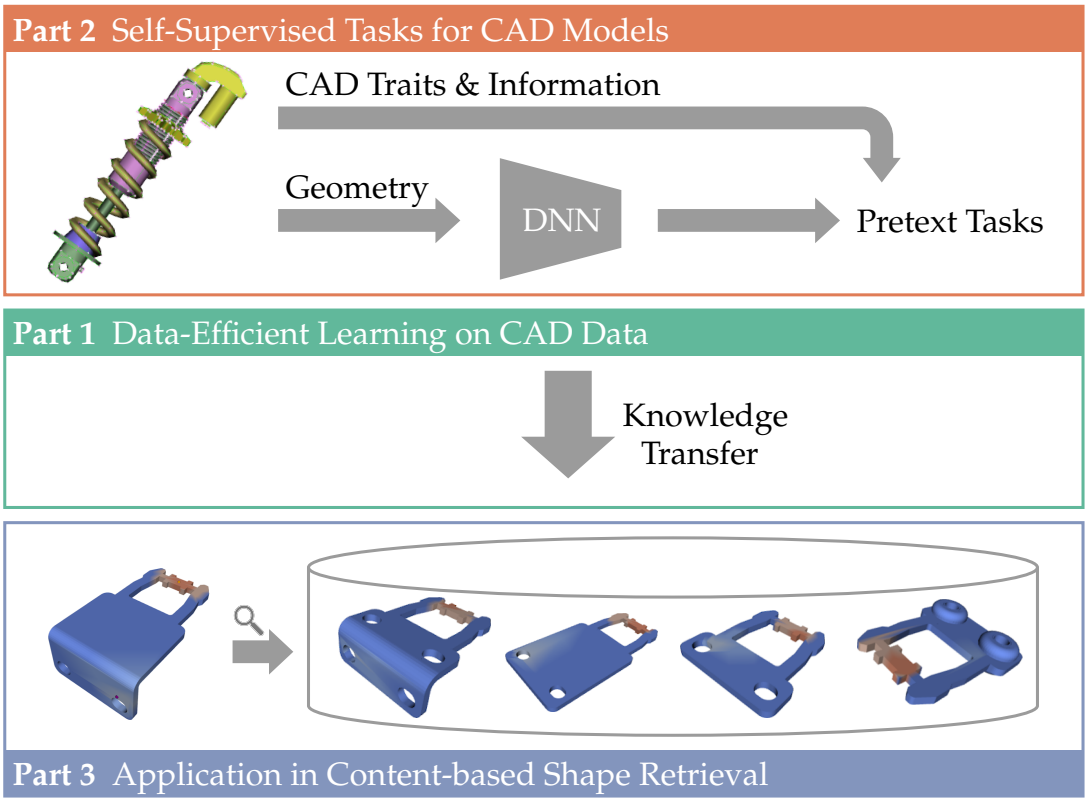


Figure 1.2: Contributions of the thesis divided into three parts.

1.3 Contributions

Figure 1.2 illustrates the thesis structure, which organizes the contributions into three parts. The following sections outline the main contributions of the thesis.

Data-Efficient Learning on CAD Data

In the first part of this thesis, we analyze the effectiveness of self-supervised representation learning for improving the performance of application tasks in mechanical engineering. We begin by examining the current data landscape in product engineering and introduce a data acquisition and usage concept that leverages knowledge transfer to mitigate data scarcity. We evaluate both supervised and self-supervised transfer learning strategies to improve data and label efficiency for mechanical component classification. The experimental study centers on two main aspects. First, it compares the performance of self-supervised pretraining, which does not require manually labeled data, with supervised pretraining. Second, it analyzes the effectiveness of knowledge

transfer across domain gaps, focusing on how the similarity between source and downstream datasets affects the performance of pretraining. This chapter is based on Herzog and Suwelack (2022).

Self-Supervised Tasks for CAD Models

We extend the research field of self-supervised 3D representation learning by developing pretext tasks specifically tailored to CAD models. We provide a systematic overview of strategies for exploiting native CAD format information, design features, and multimodal data in self-supervised representation learning, and address challenges related to the availability and heterogeneity of such data. We demonstrate how the axis alignment of CAD models can be used to formulate a pretext task that enforces 90° rotation invariance to effectively reduce the complexity of the learned representations. A major contribution of this chapter is the development of a novel cross-modal pretraining method that combines point cloud and image features extracted from CAD models to improve object understanding. This method captures rich contextual cues by predicting correspondences between point cloud and image features at both local and global scales. This chapter is based on Herzog and Suwelack (2025).

Application in Content-based Shape Retrieval

In the final part of the thesis, we explore the practical application of self-supervised representation learning in shape retrieval for design reuse. We present a partial shape retrieval system that allows users to interactively define weighted Regions of Interest (ROIs) to assess design similarity based on user intent and application context. This system addresses a major barrier of current content-based retrieval solutions in bridging the semantic gap between geometric information and design intent. We develop hierarchical, vector-based shape signatures by combining self-supervised learning-based representations that capture nuanced semantic shape relationships at the global level with traditional shape descriptors that capture fine details without requiring training on local regions. We achieve interactive partial shape matching at multiple scales through efficient index-based similarity search functionalities. This chapter is based on Herzog and Suwelack (2023).

1.4 Structure of the Thesis

The thesis is structured as follows:

Chapter 2 provides the relevant background and related work for this thesis. We begin with an introduction to solid modeling and CAD model representations. We then provide an overview of geometric deep learning, highlighting the advantages and limitations of different input representations for deep learning, and discuss architectural considerations for deep learning on point clouds. The chapter concludes with a review of relevant literature on self-supervised learning for point clouds, and positions this work within the broader research landscape.

Chapter 3 begins with an introduction to the self-supervised representation learning framework. The chapter continues with a study of supervised and self-supervised transfer learning to improve the data efficiency of downstream tasks in mechanical engineering.

Chapter 4 focuses on strategies for leveraging CAD-specific information and traits to design pretext tasks on point clouds. In particular, this chapter introduces a novel cross-modal pretraining method that combines point cloud and multi-view data extracted from CAD models to learn rich, discriminative representations.

Chapter 5 demonstrates the application of learned feature representations in content-based shape retrieval for design reuse. The chapter introduces partial shape retrieval and user feedback mechanisms, and discusses the construction of shape signatures. Finally, we present the index-based search structure and interactive search process, and provide a comprehensive evaluation of the developed retrieval system.

Chapter 6 concludes the thesis by discussing the scientific contributions and providing insights into industrial applications. Finally, we draw connections to related research areas and outline directions for future work.

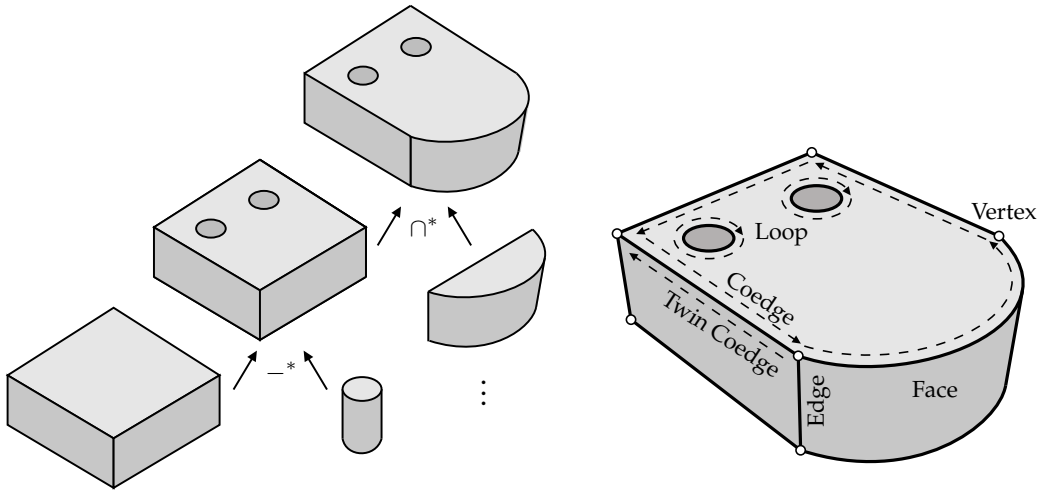
CHAPTER 2

Foundations & Related Work

This chapter presents the foundational concepts and relevant literature for this thesis. Section 2.1 begins with an introduction to CAD modeling, providing an overview of CAD representation schemes and shape descriptors. In Section 2.2, we dive into deep learning techniques on 3D geometry, starting with a discussion of different input representations derived from CAD models for machine learning. This is followed by an overview of neural network architectures for point clouds. Section 2.3 introduces key terminology in self-supervised learning and reviews methods relevant to this work. Finally, Section 2.4 summarizes this chapter.

2.1 Representation of CAD Models

Modern CAD systems take a holistic approach to modeling a physical product and its production process. The core of the product model is the description of the object's 3D shape. Traditionally, this is modeled using solid modeling techniques, where the interior of an object is defined by its bounding surface or by a logical combination of geometric primitives. Shape modeling information such as materials, tolerances, kinematic constraints, and assembly structure can optionally be associated with the shape model, along with production-related information such as analysis results and process plans.



(a) Constructive Solid Geometry (b) Boundary Representation

Figure 2.1: Solid representation schemes.

2.1.1 Solid Modeling

Solid modeling defines the principles of modeling the 3D shape of a real physical object by computer. As a subfield of geometric modeling, solid modeling emerged from the need for *informational completeness* and *physical fidelity* in engineering systems (Shapiro, 2002). A solid model completely represents the bounded volume of a physical object. Solid modeling encompasses various representation schemes that naturally and efficiently facilitate different geometric queries. The two main representation schemes used by solid modelers are Constructive Solid Geometry (CSG) and Boundary Representation (B-Rep).

Constructive Solid Geometry

First introduced by Voelcker and Requicha (1977), CSG defines a solid by a set-theoretic Boolean expression of closed regular primitives positioned in space by rigid-body transformations. Traditionally, these solid primitives include cuboid, sphere, cylinder, cone, and torus. Figure 2.1a shows an example of a CSG expression tree. To guarantee the solidity of constructed shapes, set operations are regularized. Regularized union, intersection and difference, denoted by $\cup^*$, $\cap^*$, and $-^*$, take the closure of the interior of the set operation result, eliminating any dangling lower-dimensional structures. A major advantage of CSG is the by definition guaranteed validity of the constructed shape under closed regular primitives and regularized set operations. However, CSG designs are limited by the choice of solid primitives. As a practical extension, CSG

is combined with other representation schemes such as B-Rep or feature-based modeling, which encompasses sweeping, offsetting, and blending operations, to fully describe objects of higher complexity.

Boundary Representation

In B-Rep, introduced independently by Baumgart (1974) and Braid (1975), a solid is implicitly defined by its oriented bounding surface. The topological entities – face, edge, and vertex – define the surface connectivity. They are associated with a geometric carrier – surface, space curve, and point – that describes the underlying location and shape. Faces and edges also include a boundary description that delimits the subarea or segment. Other topological entities typically include shells (a set of faces bounding a solid), loops (a circuit of coedges bounding a face), and coedges (a link that records the occurrence of an edge in a loop of a face). Figure 2.1b shows an example of a B-Rep shape description. B-Reps have a number of attractive properties. All solids can be represented accurately. The boundary of the solid is spatially addressable, making it suitable for applications that require computations such as rendering or boundary traversal. A drawback of B-Reps is the redundancy of geometric and incidence information, which must be kept consistent throughout the shape editing process. Errors and inconsistencies in the description can result in invalid (non-solid) shapes.

Solid Modeling Systems

Although manufactured parts typically have finite size and well-behaved boundaries, most solid modelers allow non-manifold solids, i.e., solids whose boundaries are not strictly 2-manifold. These non-manifold features are used, for example, to model shapes that meet at a common edge or vertex. Because solid modeling often blends multiple representation schemes, maintaining topological integrity is a non-trivial task that involves satisfying not only topological constraints, but also geometric constraints, such as avoiding self-intersections. Many modern solid modeling systems include freeform surface modeling capabilities such as Non-Uniform Rational B-Spline (NURBS) surfaces, which expand the range of achievable models to include thin plate objects, interior faces, dangling edges, and more. Solid modelers typically do not enforce topological integrity, leaving this responsibility to the user. This can

cause problems later on because processes such as Finite Element (FE) analysis require models to be watertight and manifold.

The widespread demand for solid modeling tools in various engineering environments and applications has led to the development of numerous commercial (e.g., Parasolid, ACIS) and open source (e.g., Open Cascade) modeling kernels. However, due to differences in representation schemes, tolerances, application-specific features, and file formats, system interoperability and lossless model exchange are not guaranteed. To improve compatibility, neutral exchange formats such as IGES and STEP have been developed. Despite these efforts, the seamless and error-free exchange of product model data remains unresolved. While solid models provide a comprehensive product description for design, simulation, and manufacturing purposes, alternative representations such as tessellated shape models have emerged to facilitate consistent and simplified handling of CAD models.

2.1.2 Surface Meshes

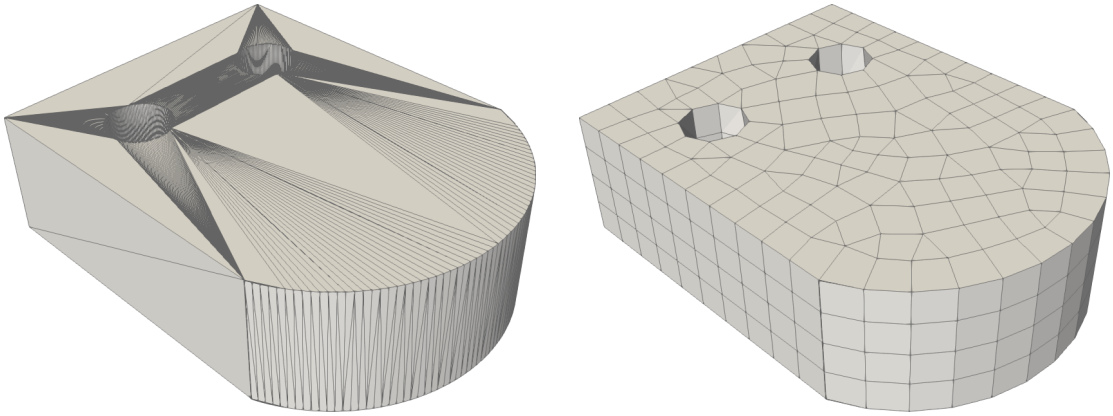
Meshes represent a body's surface using planar faces defined by simple polygons, such as triangles and quadrilaterals. Meshes are standardized and easier to manage than native CAD formats. They can also be captured using laser scanning or CT imaging methods, allowing comparison between digital and physical artifacts. However, converting a CAD model to mesh format results in a discretization of the domain and a partial loss of advanced shape modeling information, such as underlying geometric primitives, modeling operations, materials, and units.

Mesh generation algorithms are controlled by a combination of parameters related to the following criteria:

1. the precision of the geometric approximation,
2. the fineness of the mesh, and
3. the quality of the mesh elements.

STL Triangulation

STL triangulation is a standard functionality available in most CAD modeling tools. STL files describe the boundary of a solid by a list of triangular facets. They are widely used for prototyping, rendering, and manufacturing purposes.



(a) STL Triangulation (b) Advancing Front

Figure 2.2: Mesh generation methods.

The algorithms used for STL triangulation are highly efficient and are designed to minimize geometric approximation errors on the body surface regardless of the shape of the mesh elements. Figure 2.2a shows an example of a typical STL triangulation, where thin triangles appear on curved surfaces and a high vertex order can be observed at the outer corners.

The STL file format defines each triangular face by its face normal and vertex coordinates. Face connectivity is not explicitly stored. As a result, topological errors are common in STL meshes. While newer mesh formats such as PLY and OBJ have been developed to include explicit connectivity information, STL remains a widely used standard in engineering.

FE Meshes

Mesh generation plays an integral role in physical simulations such as FE analysis, where numerical solutions to partial differential equations are calculated based on the mesh elements. The shape and quality of these mesh elements have a major impact on the results of numerical methods. Mesh elements with high aspect ratios are typically avoided because they produce ill-conditioned matrices that adversely affect the accuracy, speed, and convergence of the solver.

Common mesh generation methods include Delaunay triangulation (Delaunay et al., 1934) and the Advancing Front method (George, 1971). Delaunay triangulation is based on the Delaunay criterion, which states that no vertex should lie within the circumsphere of any triangle of the mesh. By design, the Delaunay triangulation maximizes the smallest angle among all triangulations.

The Advancing Front method constructs mesh elements by iteratively filling the unmeshed region of the domain, starting from the boundary. Figure 2.2b shows an example of an Advancing Front mesh, characterized by isotropic mesh elements and regular connectivity.

Mesh generation for physical simulation is a well-studied field with extensive literature covering unstructured (Ho-Le, 1988; Owen, 1998; Shimada, 2006), structured, and block-structured methods (Armstrong et al., 2015). Meshing algorithms can suffer from robustness issues and long computation times. Mesh facets must satisfy strict quality criteria in terms of minimum angles and aspect ratios. At the same time, these algorithms must deal with irregularities and degeneracies present in the shape models and guarantee watertightness, as incomplete connectivity can severely compromise the validity of calculations performed on the meshes.

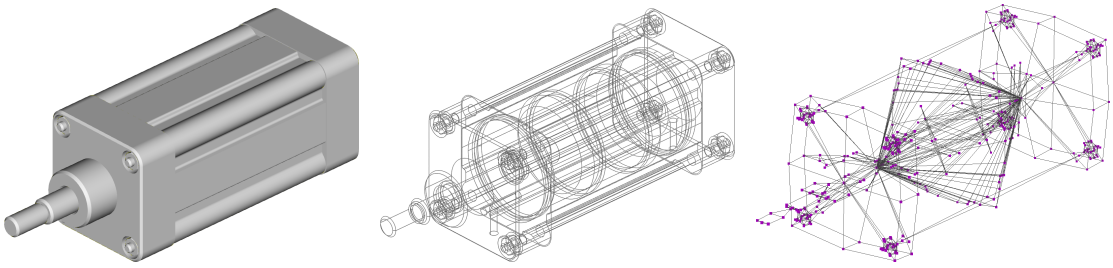
2.1.3 Shape Signatures

Although 3D shape representations such as B-Reps and meshes accurately and comprehensively describe a physical object, they do not provide an easy way to evaluate, compare, and classify such objects. Shape signatures, also referred to as shape descriptors, encode essential shape characteristics in a compact form, enabling efficient recognition and comparison.

Loncaric (1998) strictly differentiates between *shape representations*, which he defines as non-numeric depictions of a physical shape, and *shape descriptions*, which he views as numeric feature vectors. More recent literature tends to adopt a more flexible definition that blurs the boundaries between these two concepts. Kazmi et al. (2013) define shape descriptors as simplified shape representations in the form of vectors or graph-like structures that describe a shape geometrically or topologically. We adopt their definition in this work.

Techniques for Solid Models

Traditionally, CAD model signatures are graph-based representations of solid models based on associated design and manufacturing information. In the following, we introduce two common types of graph-based model signatures and discuss their strengths and limitations. For a comprehensive review of shape signatures for solid models in the context of shape search, we refer the reader to the survey by Iyer et al. (2005).



(a) Solid model surface (b) Solid model wireframe (c) Face adjacency graph

Figure 2.3: Illustration of an actuator solid model and its face adjacency graph. The graph is depicted embedded in $\mathbb{R}^3$ for better visibility.

Machining feature graphs describe features for manufacturing, such as holes, pockets, and fillets, and model the relationships between them. Elinson et al. (1997) present a shape signature based on various design attributes associated with machining features, such as the direction and radius of milling and drilling features and their directed dependencies. The nodes of the graph hold parameters of the design features, while the edges represent interactions that affect the manufacturing process, such as feature intersections that require precedence constraints for manufacturing. Cicirello and Regli (2001) use an automatic feature recognition system to identify a fixed set of machining features from a given CAD model. They then construct an undirected, acyclic graph that defines design feature types and their intersections.

Methods that rely on machining features require either model-associated manufacturing plans or the use of feature recognition systems to identify a set of machining feature types. More complex designs, such as those with free-form surfaces, are typically not considered. Measuring the similarity between two feature graphs involves computing the largest common subgraph, a problem known to be NP-complete. Feature graphs are not unique for a given solid because features can be constructed in different ways.

A second group of methods defines model signatures as face adjacency graphs extracted from solid models, as illustrated in Figure 2.3. McWherter et al. (2001) construct attributed face adjacency graphs from solid models. The node attributes define face properties like face area and surface type, while the edge attributes define the concavity or convexity of the edge curve, the type of curve function, and the bounded curve length. Graph partitioning techniques are used to recursively isolate highly connected component graphs, and models are compared by computing the distance between eigenvalue vectors. El-Mehalawi and Miller (2003) propose an approximate graph matching method based on integer programming to compare attributed face adjacency graphs.

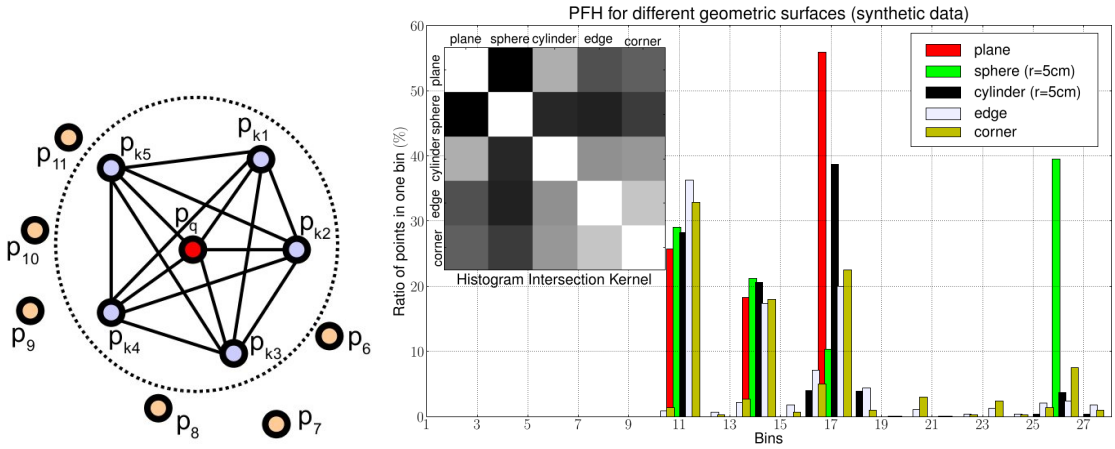
Methods that compute topological graphs from the B-Rep data structure work on the solid models themselves, without the need for additional design or manufacturing information. However, face adjacency graphs can become large even for simple shapes, making graph matching inefficient. Furthermore, techniques that rely on the connectivity of boundary models have proven to be sensitive to changes in topology. As noted by Ma et al. (2010), small features such as threading, ingrained text signatures, fillets, and chamfers can significantly alter model signatures.

Techniques for Shape Models

Shape-based techniques capture global or local features of an object's geometric shape. These descriptors, which are typically represented as vectors, can be computed from mesh or point cloud representations. Shape descriptors encompass a wide range of methods from computer vision, computational geometry, and computer graphics.

Global shape descriptor methods aim to characterize 3D models using global features or feature distributions. Examples of such features include geometric measurements, skeletal structures, visual features, moments, invariants, and frequency-based descriptors. Osada et al. (2002) propose a statistical descriptor that computes shape distributions based on several global properties such as distance, angle, area, and volume between pairs of random surface points. Hilaga et al. (2001) propose a graph-based descriptor based on multiresolution Reeb graphs for which topological matching is performed. Bespalov et al. (2003) and Biasotti et al. (2006) adapt Hilaga's method for matching mechanical CAD models. Chen et al. (2003) propose the *Light Field Descriptor (LFD)*, a descriptor based on visual features using silhouette images from uniformly distributed viewing angles encoded by Zernike moments and Fourier descriptors. Another view-based descriptor is *DESIRE*, proposed by Vranic (2005). This composite shape descriptor combines various view-based features of a 3D object, namely depth buffer images, silhouettes, and ray extents. Kazhdan et al. (2003) present the *Spherical Harmonics Descriptor (SHD)*, a rotation-invariant descriptor that represents a voxelized object through spherical harmonic functions by intersecting the voxel grid with concentric spheres and computing the frequency decomposition of each spherical function.

Local descriptors describe geometric and topological shape properties of a small neighborhood, referred to as the support region, around 3D key points. Because global methods compute a single description for the entire shape,



(a) Support region of point p_q (b) Example histograms for primitive surfaces

Figure 2.4: Illustrations of Point Feature Histograms presented in Rusu et al. (2009), © 2009 IEEE. *Left*: Computation of the support region for a red-colored query point p_q. *Right*: Comparison of histograms for different geometric surfaces.

they struggle to capture fine details and withstand clutter and occlusion. Local descriptors are typically accumulated through histograms or use a spatial ordering within a local reference frame. Johnson and Hebert (1999) introduced one of the earliest local descriptors based on spin images. Their method projects each point in the support region onto a plane that spins around the normal of the point to be described. Rusu et al. (2008) propose the use of a persistent *Point Feature Histogram (PFH)* to build point density and pose invariant shape descriptions, as depicted in Figure 2.4. A local reference frame is defined based on the point positions and surface normals for each pair of points in the support region. A multidimensional histogram is then computed over several angular features of all point pairs. In a later refinement, Rusu et al. (2009) introduced *FPFH*, which optimizes computation time by reducing the number of point pairs. Salti et al. (2014) propose the *SHOT* descriptor, a signature of histograms of orientations. It divides the support region into several subregions and concatenates the histograms of the point normal angles relative to the key point. The *Heat Kernel Signature (HKS)* developed by Sun et al. (2009) describes a shape based on the heat diffusion process over the surface. It is robust to isometric deformations, making it well suited for non-rigid shape matching.

Machine learning methods are changing the landscape of shape analysis and object recognition, shifting from knowledge-driven approaches like traditional shape signatures to data-driven shape representations (Rostami et al., 2019). Central to this paradigm shift are deep learning methods that automatically

extract descriptive and compact feature representations from large datasets through training.

2.1.4 Summary

CAD models are traditionally created using solid modeling techniques with the aim of achieving both informational completeness and physical fidelity (Shapiro, 2002). Solid models can incorporate a blend of various representation schemes, with CSG and B-Rep being the most prominent. CSG represents solids using Boolean expressions of regular primitives. B-Rep defines the boundary of solids using topological entities associated with geometric carriers that describe trimmed surfaces and curve segments. The high level of topological complexity and diversity, coupled with a lack of interoperability between CAD modeling kernels, presents challenges for unified operations on solid models.

To simplify and standardize solid models, shape boundaries can be approximated using meshes. Different types of meshes vary in the precision of the geometric approximation, the regularity of the connectivity, and the quality of the element shapes. The standard STL triangulation method is widely used to tessellate shape boundaries with precise geometric approximation and efficient generation. Conversely, FE meshes for physical simulation are highly regular in connectivity and element shape, but the meshing algorithms can be computationally expensive and prone to instability. Converting CAD models to meshes or other derived geometric representations typically implies losing associated shape modeling and product information.

Shape signatures are concise vector- or graph-based descriptions of 3D models that facilitate simple analysis and comparison of shapes. Traditionally, CAD model signatures are constructed from machining feature graphs based on associated manufacturing plans or from the connectivity information of boundary models. While graph-based methods effectively capture topological information, graph matching can be inefficient for topologically complex shapes. Alternatively, shape descriptors can be computed from shape models such as meshes or point clouds. Traditional shape signatures are limited in their capacity for semantic understanding and can be sensitive to noise and occlusion.

2.2 Geometric Deep Learning

Geometric data is ubiquitous in many fields, including computer vision, robotics, particle physics, chemistry, and medical imaging. Bronstein et al. (2017) coined the term geometric deep learning as:

> "an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains such as graphs and manifolds."

Geometric deep learning is mainly concerned with developing neural network architectures that overcome the high dimensionality of geometric domains by exploiting regularities present in physically structured data. The phenomenon known as the *curse of dimensionality* refers to the difficulties associated with learning in high-dimensional space. Approximating a target function f in high-dimensional space $\mathbb{R}^d$ generally requires a number of observations that grows exponentially with the dimension d. To mitigate this exponential growth, geometric deep learning methods exploit the spatial structure of physical systems.

According to Bronstein et al. (2021), all physically structured domains abide by the two fundamental principles of *symmetry* and *scale separation*, referred to as *geometric priors*. Symmetry imposes a structure on the function f that reduces the space of possible interpolants. Invariant functions ensure that the defining properties of an objects remain unchanged under certain transformations, such as translation (shift invariance). Equivariant functions make the output undergo the same transformation as applied to the input. Scale separation refers to a hierarchical structure imposed by the domain. It allows a function to be approximated as the composition of a coarse-scale function and a coarse-graining operator.

These principles are deeply rooted in the design of neural networks, especially Convolutional Neural Networks (CNNs). Convolutional filters with shared weights effectively exploit the shift equivariance of geometric domains. Their use of receptive fields at increasing scales exploits the principle of scale separation. While Euclidean-structured data has desirable properties such as a global parameterization and a common coordinate system that allow for natural application of CNNs, adapting these architectures to non-Euclidean domains requires a tailored approach for unstructured data. A major research focus is the development of convolutional architectures suitable for non-Euclidean domains such as graphs, manifolds, and point clouds.

2.2.1 Representing CAD Models for Deep Neural Networks

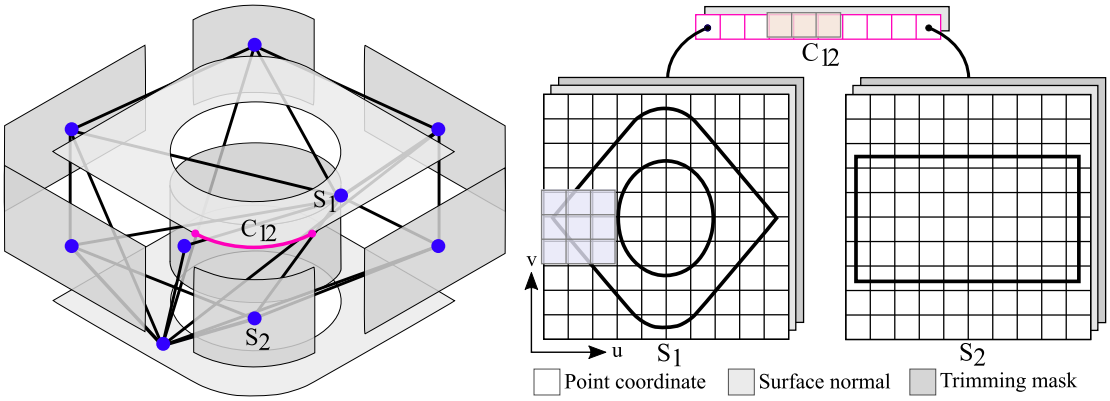
Learning feature representations for 3D CAD models requires careful consideration of the input data representation to choose for deep learning. Since CAD models provide a comprehensive symbolic description of 3D shapes, they can be converted into a variety of structured and unstructured formats. Table 2.1 provides an overview of input representations for the application of DNNs, outlining strengths and weaknesses and listing relevant research. The following sections provide short summaries of each input representation.

CAD (B-Rep)

Neural network architectures tailored for CAD data structures are still relatively uncommon due to the complexity and variety of solid model representation schemes, flexible topological constraints, and non-standardized formats. Existing methods typically focus on B-Rep models and consider only a subset of CAD modeling features. Due to the graph-like structure of B-Rep data, methods primarily use Graph Neural Networks (GNNs) on an attributed face adjacency graph. Geometric information is encoded as node and edge attributes. Cao et al. (2020) extract a face adjacency graph from the B-Rep data structure and apply a GNN, using planar equations of B-Rep faces as node attributes to represent local geometry. This limits their method to B-Rep models with only planar faces. Jayaraman et al. (2021) propose *UV-Net*, which can handle different types of surfaces (e.g., curved, planar, defined by primitives) and edges (e.g., lines, circles, ellipses, helixes). UV-Net discretizes the parameter domain of surfaces and edges into UV grids and feeds them into a CNN, as illustrated in Figure 2.5. The resulting CAD element embeddings are combined using a small GNN on the face adjacency graph. In contrast, *BRepNet* by Lambourne et al. (2021) forgoes positional information and instead focuses on accurate topological modeling. They perform topological walks on co-edges, combined with handcrafted features such as face areas, surface types, and edge convexity. Colligan et al. (2022) propose a two-level graph representation that combines a face adjacency graph to describe the B-Rep topology with a mesh face graph to capture the discretized geometry. Guo et al. (2022) propose to convert manifold B-Rep solids into an implicit representation by constructing a Boolean tree of implicit functions that encompass the B-Rep faces. Hou et al. (2023) introduce an attention-based pooling mechanism to combine topological and geometric information in the GNN.

	Properties	Methods
CAD	+ comprehensive shape information + extended shape modeling information − mix of modeling schemes (B-Rep, CSG, ...) − complex data structures − non-standardized and proprietary formats − data may not be available in a parametric CAD format	CADNet (Cao et al., 2020), UV-Net (Jayaraman et al., 2021), BRepNet (Lambourne et al., 2021), NH-Rep (Guo et al., 2022), H-CADNet (Colligan et al., 2022), FuS-GCN (Hou et al., 2023)
Mesh	+ simple structure + preserves geometric & topological information − sensitive to connectivity − tight input constraints (manifoldness, watertightness, element count, quality)	MeshCNN (Hanocka et al., 2019), MeshWalker (Lahav and Tal, 2020), HodgeNet (Smirnov and Solomon, 2021), DiffusionNet (Sharp et al., 2022), SubdivNet (Hu et al., 2022), TPNet (Li et al., 2023c)
Graph	+ convolution well-rooted in spectral theory − loss of positional information − sensitive to connectivity − spectral methods are domain-dependent, can suffer from scalability issues − spatial methods may suffer from limited expressiveness	GCN (Kipf and Welling, 2016), GAT (Veličković et al., 2017), PNA (Corso et al., 2020), DGN (Beaini et al., 2021), GSN (Bouritsas et al., 2022), Exphormer (Shirzad et al., 2023)
Voxels	+ structured data + natural extension of convolution operator − quantization-related loss of details − high memory and computational cost − octrees and sparse convolutions require specialized data structures	3DShapeNets (Wu et al., 2015), VoxNet (Maturana and Scherer, 2015), OctNet (Riegler et al., 2017), O-CNN (Wang et al., 2017), MinkowskiNet (Choy et al., 2019)
Multi-view	+ reduces a dimension + can use pretrained vision foundation models with off-the-shelf weights − quantization-related loss of details − dependency on good coverage of views, misses occluded parts	MVCNN (Su et al., 2015), RotationNet (Kanezaki et al., 2018), View-GCN (Wei et al., 2020), MVTN (Hamdi et al., 2021), ViewFormer (Sun et al., 2023), MVCVT (Li et al., 2023a)
Point cloud	+ simple and flexible + common denominator of 3D shape data + widespread use and extensive research − loss of topological information − under-sampling loses small features	DGCNN (Wang et al., 2019), PCT Guo et al. (2021), PointMLP (Ma et al., 2022), PointNeXt (Qian et al., 2022a), StratifiedFormer (Lai et al., 2022), DeLA (Chen et al., 2023a), Mamba3D (Han et al., 2024)

Table 2.1: Comparing input representations derived from CAD models for 3D object recognition.



(a) Face adjacency graph (b) UV-grids for curves and surfaces

Figure 2.5: Processing of the B-Rep data structure in UV-Net presented in Jayaraman et al. (2021), © 2021 IEEE. B-Rep topology is transformed into a face adjacency graph. Geometric entities (trimmed surfaces and curves) are discretized into UV grids, processed by CNNs, and stored as node and edge attributes in the graph.

Meshes & Graphs

Meshes provide a simplified and standardized representation of CAD shape boundaries. Many GNNs have a generalized formulation of the convolution operator, which allows them to be applied across graphs, manifolds, and meshes. Existing convolutional GNN methods can be broadly categorized into spectral and spatial filtering approaches.

Spectral methods define convolution in the Fourier domain. These methods rely on the eigendecomposition of the graph Laplacian to transform graph signals into the spectral domain, where convolution is performed as a multiplication of the signal with learned filters. Bruna et al. (2013) first introduced the convolutional operator in the spectral domain, allowing CNNs to be extended to graphs. Subsequent works by Defferrard et al. (2016) and Kipf and Welling (2016) improve the computational efficiency by approximating filters and simplifying the convolution process. A key limitation of spectral methods is that the learned filters are specific to the structure of the graph, making it difficult to generalize the models to new graphs or shapes. As a result, spectral methods are well suited for single-graph tasks but struggle with tasks that require comparison across multiple graphs or shapes.

Spatial domain methods apply convolution to local graph neighborhoods such as pseudo-Euclidean neighborhoods. Monti et al. (2017) formulate the patch operator as a mixture of Gaussian kernels on local pseudo-coordinates, such as relative polar coordinates. In *SplineCNN*, Fey et al. (2018) propose a

continuous convolution function based on B-splines with a computation time independent of the kernel size. *DiffusionNet* by Sharp et al. (2022) aims to achieve robustness against resolution variance by learning a diffusion process with variable diffusion time.

GNNs using the typical message-passing architecture have been shown to be at most as expressive as the Weisfeiler-Lehman isomorphism test (Xu et al., 2018a) in distinguishing non-isomorphic graphs. *DGN* (Beaini et al., 2021) and *GSN* (Bouritsas et al., 2022) surpass this expressiveness by incorporating asymmetry and substructure information, respectively, into the message passing process.

Hanocka et al. (2019); Lahav and Tal (2020); Hu et al. (2022); Li et al. (2023c) propose GNN architectures specialized for mesh structure by introducing operators that specifically operate on mesh vertices, edges, or faces. Hanocka et al. (2019) propose a pooling operator based on edge collapse. Lahav and Tal (2020) perform random walks on mesh edges that are fed into a Recurrent Neural Network (RNN). Meshes discretize CAD models while preserving both geometric and topological shape information. Limitations of mesh-based neural networks include sensitivity to the mesh connectivity, such as the meshing algorithm and settings being used, and strict input constraints. GNNs often assume that the input meshes are manifold and watertight. In practice, non-watertight conditions can occur for a variety of reasons, including errors during construction, mesh generation, and file conversion. The standard STL format is particularly susceptible to floating-point precision issues because it lacks an explicit definition of connectivity. In addition, GNN methods tend to perform poorly on anisotropic meshes because they typically require well-formed faces, regular connectivity, and coarse resolutions for optimal performance.

Voxel Grids & Multi-View

Conversion to Euclidean-structured domains allows intuitive generalization and application of standard CNNs. Volumetric voxel grids divide the domain into a 3D lattice of voxels, each of which expresses its occupancy as a binary state or distance function. Wu et al. (2015) and Maturana and Scherer (2015) made early attempts to generalize convolutional architectures from 2D images to 3D voxel grids. However, a key limitation of these networks is that the regular structure of voxel grids can unnecessarily inflate the data volume, because all regions of a shape, including unoccupied spaces, are modeled equally. Memory and computational costs grow cubically with the grid resolution, making voxel grids impractical at high resolutions. Coarse grids, on the other

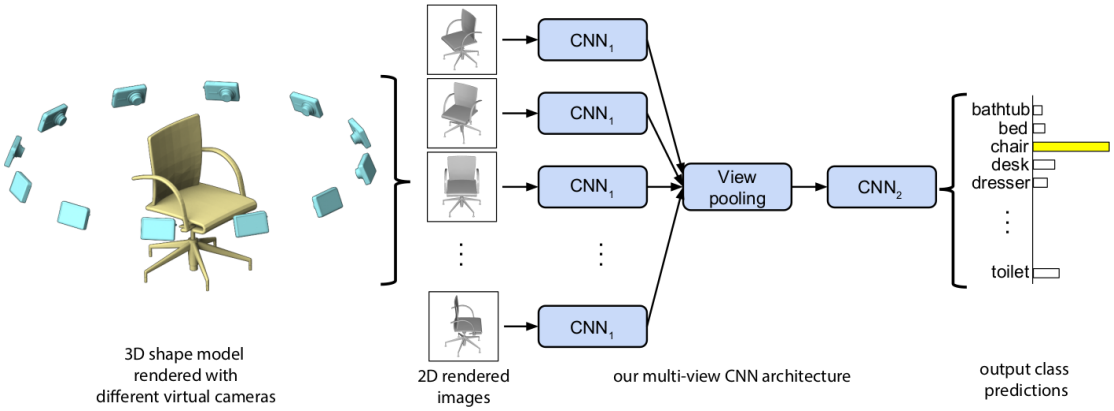


Figure 2.6: Multi-view CNN architecture for 3D shape recognition presented in Su et al. (2015), © 2015 IEEE. At test time, an object is rendered from 12 different views, which are passed separately through a CNN. Features are pooled across views and passed through a second CNN.

hand, introduce a significant loss of detail due to quantization. One line of methods to solve this problem are octree-based methods (Riegler et al., 2017; Wang et al., 2017), which recursively subdivide occupied voxels to a maximum depth. These methods require special considerations for efficient application and parallelization of convolutions, which are computed only on voxels of the same depth. Another line of methods uses sparse convolutions (Choy et al., 2019; Graham et al., 2018) to increase computational efficiency. While voxel-based approaches are less commonly applied to object recognition, they are widely used in LiDAR-based 3D object detection due to the highly sparse and systematic sweep patterns of points (Zhou and Tuzel, 2018; Deng et al., 2021; Chen et al., 2023b).

Another structured representation is multi-view images, which capture a 3D shape from multiple viewpoints. The stack of views is fed into 2D CNNs and the resulting features are fused. Multi-view models differ mainly in aspects of view selection and fusion of learned features across views. Su et al. (2015) propose *MVCNN*, which takes renderings captured by a set of surrounding virtual cameras as input and uses a simple pooling layer to fuse view-wise features, as depicted in Figure 2.6. Wei et al. (2020) improve on the aggregation of view information by constructing a small view graph to learn relations between views. Sun et al. (2023), on the other hand, devise an attention module to learn multi-view correlations. Hamdi et al. (2021) use a differentiable renderer to regress optimal adaptive viewpoints in an end-to-end pipeline.

Multi-view methods can handle higher resolutions compared to standard grid-based methods that do not employ specialized architectures such as octrees

or sparse convolutions. In addition, they can directly apply large image foundation models, benefiting from sophisticated architectures and off-the-shelf pretrained weights. Difficulties with multi-view approaches are related to view selection issues and self-occlusion. In the context of mechanical CAD models, self-occlusion can present a significant challenge because much of the functionality resides within the covering of the object. Rendering the object from surrounding viewpoints misses these hidden functional features.

Point Clouds

Point clouds can be sampled directly from CAD or mesh surfaces. They can also be obtained from sensor data, such as during manufacturing inspection or CAD reconstruction. As such, point clouds serve as a common denominator for different shape models. Point clouds are characterized by a simple yet flexible structure that effectively bypasses the combinatorial complexity associated with graphs and meshes. At the same time, they preserve the original geometric information in 3D space without discretization. Weaknesses of the point cloud representation include the loss of topological information and the loss of small design features (e.g., sprockets, notches, fillets) when under-sampling.

Significant research has been devoted to deep learning on point sets. In our work, we primarily use point cloud inputs because of their flexibility, ease of use, widespread adoption, and extensive research supporting them. The following section discusses techniques for applying neural networks to point clouds and introduces relevant work.

2.2.2 Deep Learning on Point Clouds

Given their extensive applications in robotics, autonomous driving, remote sensing, and 3D printing, point clouds are among the most studied three-dimensional input representations for machine learning.

Point clouds are characterized by the following properties:

- they have **irregular** sampling density in different regions of an object or scene,
- they are **unstructured**, so they cannot be placed on a regular grid, and
- they are **unordered** sets by definition, meaning they remain unchanged under any rearrangement of points.

A point cloud is represented as an unordered point set $P = \{p_1, \ldots, p_n\}$, typically sampled from a surface in three-dimensional Euclidean space. Each point p_i is optionally associated with c-dimensional features, such as surface normals or colors. Neural networks are trained to approximate a function $f : \mathcal{X} \to \mathcal{Y}$ that transforms the geometric input domain $\mathcal{X} \subseteq \mathbb{R}^3$ into a target space $\mathcal{Y}$. In the case of classification, for instance, $\mathcal{Y}$ is a set where $|\mathcal{Y}|$ represents the number of classes.

Early efforts in deep learning on point clouds involved converting the data into a structured input representation, such as voxel grids (Wu et al., 2015; Maturana and Scherer, 2015) or multi-view images (Su et al., 2015). However, this renders the data unnecessarily voluminous and can introduce quantization-related artifacts and information loss. Subsequent research has shifted to the direct application of convolutional architectures to point sets. This requires special design considerations to effectively handle their irregular structure. A major concern is the design of built-in permutation invariance. Other design choices revolve around handling hierarchical point relationships and achieving invariance under rigid motion transformations.

According to Qi et al. (2017a), there are three strategies that can be applied to make a model invariant to permutations of the input point set:

1. Establish a **canonical order**. *PointCNN* by Li et al. (2018) learns a so-called χ-transformation of the input points, which weights the input features and permutes them into a latent ordering where convolution can be applied. This approach is able to model spatially-local correlations, yet fails to fully preserve permutation invariance. In fact, Qi et al. (2017a) show that a canonical ordering that is stable with respect to point perturbations does not generally exist in higher dimensions.
2. Treat the points as **sequential input** and perform permutation augmentation. *G+RCU* by Engelmann et al. (2017) and *RSNet* by Huang et al. (2018) use RNNs to project features from unordered input points onto a structured sequence of feature vectors. This allows them to capture local dependencies in point clouds. However, RNNs face scalability issues with larger point cloud sizes.
3. Apply a **symmetric function** to aggregate global information. *PointNet* by Qi et al. (2017a) and *Deep Sets* by Zaheer et al. (2017) both learn local features independently for each point and aggregate information via global pooling. This simple approach achieves implicit permutation invariance. A major drawback is the loss of information during pooling.

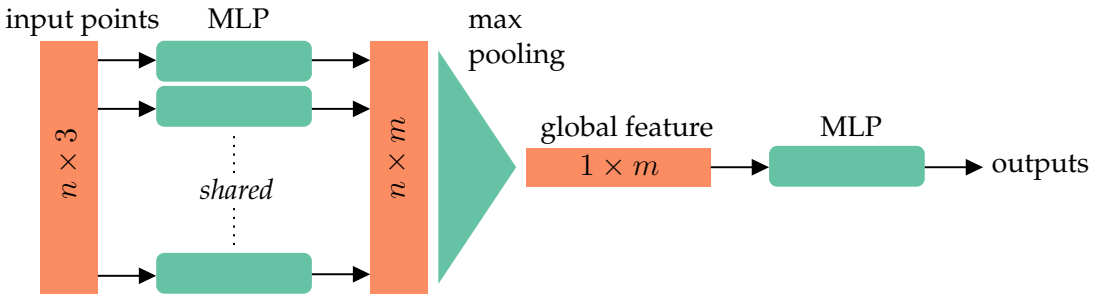


Figure 2.7: A simplified PointNet architecture. PointNet learns pointwise features using shared weight MLPs and subsequently applies max pooling to extract global information.

Techniques for applying neural networks to point clouds can be divided into methods that apply Multi-Layer Perceptrons (MLPs) to point sets, methods that attempt to generalize convolutions to point sets, and methods based on transformer architectures.

MLP-based Methods

Qi et al. (2017a) introduced with *PointNet* a pioneering work for the direct application of neural networks to point sets. To achieve permutation invariance, PointNet approximates a general function f defined on a point set by applying a symmetric function to transformed elements in the set:

$$f(\{p_1, \ldots, p_n\}) \approx \mathcal{A}(h(p_1), \ldots, h(p_n)), \tag{2.1}$$

where h is a pointwise MLP network and $\mathcal{A}$ is a symmetric aggregation function like max pooling. PointNet learns spatial encodings for each point independently. The global signature of the point cloud is subsequently computed as the maximum response across all points. Figure 2.7 shows a simplified depiction of the PointNet architecture. By design, PointNet does not capture spatially-local correlations, as features are learned independently for each point. Exploiting the local structure and multi-resolution hierarchy has proven to be essential for the generalization capability and overall success of neural architectures. Several follow-up works of PointNet address this issue by adopting a hierarchical feature aggregation scheme or by modeling point cloud neighborhoods using graphs.

In their work *PointNet++*, Qi et al. (2017b) extend the PointNet architecture with a hierarchical structure. PointNet++ first divides a point cloud into overlapping local regions. It then builds a hierarchy of set abstraction levels on top of this

partition to progressively group and abstract larger local regions. Each set abstraction level consists of a sampling layer, a grouping layer, and a PointNet layer. The sampling layer performs iterative farthest point sampling to generate a subset of centroids. The grouping layer performs a ball query to find all neighboring points within a given radius for each centroid. Finally, a PointNet layer is applied to each local region to aggregate the neighborhood information into the feature vector of the centroid. PointNet++ also proposes the use of multi-scale and multi-resolution grouping to handle feature learning under non-uniform sampling density.

DGCNN by Wang et al. (2019) dynamically constructs local neighborhood graphs induced by proximity in feature space at each layer of the network. Instead of learning on individual points and subsequently grouping the information, its EdgeConv operator works directly on the relationship between a point and its neighborhood by learning embeddings for the edges of the local neighborhood graph. EdgeConv dynamically builds a local neighborhood graph from the k-nearest neighbors of each point in the feature space. Since the graphs are constructed by proximity in feature space, deeper layers of the network group semantically similar structures rather than spatially-local structures. Due to the dynamic k-NN graph construction, DGCNN has a higher computational cost than PointNet.

In *PointMLP*, Ma et al. (2022) emphasize the need for deep networks instead of sophisticated local geometric extractors. They suggest the use of lightweight geometric affine modules combined with deep residual MLPs for effective point cloud analysis.

In their work *PointNeXt*, Qian et al. (2022a) found that many performance gains since the invention of PointNet++ are due to improved training strategies instead of more sophisticated model architectures. They perform extensive studies of model training and scaling strategies of state-of-the-art point cloud networks. They then present training strategies that improve the performance of PointNet++ to that of state-of-the-art methods. In addition, they propose architectural changes to PointNet++ that improve model scaling with an inverted residual bottleneck design and separable MLPs, naming this improved network PointNeXt.

Convolution-based Methods

Because MLPs have limited ability to capture spatial-variant information, another group of methods attempts to apply convolutional kernels directly to

point sets. CNNs use convolutional kernels with shared weights to slide along regular grids and acquire translation-equivariant responses. Generalizations to point clouds can be distinguished by the use of continuous or discrete convolutions.

PointCNN by Li et al. (2018) learns a χ-transformation of the input points that permutes them into a latent and potentially canonical ordering. This transformation is associated with kernel weights, and a typical discrete convolution is subsequently applied.

Geo-CNN by Lan et al. (2019) divides a point neighborhood into 8 quadrants and computes, for each point in the neighborhood, the angle between the edge vector and the three orthogonal bases $(\vec{x}, \vec{y}, \vec{z})$ of the quadrant to learn direction-associated weight matrices.

In *SpiderCNN*, Xu et al. (2018b) propose a continuous convolution based on a special family of convolutional filters that are the product of a simple step function and an order-3 Taylor expansion defined on the k nearest neighbors. The step function captures coarse geodesic information, while the Taylor polynomial interpolates rich spatial information between vertices.

PointConv by Wu et al. (2019) approximates the weights of a continuous convolution operator with MLPs and re-weights them with an inverse density scale to compensate for non-uniform sampling.

Thomas et al. (2019) introduce in *KPConv* deformable convolutions by learning local shifts applied to the kernel points.

In *PAConv*, Xu et al. (2021) propose a position adaptive convolution kernel by dynamically assembling basic weight matrices which are stored in a weight bank. The coefficients of these weight matrices are learned from point positions via ScoreNet.

Transformer-based Methods

Since the growing success of the transformer architecture (Vaswani et al., 2017) in NLP and image analysis, transformer-based networks have also been adopted for point cloud processing. Transformers have a typical encoder–decoder structure. They are characterized by their self-attention mechanism, positional encoding, and neighborless processing, which enhance their flexibility in capturing local and global structures in unordered data. This flexibility comes at a high computational cost, as the self-attention module has a quadratic time complexity with respect to the number of points.

Guo et al. (2021) propose *PCT*, a global transformer architecture where each of the stacked attention modules operates on the entire point cloud. PCT introduces a neighbor embedding module that aggregates local features to improve the model's ability to capture local geometric patterns. Additionally, it incorporates an improved self-attention mechanism based on the relative positions between points.

Zhao et al. (2021) propose in *PT* a local transformer architecture that operates on local point cloud patches. It adopts the hierarchical architecture of PointNet++ for point cloud processing, replacing the shared MLP modules with local transformer blocks.

StratifiedFormer by Lai et al. (2022) aims to enhance the model's ability to capture long-range dependencies in point clouds. It introduces a stratified sampling strategy for the attention keys that densely samples nearby points while sparsely sampling more distant points. This approach effectively enlarges the receptive field, allowing the model to capture broader contextual information without a significant increase in computational cost.

Park et al. (2023) propose *SPoTr*, which captures long-range dependencies through the use of a local and a global self-attention module. The global attention module computes attention weights based on a small set of self-positioning points to reduce computational cost.

2.2.3 Summary

This section provided an overview of the principles and methods of geometric deep learning. Geometric deep learning extends traditional deep neural networks to non-Euclidean domains. It reduces the high dimensionality of problem spaces by designing model architectures that exploit regularities in symmetry and scale separation.

We examined different 3D representations derived from CAD models to serve as input for deep neural networks. We discussed their respective strengths and weaknesses, along with relevant work, as summarized in Table 2.1. Among these representations, point cloud inputs are primarily used in this work due to their role as a versatile and unifying framework for various sources of 3D information.

Lastly, we reviewed deep learning architectures for point clouds. Point clouds are characterized by being (1) irregular in sampling density, (2) unstructured, and (3) unordered. The extension of neural network architectures to point

clouds primarily revolves around built-in permutation invariance designs. Neural networks applied to point clouds can be further divided into methods that apply MLPs to point sets, those that aim to generalize convolutions to point sets, and transformer-based point cloud networks.

2.3 Self-Supervised Learning on Point Clouds

Self-supervised learning harnesses the intrinsic structure of input data as its guiding signal to bypass the need for human-generated labels. This strategy also marks a shift from traditional supervised learning approaches, which focus on excelling at specific tasks, towards representation learning, which focuses on understanding and capturing the broader structure and relationships within the data.

Self-supervised learning first gained prominence in NLP, where latent word and sentence embeddings were built from a large corpus of text without relying on manual annotations. A seminal example is the language model *BERT* by Devlin et al. (2018), which is trained on several pretext tasks, including masked word prediction. For example, given the sentence

"A [redacted] is worth a thousand words",

BERT is trained to fill in the word "picture" based on the left and right context.

The idea of designing tasks that derive supervision signals from the data itself has since been applied to various domains, including images, video, and 3D data. Analogous to masked word prediction on text data, a typical pretext task on point clouds is to reconstruct masked parts of a 3D object. A comprehensive taxonomy of pretext tasks for point clouds is presented in the survey of Xiao et al. (2022).

This section introduces self-supervised learning techniques for point clouds and reviews relevant literature. Methods are categorized into *generative* (subsection 2.3.1) and *contrastive* (subsection 2.3.2) approaches according to the learning objective. In subsection 2.3.3, we take a look at a third, specialized category of *cross-modal* methods that take advantage of multiple modalities to form self-supervised signals. These methods are particularly relevant to our work because multiple aligned input representations, such as point clouds and images, can be extracted from CAD models. Table 2.2 provides a summary of the methods reviewed.

Method	Type	Task: Learning by ...
FoldingNet (Yang et al., 2018)	G	Deforming 2D grids onto object surfaces
Jigsaw (Sauder and Sievers, 2019)	C	Solving 3D Jigsaw puzzles
Info3D (Sanghi, 2020)	C	Contrasting transformed 3D objects
PointGLR (Rao et al., 2020)	C	Bidirectional hierarchical reasoning
OcCo (Wang et al., 2021)	G	Completing view-occluded point clouds
CMCV (Jing et al., 2021)	M	Cross-modal & cross-view correspondences
STRL (Huang et al., 2021)	C	Spatio-temporal correlation
Point-Bert (Yu et al., 2022)	G	Recovering masked point cloud regions
Point-MAE (Pang et al., 2022)	G	Recovering masked point cloud regions
CrossPoint (Afham et al., 2022)	M	Cross-modal & intra-modal correspondences
IAE (Yan et al., 2023)	G	Reconstructing implicit surface
PointVST (Zhang and Hou, 2023)	M	Point cloud to image translation
TAP (Wang et al., 2023)	M	Point cloud to photo translation

Table 2.2: Overview of self-supervised learning methods on point clouds divided into generative (G), contrastive (C) and cross-modal (M) approaches.

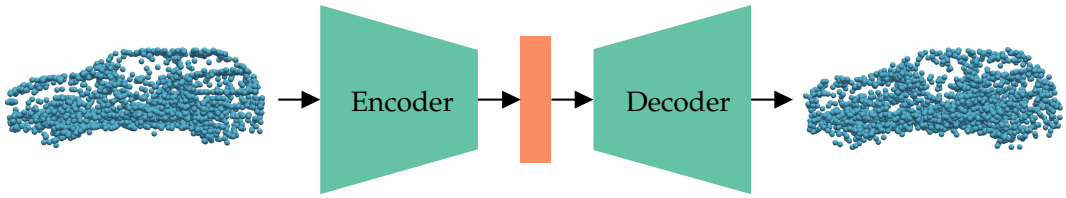


Figure 2.8: Illustration of a point cloud autoencoder.

2.3.1 Generative Learning

Generative learning aims to model the underlying data distribution from which new data points can be generated. Generative self-supervised learning encompasses several methods, including autoencoders, auto-regressive models, flow-based models, and generative adversarial networks. The survey of Liu et al. (2021) provides an overview of generative models for self-supervised learning with applications in NLP, computer vision, and graph learning.

The autoencoder is one of the earliest self-supervised models. It is designed to reconstruct its input at the output layer, as illustrated in Figure 2.8. An autoencoder consists of an encoder f that compresses a point cloud P into a low-dimensional latent representation $z \in \mathbb{R}^d$, referred to as a *codeword*, followed by a decoder g that attempts to reconstruct P with $P' = g(f(P))$. The fidelity of the output reconstruction is measured using permutation-invariant distance metrics such as the Earth Mover's Distance (EMD) or the Chamfer Distance (CD).

Earth Mover's distance calculates the minimum cost of transporting the points in the original set P to match the points in the reconstructed set P' using a one-to-one mapping ϕ:

$$\text{EMD}(P, P') = \min_{\phi: P \to P'} \sum_{p_i \in P} \|p_i - \phi(p_i)\|_2. \tag{2.2}$$

Due to its computational complexity in $O(n^3)$, the Earth Mover's Distance is typically used for evaluation purposes only. Chamfer distance, which is a more common choice for loss computation during training, has a complexity in $O(n^2)$. It measures the squared distance between each point in the original set P to the nearest point in the reconstructed set P', and vice versa:

$$\text{CD}(P, P') = \sum_{p_i \in P} \min_{p'_j \in P'} \|p'_j - p_i\|_2^2 + \sum_{p'_j \in P'} \min_{p_i \in P} \|p_i - p'_j\|_2^2. \tag{2.3}$$

Yang et al. (2018) present *FoldingNet*, an autoencoder with a novel folding-based decoder that deforms a canonical 2D grid onto the input point cloud. FoldingNet aims to reconstruct discretized object surfaces through the implicit 2D grid constraint on the decoder. Because of this constraint, it is able to learn discriminative representations with only a fraction of the parameters of a standard fully-connected decoder.

Instead of improving the autoencoder architecture, Wang et al. (2021) modify the task objective to increase the expressiveness of the learned representations. Motivated by scene segmentation, in their work *OcCo* they occlude point clouds from randomly sampled camera viewpoints. The task of the autoencoder is to complete the full point clouds from occlusion-masked versions.

Masked autoencoding tasks have since gained traction with the growing popularity of transformer architectures. Yu et al. (2022) present *Point-BERT*, an adaptation of the BERT transformer architecture to point clouds. They develop a point tokenizer that uses a discrete variational autoencoder to encode point cloud patches into discrete point tokens, analogous to the tokens of a language vocabulary. The model is tasked with reconstructing masked point patches. A drawback of the Point-BERT model is that its point tokenizer must be trained separately beforehand.

Pang et al. (2022) propose *Point-MAE*, a generalization of the MAE image transformer (He et al., 2022) to point clouds. They improve on several shortcomings of previous architectures, including basing their model entirely on a standard transformer backbone, without any auxiliary model like the point tokenizer

used by Point-BERT. In addition, they propose to feed mask tokens only to the decoder instead of dragging them through the heavyweight encoder to avoid early information leakage and reduce computational cost.

Point cloud autoencoders are subject to sampling variation issues, as the fidelity of the reconstruction is affected by the discrete point sampling captured in the latent code. To avoid this, in their work *IAE*, Yan et al. (2023) proposed the use of an implicit decoder that reconstructs a continuous representation, such as a signed distance function or an occupancy grid, of the 3D object.

Generative models typically reconstruct directly in point cloud space, relying on point set similarity metrics such as Chamfer distance to define the reconstruction loss. However, the use of these metrics is computationally expensive and has been found to be imprecise and hard to optimize (Wu et al., 2021; Huang et al., 2023).

2.3.2 Contrastive Learning

Contrastive learning leverages associations between similar and dissimilar data points to learn a mapping to an embedding space, where the distance between data points reflects a measure of their similarity. As a discriminative approach, contrastive learning focuses on modeling the decision boundary between classes, promoting representations with high inter-class separability and intra-class compactness.

Given an *anchor* sample, contrastive learning pulls similar samples (*positives*) closer, while pushing dissimilar samples (*negatives*) farther apart in the embedding space. Positive samples are considered to be semantically related to the anchor, while negatives are typically chosen to be unrelated samples from the same batch. A common contrastive task is to minimize the distance between original objects and transformed or augmented versions, which promotes robustness of the learned representation to variation and noise.

One of the earliest contrastive losses is the Triplet loss by Schroff et al. (2015), illustrated in Figure 2.9. It trains an encoder network f to produce embeddings that preserve semantic relationships between data points by operating on triplets of input samples, consisting of an anchor x, a positive sample x^+, and a negative sample x^-. The encoder f maps these inputs to a shared embedding space:

$$z = f(x), \quad z^+ = f(x^+), \quad z^- = f(x^-). \tag{2.4}$$

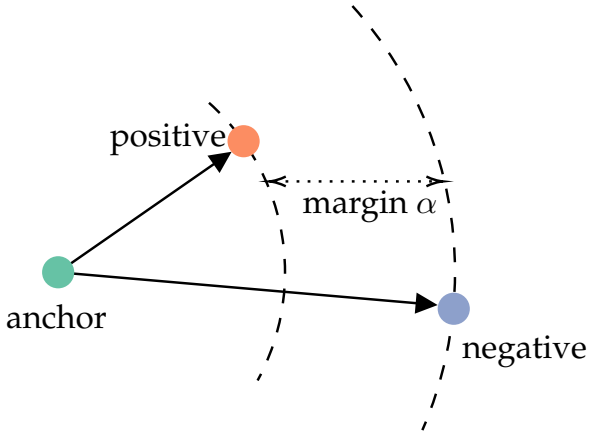


Figure 2.9: Illustration of the Triplet Loss by Schroff et al. (2015). The positive sample is pulled closer to the anchor than the negative sample by margin α.

The Triplet loss encourages the positive embedding z^+ to be closer to the anchor embedding z than the negative embedding z^- by a margin α:

$$\mathcal{L}_{\text{triplet}}(z, z^+, z^-) = \max(0, \|z - z^+\|_2^2 - \|z - z^-\|_2^2 + \alpha). \tag{2.5}$$

One widely adopted form of contrastive learning is *instance-instance* contrast, where contrast is performed at the instance level, typically using augmented versions of the same object. Another form, known as *context-instance* contrast, exploits the associations between local features and their global context. This can involve, for instance, predicting the relative arrangement of parts by exploiting the rich spatial relations inherent in natural image or shape compositions. A representative method for using spatial context is *Jigsaw* by Noroozi and Favaro (2016), which aims to recover the position of shuffled image segments. The Jigsaw method was extended to 3D point clouds by Sauder and Sievers (2019).

Another form of context-instance methods are mutual information maximization methods, which aim to capture the shared information between a sample's local features and its global representation without using explicit spatial context. They identify patterns of mutual dependence within different parts of the data, ensuring that local features contribute meaningfully to the overall structure of the sample. These principles, originally rooted in statistics, were adapted to contrastive learning in two seminal papers by Hjelm et al. (2018) and Oord et al. (2018).

Hjelm et al. (2018) introduce *Deep InfoMax*, which maximizes the mutual information between local patches and the global representation of image

samples to capture hierarchical feature relationships. In their work *Info3D*, Sanghi (2020) extend this idea to 3D shapes. They encourage the network to align global geometric structures with local chunks of 3D shapes and geometrically transformed versions.

Oord et al. (2018) introduce *Contrastive Predictive Coding (CPC)*, which uses autoregressive models to predict future observations by maximizing the mutual information across time steps. A key contribution of their work is the introduction of the InfoNCE loss, which allows the model to simultaneously compare an anchor element to multiple negative samples.

The InfoNCE loss has been widely adopted in the literature, appearing in various forms such as the N-pair loss (Sohn, 2016) and the NT-Xent loss (Chen et al., 2020). Again, let z denote the embedding of an anchor sample, z^+ the embedding of a positive sample, and $Z^- = \{z_1^-, \ldots, z_k^-\}$ a set of k negative sample embeddings. The embeddings are normalized to lie on a unit hypersphere via L2-normalization. The InfoNCE loss is then defined as

$$\mathcal{L}_{\text{InfoNCE}}(z, z^+, Z^-) = -\log \frac{\exp(z^\top z^+/\tau)}{\exp(z^\top z^+/\tau) + \sum_{z_i^- \in Z^-} \exp(z^\top z_i^-/\tau)}. \tag{2.6}$$

In this formulation, the similarity between samples is measured using cosine similarity, which is computed as the dot product of L2-normalized elements. The temperature parameter τ adjusts the sharpness of the probability distribution. Intuitively, the InfoNCE objective seeks to correctly identify the positive sample among the negatives using a softmax classifier.

The importance of incorporating hierarchical knowledge was further explored by Rao et al. (2020) in their work *PointGLR*, where they use bidirectional local-global reasoning tasks for self-supervised representation learning on point clouds. In particular, local-to-global reasoning is realized by maximizing the mutual information between local features and the global representation across multiple abstraction levels. Global-to-local reasoning is realized by learning to recover fine-grained structural information from the global representation with normal estimation and point cloud reconstruction tasks.

STRL by Huang et al. (2021) uses spatio-temporal cues from point cloud sequences for self-supervised learning. Inspired by Grill et al. (2020), they use an online network that tries to predict the representation using a target network that sees a different temporally and spatially correlated input.

The effectiveness of contrastive learning methods is highly dependent on the quality of the information encoded in the data and the design of the self-

supervised objective. One limitation is the reliance on carefully selected negative samples, with the potential for performance degradation if the negatives are either too dissimilar (making the task trivial) or too similar (causing false negatives that confuse the model). Some methods require large batch sizes to provide a sufficiently diverse set of negatives. In addition, instance-instance contrast methods in particular tend to capture a coarse global signal that overlooks local relationships (Qi et al., 2023) and limits the richness of the learned representations.

2.3.3 Multimodal Learning

Different modalities such as point clouds, images, and textual descriptions convey unique perspectives on objects. Combining cues from different modalities contributes to learning rich, comprehensive object representations beyond what a single modality can achieve. Unlike humans, who excel at multisensory perception, machines struggle with the discrepancy in feature subspaces across modalities (Guo et al., 2019). To overcome this, cross-modal methods aim to learn a joint or aligned feature subspace.

The use of natural language to supervise visual or geometric models (Radford et al., 2021; Zhang et al., 2022b) has received attention in the context of zero-shot learning, such as performing recognition on unseen categories. Image data can complement the 3D shape information captured by point clouds by providing a different perspective on the 3D shape. Point clouds provide a flexible description of the overall 3D shape, while image renderings provide detailed, structured cues. Both point clouds and renderings can be easily generated from CAD models.

The field of multimodal learning encompasses a wide range of methods. Some methods use specialized backbone architectures or adaptation processes to adapt vision and language foundation models for 3D point cloud analysis. Prompt tuning methods (Zhang et al., 2022b; Wang et al., 2022) tune 3D shape analysis problems to fit to vision or image foundation models. They freeze the model backbone and modify the input to fit the original task, such as projecting input point clouds to colored images to perform 3D shape analysis using an image transformer. Other methods (Xu et al., 2022; Qian et al., 2022b) apply 2D-to-3D architectural modification, adapting 2D models to handle 3D data. Knowledge distillation methods (Dong et al., 2022; Qi et al., 2023) use cross-modal teacher-student frameworks to transfer knowledge between

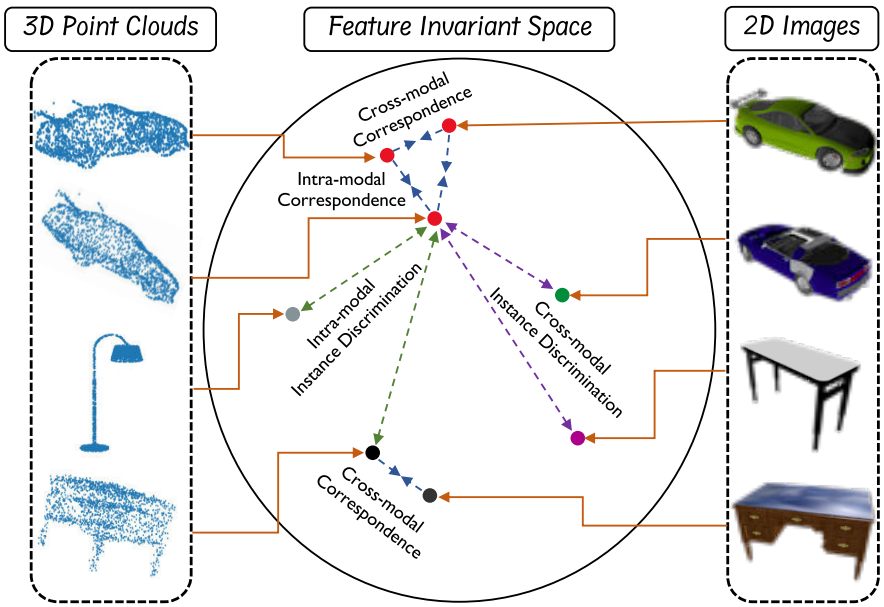


Figure 2.10: Illustration of intra-modal and cross-modal correspondences presented in Afham et al. (2022), © 2022 IEEE. The model maximizes the agreement between point clouds and their corresponding rendered 2D image, while encouraging invariance to transformations in the point cloud modality.

modalities. While these methods can achieve competitive performance, they rely on specialized architectures or adaptation processes, limiting their direct applicability to generic 3D backbones.

Architecture-agnostic approaches design cross-modal pretext tasks directly for point cloud networks, allowing the use of arbitrary point cloud backbone models in pretraining and downstream use. Jing et al. (2021) introduced *CMCV*, an early approach to joint learning of 2D and 3D feature representations through cross-modal and cross-view correspondence. They use separate image and point cloud feature extractors, followed by an MLP to project the features into a joint subspace. Cross-modal correspondence is modeled through a simple binary classification task that determines whether point cloud–image pairs correspond to the same object.

Similarly, in *CrossPoint*, Afham et al. (2022) jointly learn cross-modal and intra-modal correspondence, as illustrated in Figure 2.10. Intra-modal correspondence is established by imposing feature invariance between two versions of augmented point clouds, which are processed by separate feature encoder and projection branches. Cross-modal correspondence is established by contrasting 2D rendering features with point cloud prototype features that are averaged over the two augmented point cloud versions.

PointVST by Zhang and Hou (2023) and *TAP* by Wang et al. (2023) both employ generative approaches to cross-modal learning. PointVST learns to generate silhouette, contour, and depth renderings from point clouds at specified viewpoints. It extracts view-conditioned pointwise embeddings and adaptively aggregates them into a view-specific global representation. Subsequently, 2D CNNs are used to generate and upsample the 2D renderings. TAP uses transformer cross-attention layers to transform point cloud features into pose-conditioned image features, which are decoded into view images.

2.3.4 Summary

In this section, we reviewed self-supervised learning methods for point clouds. Self-supervised learning aims to encode meaningful semantic and structural knowledge into feature representations by using parts of the data itself as a guiding signal. We distinguish between three main types of self-supervised methods for 3D objects.

Generative methods attempt to model the underlying data distribution from which new data points are generated. Common approaches include autoencoders and masked autoencoders. They compress the input shape into a compact latent code from which the original shape, or a subset of masked points, is reconstructed. Reconstruction in point cloud space can be hard to optimize and is prone to sampling variability.

Contrastive methods aim to learn an embedding space in which similar data points are placed close together, while dissimilar instances are pushed apart. In self-supervised contrastive learning, a common objective is to learn invariances or robustness to noise by minimizing the distance to transformed or augmented versions of objects. Another common approach is to incorporate contextual information, such as spatial or hierarchical relationships, into the contrastive task. The effectiveness of contrastive learning is highly dependent on the quality of the chosen positive and negative samples and the overall task design. Contrastive methods that focus solely on instance-level relationships are susceptible to overlooking important structural and local cues.

Cross-modal methods combine information from different modalities in the pretext task to learn richer representations. The additional modality only needs to be available during pretraining, not for any downstream tasks. 2D shape renderings can be incorporated to provide a complementary perspective of the object. Common 2D–3D cross-modal approaches include learning to generate

renderings from input point clouds and learning correspondences between point clouds and their corresponding renderings.

2.4 Discussion

This chapter covered the key concepts and related work for this thesis. We began by introducing the fundamentals of CAD modeling and CAD model representations, discussing solid models, meshes, and shape signatures. Following this, we introduced geometric deep learning for applying deep neural networks to geometric domains and examined different input representations for machine learning that can be derived from CAD models. Given the versatility of point clouds and the substantial body of research supporting their use, this work predominantly uses point clouds as input representations. Most existing methods focus on raw point clouds, typically acquired from sensors. This work addresses the research gap of developing self-supervised pretraining methods tailored to CAD models, which are widely used in digital design. Cross-modal methods are particularly relevant to this work because they take advantage of multiple input modalities extracted from the rich shape information of CAD models.

CHAPTER 3

Self-Supervised Representation Learning on 3D Geometry

The success of deep neural networks is closely tied to the amount of training data available. As network architectures have become more sophisticated, model capacities have increased. For the traditional paradigm of supervised learning, the cost of human-annotated labels has proven to be a barrier to the continued advancement and practical deployment of DNNs (Ericsson et al., 2022). Self-supervised learning offers relief from the annotation bottleneck by allowing pretraining on readily available unlabeled data, so that less labeled data is needed later to solve any target tasks of interest.

The chapter begins with an introduction to self-supervised representation learning in Section 3.1. Building on this, Section 3.2 explores transfer learning strategies based on self-supervised pretraining to improve the data efficiency of machine learning tasks in model-based product engineering. In Section 3.3, we present the results of an experimental study comparing self-supervised and supervised transfer learning strategies for improving the classification of mechanical components.

3.1 An Introduction to Self-Supervised Representation Learning

In this section, we outline the principles of self-supervised representation learning, roughly following the formalism of Ericsson et al. (2022).

3.1.1 Supervised, Unsupervised, and Self-Supervised Learning

Deep neural networks are traditionally trained in a supervised setting, where models rely on a labeled dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, consisting of N samples x_i with corresponding labels y_i, to learn a target task such as object classification. The network typically includes a feature extractor f_θ and a classifier head g_ϕ, parameterized by network parameters θ and ϕ, respectively. The goal is to find optimal parameters θ^*, ϕ^* that minimize a supervised loss function $\mathcal{L}$,

$$(\theta^*, \phi^*) = \underset{(\theta,\phi)}{\operatorname{argmin}} \sum_{(x_i,y_i)\in D} \mathcal{L}(g_\phi(f_\theta(x_i)), y_i). \tag{3.1}$$

Labeled data points are often not readily available in the quantity needed to fit the millions of trainable parameters of a deep neural network. On the other hand, unlabeled data, such as raw online audio and images, is often freely available in abundance.

Unsupervised learning focuses on discovering patterns in unlabeled data through clustering, dimensionality reduction, or generative modeling techniques. Common unsupervised methods include classical approaches such as gaussian mixture models and principal component analysis, as well as deep learning models such as autoencoders, variational autoencoders, and generative adversarial networks.

Self-supervised learning is considered to be a special case of unsupervised learning, since no manual labels are involved. In essence, self-supervised methods generate their own guiding signal directly from the input data, framing the learning objective as a supervised task, as illustrated in Figure 3.1. They construct a so-called *pretext task* on unlabeled data that is guided by a supervised loss function. The training labels, commonly referred to as *pseudo labels*, are automatically inferred from the data itself by exploiting inherent traits and co-occurrence relationships. Pretext tasks often require the model to

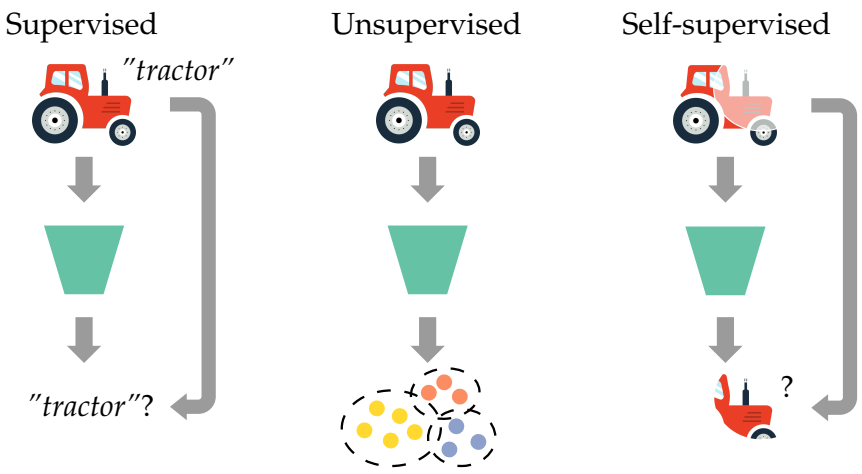


Figure 3.1: Comparison of supervised, unsupervised and self-supervised learning paradigms.

predict hidden or modified parts of the data (e.g., transformed data, masked-out regions, other modalities) from preserved parts.

Formally, given an unlabeled dataset $D = \{x_i\}_{i=1}^N$, the pretext task uses an automated process $\mathcal{P}$ to generate pseudo labels z_i from raw data points x_i (and possibly modify the data points), $\mathcal{P}(D) = \{(x_i, z_i)\}_{i=1}^N$. The pretext model consists of a feature extractor function f_θ and a task head function h_ψ, and is trained to optimize a supervised loss function $\mathcal{L}$:

$$(\theta^*, \psi^*) = \underset{(\theta,\psi)}{\operatorname{argmin}} \sum_{(x_i,z_i)\in\mathcal{P}(D)} \mathcal{L}(h_\psi(f_\theta(x_i)), z_i) \tag{3.2}$$

3.1.2 The Representation Learning Framework

Self-supervised learning is typically used in a two-stage process known as representation learning. As described by Bengio et al. (2013), this process involves "learning representations of the data that make it easier to extract useful information" later. In a broader sense, representation learning is part of the advances toward building general understanding, similar to the way humans learn from accumulated experience.

Self-supervised representation learning consists of a self-supervised pretraining stage followed by a downstream adaptation stage. The automated process $\mathcal{P}$ generates pseudo labels from the unlabeled source dataset. The resulting inputs and pseudo labels are used to pretrain a deep neural network in a supervised manner. After pretraining, the learned weights are (partially)

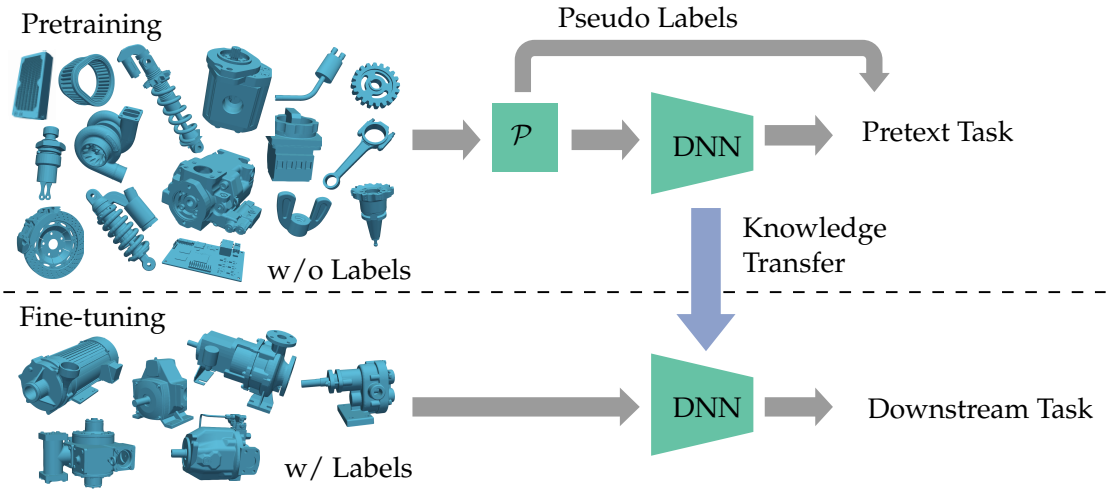


Figure 3.2: A typical self-supervised representation learning workflow with supervised downstream fine-tuning.

transferred, for example to provide initialization for solving a downstream task on a small labeled dataset, as illustrated in Figure 3.2.

Pretraining Stage During pretraining, the model is trained to solve a carefully crafted pretext task. This task is designed in a way that compels the model to develop a good intermediate representation that expresses many general priors about the observed data. The design of the pretext task is what determines how well the learned representation captures meaningful patterns and structures within the data and how effective it is on different downstream tasks. Considerable research effort is devoted to devising effective pretext tasks. Pretext tasks are often combined in multi-task setups to capture a wider range of features and improve the robustness of the learned representations.

Downstream Use After pretraining, the pretext task has served its purpose and the pretext task head h_ψ is replaced by the downstream task head g_ϕ. The intermediate feature representation is retained and transferred to the *downstream* task of interest. The following types of downstream adaptation are distinguished:

1. *trained feature extractor f_{θ^*} as representation function*: On supervised downstream tasks, pretrained feature extractor f_{θ^*} is reused as network initialization for target model $g_\phi \circ f_{\theta^*}$ with downstream task head g_ϕ.

a) *fine-tune*: Fine-tuning involves tuning all layers of the model on the downstream task, usually at a lower learning rate than training from scratch.

b) *linear readout*: In situations where downstream data is particularly scarce, a practical approach is to freeze the feature extractor and train only a simple linear downstream head g_ϕ.

2. *intermediate layer as compressed representation*: For unsupervised applications, such as dimensionality reduction or shape retrieval, the feature extractor f_{θ^*} can be applied to downstream data $\bar{x}_i$ to obtain a compressed data representation $c_i = f_{\theta^*}(\bar{x}_i)$, similar to the latent code extracted by autoencoders. The low-dimensional vector representation is a structured, condensed encoding of the input data that facilitates efficient operations such as similarity comparison on unstructured shapes.

3.2 Data-Efficient Learning on CAD Data

Machine learning applications in product development often face data-related challenges, such as limited data volumes, high data heterogeneity, and costly manual annotation. Self-supervised representation learning mitigates these problems by extracting robust foundational knowledge from large datasets. This acquired knowledge can then be transferred to improve performance in related learning problems where labeled data is scarce.

The remainder of this chapter is dedicated to investigating the effectiveness of transfer learning techniques in improving performance on downstream engineering tasks. The study is organized as follows. First, we examine the product engineering data landscape from a machine learning perspective. Next, we propose a concept for broad acquisition and usage of CAD data based on knowledge transfer. Lastly, we analyze supervised and self-supervised transfer learning strategies to improve the classification of mechanical components.

The remainder of this chapter is based on Herzog and Suwelack (2022).

3.2.1 Data Acquisition and Usage in Product Engineering

Advances in closed-loop product lifecycle management, smart manufacturing, big data, and the Internet of Things have greatly accelerated the generation of 3D data in product development. The accumulated data provides new

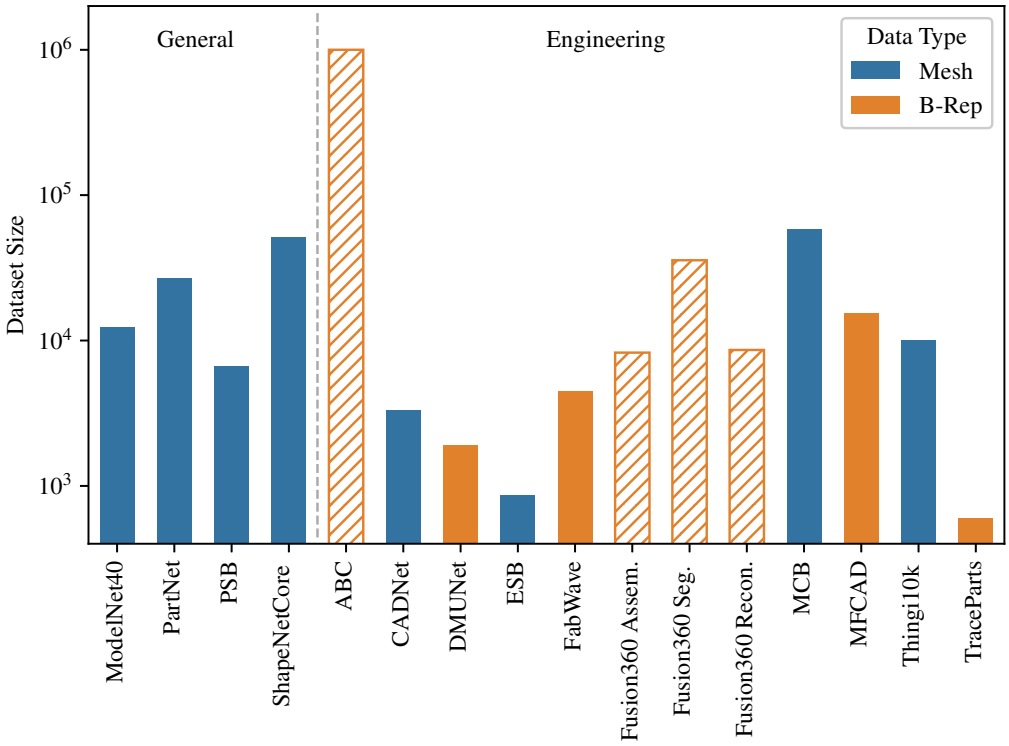


Figure 3.3: Comparison of general-purpose and engineering CAD datasets. The stroked bars denote datasets that do not include a shape taxonomy. Figure roughly adapted from Herzog and Suwelack, © 2022 ASME.

opportunities while simultaneously driving the need for advanced analytical systems (Tao et al., 2018). Still, compared to the vast datasets for machine learning in NLP and 2D vision, which can reach millions to billions of data points (Wei et al., 2021; Schuhmann et al., 2022; Chang et al., 2015), available CAD datasets are typically only a fraction of the size.

Figure 3.3 compares the size of public CAD datasets from different domains. We distinguish between CAD datasets specific to engineering, such as those containing mechanical or electrical components, and more general CAD-based shape datasets from computer vision, which contain various natural and artificial everyday objects.

For engineering, public CAD design libraries such as GrabCAD, TraceParts, 3DWarehouse, and the National Design Repository are valuable resources providing millions of professional and user-created 3D CAD models. However, not all collected CAD datasets contain parametric B-Rep models. In Figure 3.3, datasets marked in blue indicate those that only provide mesh formats (e.g., STL, OBJ), which do not retain parametric features and associated modeling information. Many public engineering datasets also exhibit high variability in

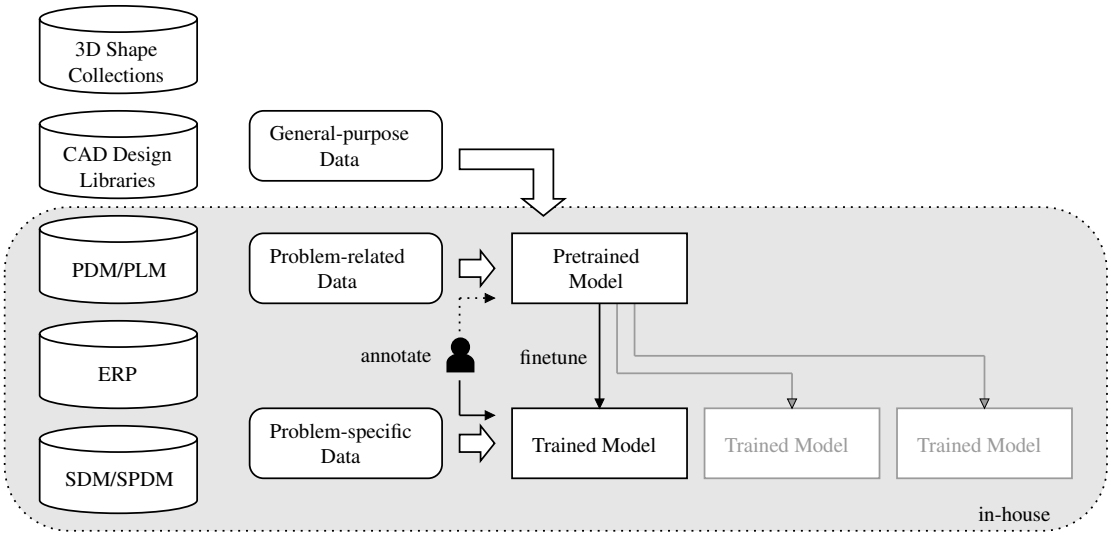


Figure 3.4: Data acquisition and usage concept for product engineering. Figure taken from Herzog and Suwelack, © 2022 ASME.

data and annotation quality, which can limit their informational value and effectiveness for knowledge transfer. For instance, the ABC dataset (Koch et al., 2019a) contains over one million 3D models, but the majority of these models are low in complexity, consisting primarily of simple components such as metal plates and brackets. The potential of these CAD datasets for effective pretraining remains largely unexplored. We investigate this in our experimental study.

In addition to engineering-specific CAD datasets, there are extensive 3D shape collections of everyday objects that are widely used in computer vision. Benchmark datasets such as ModelNet40 (Wu et al., 2015) and ShapeNetCore (Chang et al., 2015) provide well-curated, annotated meshes of common objects. These meshes, which are typically derived from CAD models, share many features that are characteristic of CAD designs. One aspect we explore in the experimental study is whether pretraining on these general 3D shape datasets can improve downstream applications in engineering.

Besides publicly available CAD datasets, in-house data management systems such as Product Data Management (PDM), Enterprise Resource Planning (ERP), and Simulation Data Management (SDM) provide various sources of data throughout the product life cycle. Effectively integrating these diverse data sources for machine learning requires techniques that can handle this broad spectrum of information. Transfer learning techniques are particularly valuable in this context. They allow the creation of large pretrained models to extract knowledge from related unannotated data, which can then be trans-

ferred to improve performance on smaller task-specific datasets. This approach forms the basis of our data acquisition and usage concept, as illustrated in Figure 3.4, which involves aggregating geometric knowledge from related in-house and public shape collections to overcome a lack of problem-specific data.

3.2.2 Transfer Learning Strategies

In the following, we study transfer learning strategies with supervised and self-supervised pretraining to investigate in which scenarios and to what extent geometric reasoning for product engineering can be improved by knowledge transfer from related domains. In particular, the study aims to address the following research questions:

1. Can pretraining on 3D shape datasets with tens of thousands of samples lead to significant improvements on smaller downstream datasets? How does the impact change with the size of the source and downstream datasets?
2. How beneficial is the use of unannotated data for pretraining? In particular, how competitive is self-supervised pretraining compared to supervised pretraining?
3. Does pretraining on widely available 3D shape datasets of everyday objects improve performance on downstream engineering tasks, or does the large domain gap diminish the impact of pretraining?

Supervised Transfer Learning

Traditionally, transfer learning in the field of computer vision has been applied in a supervised context, where large labeled datasets such as ImageNet (Deng et al., 2009) are used to pretrain the backbones of image recognition models. These pretrained models can then be employed either as a fixed feature extractor with frozen layers or, more commonly, fine-tuned for specific downstream tasks.

Supervised pretraining has been shown to yield large improvements in many tasks, including object classification, detection, and segmentation (Sun et al., 2017; Kornblith et al., 2019). Nonetheless, studies suggest that the impact of supervised pretraining may be limited in the case of large domain gaps (He et al., 2019) or when dealing with fine-grained downstream tasks (Kornblith et al., 2019).

Self-Supervised Transfer Learning

In recent years, there has been a trend toward self-supervised pretraining methods, which learn features based on the inherent patterns and structures of the data. Key factors that determine the impact of self-supervised pretraining include:

1. **Data scale and complexity.** Studies suggest a logarithmic relationship between the amount of training data and model performance. Similar findings have been discovered regarding the correlation between the size of pretraining datasets and downstream model performance (Sun et al., 2017). In addition, downstream data complexity has been observed to be closely related to the number of samples required to saturate task performance (Newell and Deng, 2020).
2. **Model capacity.** Limitations in model capacity have been shown to hamper improvements in model performance. Small models improve less than larger models as dataset size increases (Goyal et al., 2019; Sun et al., 2017).
3. **Task complexity.** Advanced difficulty of pretext tasks is necessary to gain in-depth knowledge from large-scale data (Goyal et al., 2019).

Transfer Learning Architectures

Figure 3.5 illustrates the supervised and self-supervised transfer learning architectures used in the experimental study.

To explore supervised transfer learning for 3D engineering tasks, we employ the PointNet++ model by Qi et al. (2017b). PointNet++ performs deep hierarchical feature learning on point sets by progressively aggregating and abstracting local regions at increasing scales, up to a global point cloud representation g. A classification head consisting of three fully connected layers transforms the representation g into class scores.

In supervised transfer learning, the architecture of the pretrained model can be largely retained for the downstream task. The only exception to this is the output layer, which is modified according to the number of target classes. We initialize the downstream model with the weights of the entire pretrained model and replace the output layer.

We study self-supervised transfer learning using the PointGLR model proposed by Rao et al. (2020). PointGLR performs hierarchical self-supervised

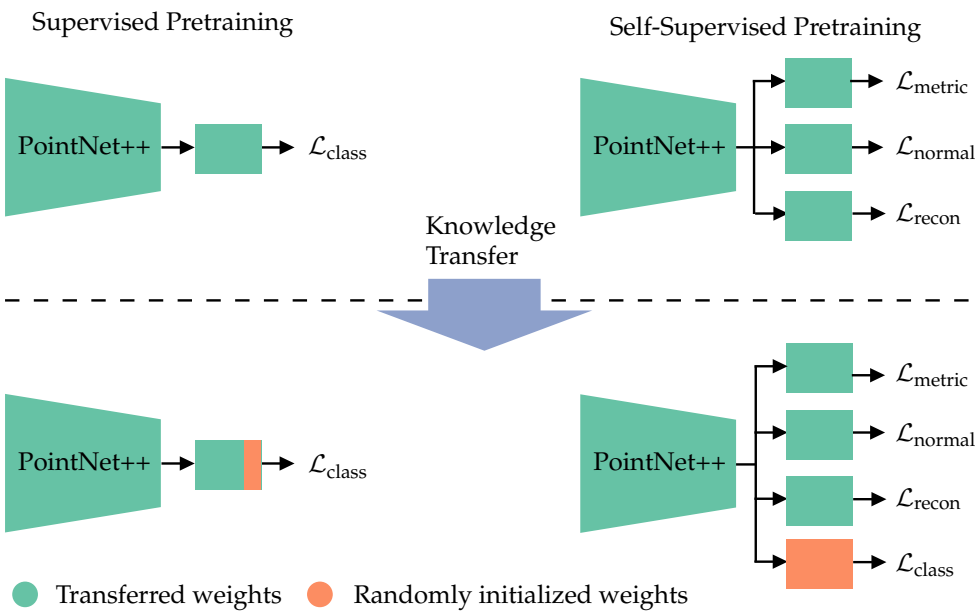


Figure 3.5: Supervised and self-supervised transfer strategies. Supervised transfer learning reuses the entire pretrained model except for the output layer. Self-supervised transfer learning in hybrid mode adds a classification head to the pretrained model. Figure roughly adapted from Herzog and Suwelack, © 2022 ASME.

representation learning through bidirectional reasoning between local and global shape structures using a PointNet++ backbone.

Local-to-global reasoning is learned by discriminating between global shapes upon seeing only local features. This is formulated as a contrastive task that, for a given object, encourages the i-th local representation q_i^l at the l-th abstraction level of the PointNet++ backbone to be closer to the global representation g of that object than to the global representations $G^- = \{g_1, .., g_m\} \setminus \{g\}$ of other objects in the minibatch of size m. Since local and global features have different channel dimensions, they are first projected into a common feature space using projection heads ϕ^l and φ, respectively. The local-to-global loss $\mathcal{L}_{\text{metric}}$ with a total of M local features is defined as

$$\mathcal{L}_{\text{metric}} = \frac{1}{M} \sum_{l \in L} \sum_{i \in N_l} \mathcal{L}_{\text{InfoNCE}}(\phi^l(q_i^l), \varphi(g), \{\varphi(g_j) \mid g_j \in G^-\}), \tag{3.3}$$

where L denotes the set of abstraction levels in PointNet++, N_l denotes the set of local feature representations at the l-th PointNet++ abstraction level, and $\mathcal{L}_{\text{InfoNCE}}$ denotes the InfoNCE loss, as defined in Equation 2.6.

Global-to-local reasoning is achieved by capturing low-level information in the global representation of each point cloud of size n through normal estimation

and self-reconstruction tasks. Normal estimation uses a simple MLP σ to estimate point normals c_i based on a concatenation of point coordinates p_i and projected global representation g, denoted by $p_i \oplus \varphi(g)$. The quality of the predicted normals is measured using a cosine similarity loss:

$$\mathcal{L}_{\text{normal}} = 1 - \frac{1}{n} \sum_{i=1}^{n} \frac{\sigma(p_i \oplus \varphi(g))^\top c_i}{\|\sigma(p_i \oplus \varphi(g))\|_2 \|c_i\|_2}. \tag{3.4}$$

Self-reconstruction uses a folding-based decoder D to reconstruct the original point cloud P using Chamfer distance, as defined in Equation 2.3:

$$\mathcal{L}_{\text{recon}} = \text{CD}(P, D(g)). \tag{3.5}$$

For the self-supervised transfer setup, we initialize with the weights of the pretrained feature extractor and the pretext task heads, as depicted in Figure 3.5. On the downstream task, we use the hybrid learning setting described by Rao et al. (2020), where a supervised classification head, denoted by the loss function $\mathcal{L}_{\text{class}}$, is added to the pretext task heads, which are kept as auxiliary losses to improve the robustness of the learned representations. We follow the classification head architecture as described in the official PointNet++ implementation by Qi et al. (2017b) and use equally weighted sub-task losses $\mathcal{L} = \mathcal{L}_{\text{metric}} + \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{class}}$.

3.3 Experiments

To investigate the research questions defined in subsection 3.2.2, we conduct experiments on the task of object classification on two mechanical component datasets. We compare supervised and self-supervised transfer learning scenarios using the PointNet++ and PointGLR models.

3.3.1 Experimental Setup

Table 3.1 shows the details of the datasets used in the experiments. The downstream datasets DMUNet and MCB-B contain 30 and 25 categories of mechanical components, respectively. These components vary in complexity, ranging from simple nuts and screws to full-size pumps and rotors. For pretraining, we test both mechanical component datasets (ABC and MCB-B) and everyday shape datasets (ModelNet). We note that the ABC dataset

	Use	Type	Labels	#Train	#Test
DMUNet (Dekhtiar et al., 2018)	D	mechanical	✓	958	958
MCB-B (Kim et al., 2020)	D/S	mechanical	✓	14 451	3587
ABC-10k (Koch et al., 2019b)	S	mechanical	×	8000	-
ABC-50k (Koch et al., 2019b)	S	mechanical	×	40 000	-
ModelNet40 (Wu et al., 2015)	S	everyday	✓	9800	-

Table 3.1: Downstream (D) and source (S) datasets used in the experiments. MCB-B refers to part B of the MCB dataset. ABC-10k/-50k refer to the 10k/50k subset of the ABC normal estimation benchmark.

does not have class labels and thus can only be used for self-supervised pretraining.

During preprocessing, all datasets are uniformly sampled with 2048 points on the mesh surface and normalized to zero mean within a unit cube $[-1, 1]^3 \subset \mathbb{R}^3$. Pointwise surface normals are used as additional input features.

Implementation The models are implemented using the PyTorch deep learning library. To ensure a consistent comparison, both models use the same PointNet++ backbone implementation with single-scale grouping, eliminating any architectural and implementation differences. In the original paper (Herzog and Suwelack, 2022), we refer to these models with consistent backbone architectures as P++* and PGLR*, respectively. The models are trained using the Adam optimizer with a base learning rate of $5e^{-4}$ and a decay rate of 0.8 every 20 epochs for a total of 400 epochs, with a batch size of 32 for both the source and downstream tasks. All results are averaged over three runs.

3.3.2 Transfer Learning Results

Table 3.2 and Table 3.3 present the transfer learning results on the downstream datasets DMUNet and MCB-B, respectively. The impact of several factors is analyzed below.

Baselines To assess the impact of different transfer learning scenarios, we first establish baseline results by training the models from scratch with random initialization. The PointNet++ and PointGLR models achieve a classification accuracy of 86.9% and 89.4% on DMUNet, respectively. These results already exceed the official multi-view based results of 82.8% accuracy reported by

Downstream Data	20%	50%	100%
Supervised Pretraining			
PointNet++ from scratch	65.4	80.3	86.9
PointNet++ pretrained on ModelNet	68.6	82.2	88.2
PointNet++ pretrained on MCB-B	**73.0**	84.7	89.7
Self-Supervised Pretraining			
PointGLR from scratch	65.3	81.8	89.4
PointGLR pretrained on ModelNet	71.7	84.6	90.7
PointGLR pretrained on MCB-B	72.2	**84.8**	**91.0**
PointGLR pretrained on ABC-10k	71.0	84.7	89.9
PointGLR pretrained on self	68.5	82.8	-

Table 3.2: Classification accuracy of supervised vs. self-supervised transfer learning on downstream 20–100% DMUNet training data.

Downstream Data	5%	20%	50%	100%
Supervised Pretraining				
PointNet++ from scratch	79.7	89.1	92.8	95.0
PointNet++ pretrained on ModelNet	81.6	89.5	93.1	94.9
Self-Supervised Pretraining				
PointGLR from scratch	80.5	89.5	92.9	95.1
PointGLR pretrained on ModelNet	82.4	90.4	93.4	**95.6**
PointGLR pretrained on ABC-10k	82.3	90.3	**93.7**	95.1
PointGLR pretrained on self	**82.9**	**90.9**	93.6	-

Table 3.3: Classification accuracy of supervised vs. self-supervised transfer learning on downstream 5–100% MCB-B training data.

Dekhtiar et al. (2018) on DMUNet. On MCB-B, the PointNet++ and PointGLR models achieve state-of-the-art classification results of 95.0% and 95.1%, respectively.

Impact of Pretraining On the full downstream DMUNet dataset, pretraining on all source datasets consistently improves classification accuracy compared to the baseline of training from scratch. The best performance is achieved by pretraining on the MCB-B dataset, which has a high similarity to the downstream DMUNet dataset. PointNet++ and PointGLR achieve 89.7% and 91.0% accuracy, respectively. On the full downstream MCB-B dataset, which is about 15 times larger than DMUNet, pretraining on source datasets of similar size does not significantly improve the baseline of training from scratch.

Impact of Downstream Size The performance of the pretrained models is also evaluated on smaller downstream subsets, ranging from 20–100% training data for DMUNet and 5-100% training data for the larger MCB-B dataset. The subsets are created using uniform sampling. The impact of pretraining becomes more pronounced as the size of the downstream training data decreases. For DMUNet, on the 20% training subset, the pretrained model achieved accuracy improvements of +7.6% and +6.9% compared to random initialization. On the downstream MCB-B, increases of +1.9% and +2.4% accuracy are achieved on the 5% subset.

We also test an approach where self-supervised pretraining and fine-tuning are performed on the same dataset, denoted by *pretrained on self*. This approach essentially implements semi-supervised learning in two stages: (1) self-supervised pretraining on the entire dataset and (2) fine-tuning on a subset with available labels. Overall, this strategy also yields good results, providing an alternative pretraining approach for cases where only a small subset of the collected data is annotated.

Impact of Source Data We further analyze the effects of source dataset size and domain. Table 3.4 shows the results of pretraining on a 20% subset of the downstream DMUNet, as illustrated in Figure 3.6. Supervised transfer learning, depicted in blue, shows a significant difference in downstream performance when pretrained on ModelNet compared to pretraining on the same number of MCB-B samples. In contrast, self-supervised transfer learning, shown in orange, exhibits similar performance across all source datasets. This

Source Data	0.1k	1k	4k	8k	40k
Supervised Pretraining					
PointNet++ pretrained on ModelNet	64.2	65.0	67.7	67.6	-
PointNet++ pretrained on MCB-B	64.6	**68.0**	**71.0**	**72.2**	-
Self-Supervised Pretraining					
PointGLR pretrained on ModelNet	**66.3**	67.6	69.2	70.5	-
PointGLR pretrained on MCB-B	66.2	67.2	69.4	70.8	-
PointGLR pretrained on ABC-10/50k	65.1	67.6	68.4	71.0	**72.2**

Table 3.4: Classification accuracy of supervised vs. self-supervised transfer learning for different source datasets and sizes on downstream 20% DMUNet.

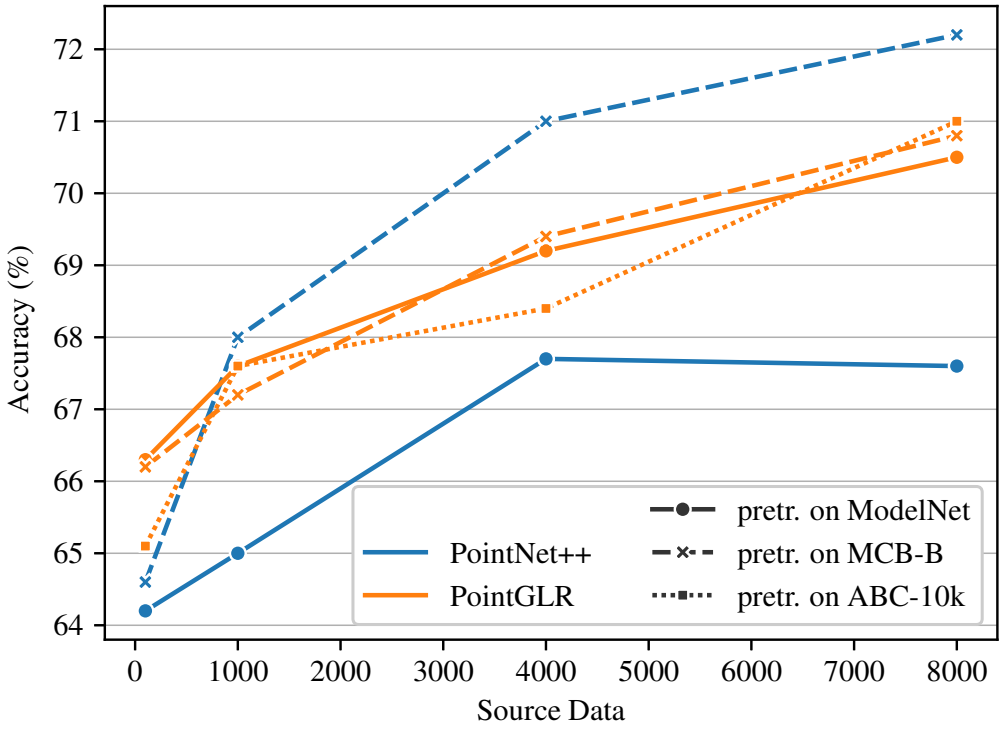


Figure 3.6: Performance of supervised vs. self-supervised transfer learning for different source datasets and sizes on downstream 20% DMUNet. Figure adapted from Herzog and Suwelack, © 2022 ASME.

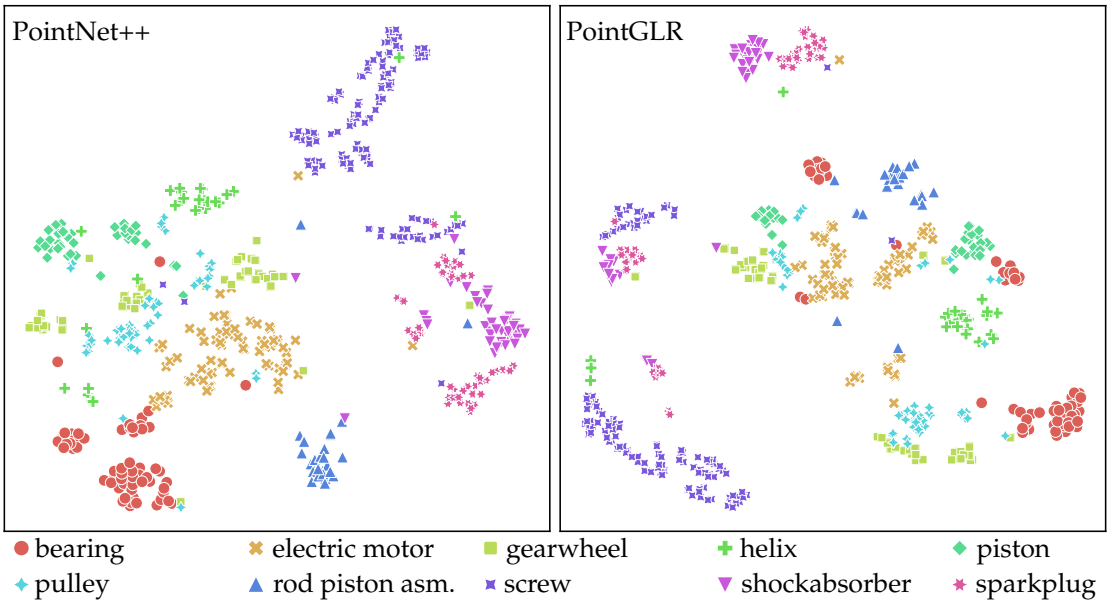


Figure 3.7: Visualization of 2D t-SNE embeddings of PointNet++ and PointGLR global representations for 10 categories of DMUNet. Figure adapted from Herzog and Suwelack, © 2022 ASME.

suggests that supervised transfer learning relies on source data that closely matches with the target domain. Self-supervised transfer learning, on the other hand, maintains robust pretraining performance even for larger domain gaps. This is presumably due to learning underlying structural properties rather than problem-specific class boundaries, allowing self-supervised methods to better adapt to large domain gaps between source and downstream data.

Feature Visualization To gain an intuitive understanding of the models' behavior, we visualize the learned features by embedding the global representations in a 2D space using nonlinear dimensionality reduction with t-SNE (Van der Maaten and Hinton, 2008). The embedding visualization in Figure 3.7 shows the feature distribution of the test samples. For both models, clustering based on class membership is evident. Classes with high geometric similarity lie close together, for example `shockabsorber` ↔ `sparkplug` and `gearwheel` ↔ `pulley`. The feature distribution for the supervised transfer learning model is highly class-driven, with most categories forming one distinct class cluster. In contrast, the feature distribution for the self-supervised transfer learning model is more geometry-driven, with multiple clusters for categories with large in-class geometric differences (e.g., different global shapes or object orientations), such as for the `shockabsorber` and `electric motor` classes.

3.4 Summary

This chapter introduced self-supervised representation learning, an approach that reflects a shift from traditional task-specific training to pretraining versatile representations from large amounts of unlabeled data. During pretraining, a carefully designed pretext task guides the model to learn essential geometric and contextual properties about the observed data. The learned representations can be transferred to reduce the number of data points required to solve downstream tasks.

The chapter proceeded to examine strategies for enhancing the data efficiency in 3D engineering tasks. We proposed a data acquisition and usage concept based on transfer learning techniques to accumulate knowledge across the various data sources available in product engineering. In the experimental study, we evaluated the effectiveness of self-supervised transfer learning compared to supervised transfer learning to improve the classification of mechanical components. Both strategies consistently outperformed training from scratch on target datasets smaller than the source dataset. Notably, the results showed that self-supervised pretraining can be competitive with supervised pretraining and even outperform it under certain circumstances. In particular, self-supervised pretraining proved to be more effective at generalizing across large domain gaps, such as those between everyday objects and mechanical components.

CHAPTER 4

Self-Supervised Tasks for CAD Models

With computer vision being a leading field in artificial intelligence, research in self-supervised learning has primarily focused on pure, raw point clouds. CAD models provide a richer source of information, including detailed geometric and topological descriptions, as well as design metadata such as materials, tolerances, kinematic constraints, and assembly structure. In addition, CAD models possess distinctive characteristics, such as a 2.5D-like structure, large planar surfaces, and sharp edges, which differentiate them from the organic shapes observed in natural environments. These additional details and structural regularities provide opportunities to introduce assumptions that simplify the representation learning process and aid in the design of more effective pretext tasks.

This chapter explores approaches for enhancing representation learning for CAD models. Specifically, we investigate how the rich information and unique features of CAD models can be leveraged to design effective pretext task objectives, as illustrated in Figure 4.1.

The chapter navigates through three different types of CAD-based knowledge:

(1) **Information from native CAD formats** embedded in B-Rep models;

(2) **Design traits** resulting from the shape modeling process, functional requirements, and manufacturing constraints;

(3) **Multimodal information** extracted from CAD models.

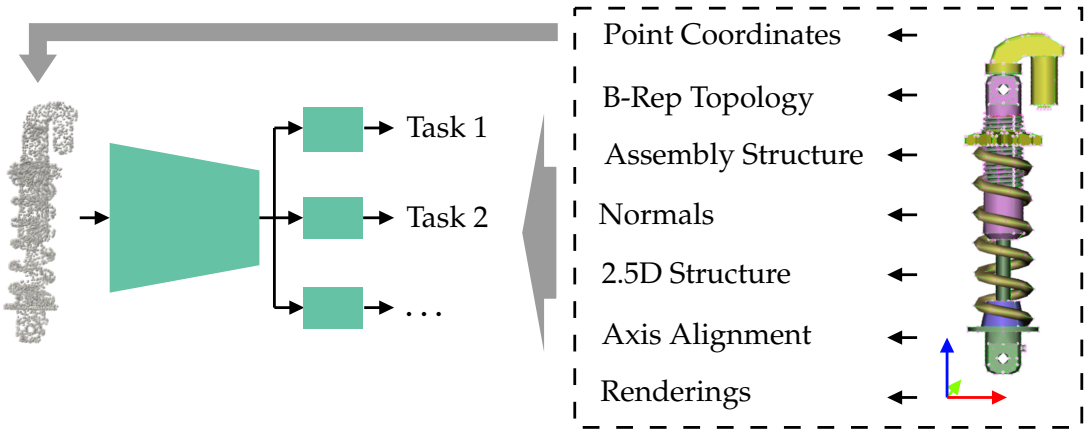


Figure 4.1: Using CAD model-specific information and design traits in the formulation of pretext tasks.

4.1 Leveraging B-Rep Information

B-Rep is the de facto standard representation scheme in parametric CAD modeling. B-Reps define a graph of topological entities associated with geometric carriers that describe trimmed sections of the underlying surface or space curve. Two important knowledge structures in B-Rep models are assembly structure and topological graphs.

Assembly Structure As described by Lupinetti et al. (2019), an assembly organizes the parts of a product into a hierarchical structure according to the designer's needs. Assemblies can vary greatly, reflecting different organizational concepts such as design sequences, functional units, organization for manufacturing, or project planning. Because the assembly structure can be stored independently of the geometry, it may not be extractable from the B-Rep model, but instead appear as a single part. Few methods apply neural networks directly to the assembly structure. These methods typically address specific assembly-related tasks, such as determining the joints between assembly parts (Jones et al., 2021; Willis et al., 2022).

Topological Graphs The topological structure of B-Rep models can be transformed for processing with GNNs. This typically involves converting the data structure into a face adjacency graph. One of the main considerations when applying neural networks to B-Rep structures is how to integrate geometric data from associated curves and surfaces, which may adhere to different modeling schemes (e.g., planar equations, volumetric primitives, NURBS). In

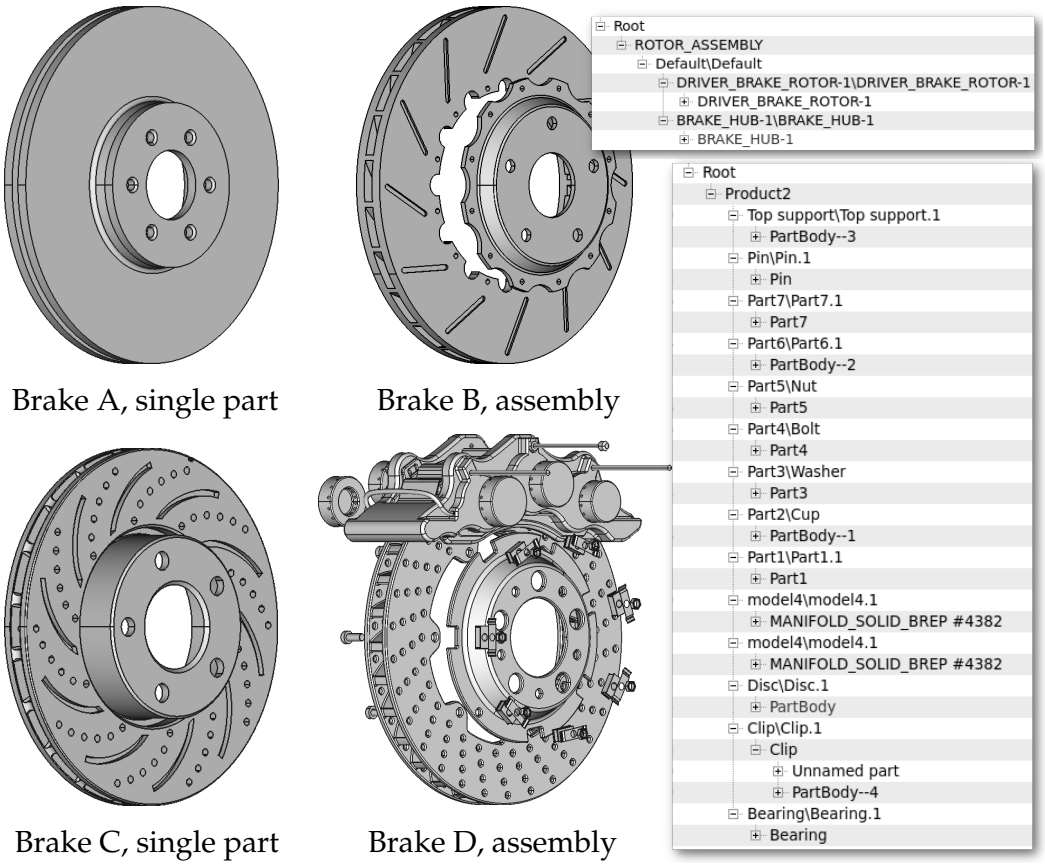


Figure 4.2: Examples of brake discs with large differences in topology and assembly structure, including varied descriptiveness of part names. Assembly models are depicted in exploded view.

UV-Net, Jayaraman et al. (2021) standardize curves and surfaces by performing grid sampling in the UV parameter domain and use 1D and 2D CNNs to extract edge and node features of the graph. Alternatively, Lambourne et al. (2021) propose a method based on topological walks on coedge neighboring structures, focusing on accurate modeling of topological relationships.

A key challenge in applying neural networks to the graph-based knowledge structures of B-Reps models is the heterogeneous nature of this information. CAD models are developed with varying levels of detail at different stages of the design and development process. Early conceptual models tend to be abstract, while later stages require more detailed representations. For instance, prior to running simulations, it is common practice to simplify certain components or replace them with metamodels to reduce computation time. This heterogeneity in levels of detail is exemplified in Figure 4.2, which depicts brake disc models with large differences in both topology and assembly structure.

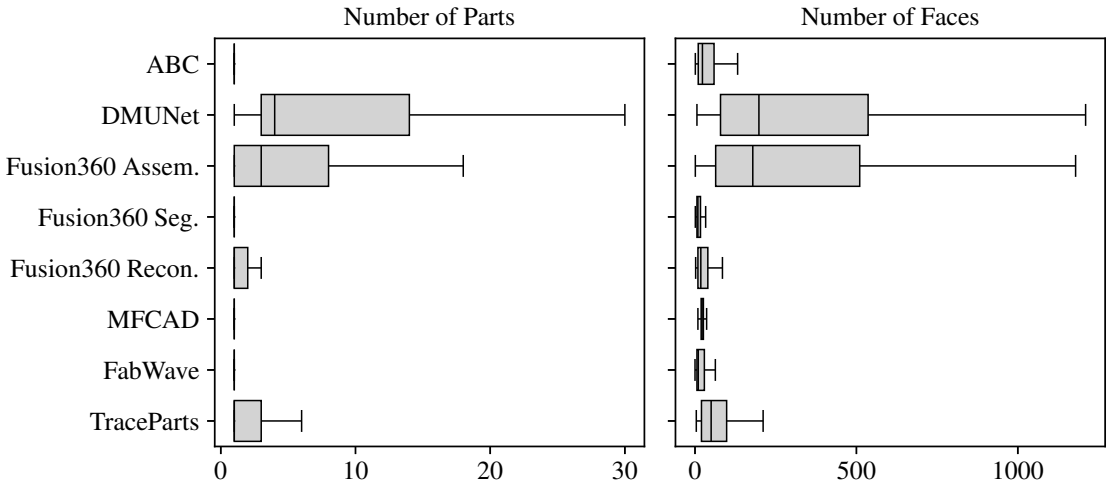


Figure 4.3: Statistics on the design complexity of public parametric CAD datasets. We measure the number of assembly parts and the number of CAD faces.

To assess the potential of using B-Rep information for pretraining, we further examined the quality of CAD models from publicly available datasets. We compiled statistics for eight popular parametric CAD datasets, presented in Figure 4.3. Specifically, we assessed

(1) the **number of assembly parts** per object, which reflects the richness of available product structure information, and

(2) the **number of CAD faces** per object, which indicates the shape complexity.

The results reveal a significant lack of detail in the product structure information in public parametric CAD datasets. The median number of assembly parts per model is close to zero in most datasets, with up to five parts in the DMUNet dataset. In terms of shape complexity, the median number of CAD faces is below 50 for all datasets except for Fusion360 Assembly and DMUNet, which average around 200 faces per model. Both of these datasets show considerable variance in the number of CAD faces.

Overall, the quality of B-Rep data in existing parametric CAD datasets is inadequate in terms of complexity, consistency, and completeness. Many CAD datasets that provide realistic shape models in large quantities favor mesh-based models over native CAD formats because they are easier to store and process. The following sections explore alternative approaches to leveraging CAD information in representation learning that focus on non-parametric shape models.

4.2 Leveraging CAD Design Traits

Exploiting the regularities and underlying structure of physically structured systems to simplify high-dimensional learning problems is a fundamental principle of geometric representation learning, as outlined by Bronstein et al. (2021). According to Bronstein et al., the two main geometric priors that physically structured systems abide by are symmetry and hierarchical organization. Since CAD models are designed as digital replicas of physical objects, these regularities naturally apply.

CAD models are subject to even more regularities stemming from functional requirements, manufacturing considerations, and shape modeling operations. A common regularity observed in CAD models is axis alignment, where components are modeled such that their overall shape aligns with the coordinate axes of the space. Another common feature is the 2.5D-like structure resulting from the extrusion and sweeping techniques used to create the model.

In the following section, we show how to exploit the axis alignment of CAD models in learning discriminative, rotation-invariant feature representations.

4.2.1 90° Rotation Invariance Learning

Invariance to an object's spatial orientation is a desired property in many engineering applications, as it allows for the assessment of functionality and visual appearance independent of orientation in space. While 3D objects captured through sensors are often arbitrarily oriented, CAD models created during product design are commonly axis-aligned. In product design and development, variations in object rotation are related to differences in the configuration of coordinate axes and the alignment of objects along those axes due to design conventions, different software platforms, and integration into larger assemblies. To address these variations while harnessing the axis alignment of CAD models, we aim to develop feature representations that are invariant to 90° rotations around the coordinate axes.

Several deep learning approaches have been proposed to achieve rotation invariance and equivariance on 3D point clouds, including the construction of rotation-invariant features from coordinate inputs (Zhang et al., 2019) and the use of specialized architectures such as spherical CNNs (Cohen et al., 2018; Esteves et al., 2018). In self-supervised representation learning, rotation invariance can be enforced through an auxiliary pretext task, which serves

Algorithm 1: 90° Rotation Invariance Learning with Memory Bank

Data: Embedding function f_θ parameterized by network weights θ, dataset $D = \{x_i\}_{i=1}^{N}$, memory bank M, rotation augmentation function T, batch size m, number of negative samples k

Result: Updated embedding function f_θ

foreach *mini-batch* $\mathcal{B} = \{x_1, \ldots, x_m\}$ *sampled from* D **do**

  for $i \in \{1, \ldots, m\}$ **do**

    $x_i' \leftarrow T(x_i)$;

    Compute embedding $f_\theta(x_i')$;

    Retrieve positive sample v_i from memory bank M;

    Randomly sample k negatives V_i^- from memory bank M;

    Compute loss $\mathcal{L}_{\text{rot}}$ for sample x_i;

  Average loss over mini-batch;

  Update model parameters θ using gradient descent;

  Update memory bank M by replacing with new embeddings;

two purposes. First, it provides an architecture-agnostic approach for obtaining rotation-invariant features. The auxiliary pretext task can be seamlessly integrated in a multi-task setup. Second, it compels the model to focus on intrinsic object characteristics, aiding in the development of robust discriminative feature representations.

The method described in the following uses contrastive learning techniques in combination with data augmentation to achieve 90° rotation-invariant feature representations for CAD models. It builds on the work of Sanghi (2020), a mutual information maximization method that learns to maximize the agreement between objects and their local patches or geometrically transformed versions, including random rotations in $SO(3)$. The described method is employed as part of a multi-task setup to learn vector-based shape representations for shape retrieval in Herzog and Suwelack (2023).

Let T denote a data augmentation function that randomly rotates a point cloud P by multiples of 90° around the coordinate axes. The model is trained to learn a nonlinear embedding function f that maps the input data to an embedding space, where the model induces a distance metric d. To obtain 90° rotation-invariant feature representations, the distance $d(x_i, T(x_i))$ between the original object x_i and any rotation-augmented version $T(x_i)$ is minimized.

The task objective is modeled using contrastive learning techniques, where an anchor sample x_i and its transformed version $T(x_i)$ are pulled together in feature space, while other samples x_j in the batch are pushed away. A direct comparison of $f(x_i)$ with $f(T(x_i))$ would require a forward pass of f for both the original and augmented versions of each sample x_i. To avoid this computational and memory overhead, a memory bank is used to store the features from the previous iteration. Sample $f^{(t)}(T^{(t)}(x_i))$ of the current iteration t is compared to the augmented version of the previous iteration $f^{(t-1)}(T^{(t-1)}(x_i))$, in a trade-off between data staleness and computational demands. In addition, the memory bank allows the number of negative samples to be increased beyond the batch size. This is highly useful for instance-level contrastive learning, which benefits from large batch sizes to compare many samples simultaneously.

The training process is outlined in Algorithm 1. At the beginning of training, the memory bank M is initialized with random vectors, $M = \{v_1, \ldots, v_N\}$. During each iteration t, random data augmentation T is applied to each sample x_i. Positive sample v_i, which corresponds to the same sample from the previous iteration $t-1$, and a set of k randomly sampled negative samples V_i^-, with $v_i \notin V_i^-$, are taken from the memory bank M. At the end of the iteration, the memory bank is updated with the new feature representations. The objective is formulated as an InfoNCE loss

$$\begin{aligned}\mathcal{L}_{\text{rot}} &= \mathcal{L}_{\text{InfoNCE}}(f(T(x_i)), v_i, V_i^-) \\ &= -\log \frac{\exp(f(T(x_i))^\top v_i/\tau)}{\exp(f(T(x_i))^\top v_i/\tau) + \sum_{v^- \in V_i^-} \exp(f(T(x_i))^\top v^-/\tau)}.\end{aligned} \tag{4.1}$$

Rotation invariance methods based on loss optimization and data augmentation are classified as *weak rotation invariance methods* by Fei and Deng (2024). As noted in their survey, a key limitation of loss optimization-based methods for achieving invariance to arbitrary rotations in $SO(3)$ is that the neural network cannot minimize the loss on rotations it has not encountered, making it difficult to give guarantees for these unseen rotations. Restricting the invariance to 90° rotations avoids this problem, as these rotations form a discrete subgroup of $SO(3)$.

4.3 Multimodal Learning

Point clouds are not the only data representation that can be derived from CAD models as input to neural networks. As discussed in subsection 2.2.1, there are several alternatives for representing 3D input data. One simple approach is to generate photorealistic multi-view renderings from CAD models, where a set of virtual cameras is arranged in a regular pattern around the object to capture its appearance from multiple angles. Multi-view renderings effectively capture structured, local cues. On the downside, they can struggle with self-occlusion, which is particularly problematic for mechanical parts where most of the functionality is concealed inside.

To overcome the shortcomings of individual representations, different modalities can be combined to achieve a more holistic object understanding. Multimodal learning has been shown to help increase the richness of representations by integrating diverse cues (Radford et al., 2021; Zhang et al., 2022c; Ruan et al., 2023). Supplementing information from one modality with an additional modality to improve performance is commonly referred to as *cross-modal learning*.

In the following, we discuss cross-modal representation learning and present a self-supervised pretraining method designed to learn from structural point cloud–image correspondences. This section is based on Herzog and Suwelack (2025).

4.3.1 Approaches to 2D–3D Cross-Modal Pretraining

Cross-modal representation learning enhances learned representations for downstream tasks by incorporating an additional data modality during pretraining. Multi-view renderings can be generated from any 3D shape dataset containing CAD models or meshes. Importantly, the additional 2D data is used only during pretraining and is not required for downstream tasks.

Cross-modal pretraining methods can be broadly categorized into generative and contrastive approaches, as illustrated in Figure 4.4. Generative cross-modal methods aim to synthesize images from input point clouds at specified camera views, effectively using image generation as a proxy for learning richer 3D representations. For instance, PointVST by Zhang and Hou (2023) generates view-conditioned point cloud features that are fed into 2D translation heads to generate silhouette, contour, and depth images. Similarly, TAP by

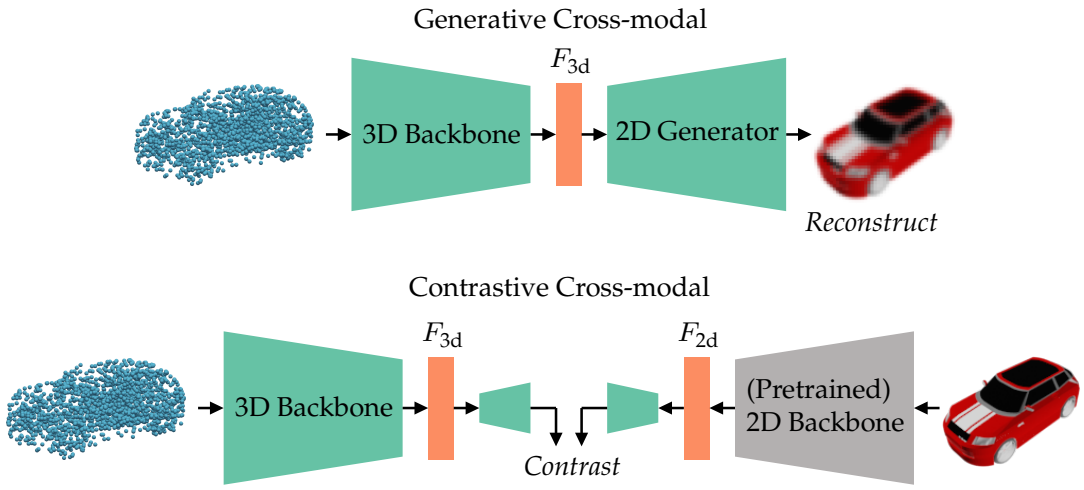


Figure 4.4: Approaches to generative and contrastive cross-modal pretraining.

Wang et al. (2023) applies a cross-attention mechanism to transform point cloud features into image features according to specific poses, subsequently generating images from these views.

A drawback of these generative cross-modal methods is that their typical architecture does not include a pretrained 2D backbone. Instead, they rely on a custom 2D generator built on top of a 3D backbone, using the generated images as ground truth for reconstruction, as illustrated in Figure 4.4. This setup prevents them from leveraging the extensive knowledge embedded in pretrained vision foundation models, which can provide valuable insights from vast image sources to complement the information available from the 3D pretraining dataset.

Contrastive cross-modal methods learn from correspondences between point clouds and images. The seminal work CrossPoint by Afham et al. (2022) jointly performs inter-modal contrast by imposing invariance to point cloud augmentations and cross-modal contrast by maximizing the agreement between point clouds and renderings of the same object. Like most contrastive methods, CrossPoint only considers global point cloud–image correspondences. This can result in a coarse guiding signal that neglects local relationships. Conversely, low-level point–pixel correspondence methods such as Tran et al. (2022) require costly upsampling layers and loss computation, while being unnecessarily fine-grained for learning meaningful contextual relationships.

In the following, we present an efficient contrastive cross-modal method that overcomes the limitations of coarse global guidance by learning 2D–3D correspondences at a structural level.

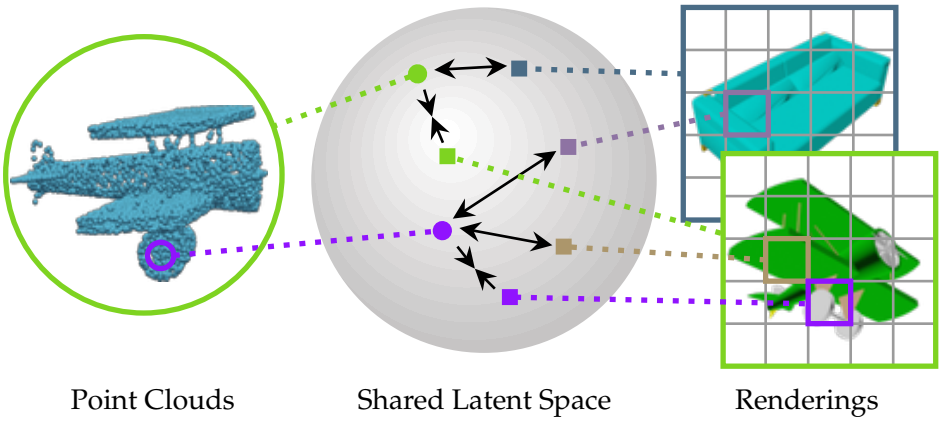


Figure 4.5: Illustration of the proposed structural 2D–3D correspondence. Point cloud and image features of various scales are projected into a shared, modality-invariant latent space, where cross-modal correspondence is enforced. Figure taken from Herzog and Suwelack, © 2025 PMLR.

4.3.2 Structural Point Cloud–Picture Correspondence

We propose a contrastive self-supervised pretraining method, **Pic@Point**, that is designed to learn point cloud representations by exploiting structural features of ***pic***tures ***at*** different ***point*** cloud abstraction levels. Specifically, we extract 3D and 2D features at both global and local scales using a generic 3D backbone and a pretrained 2D backbone, respectively. We then employ global and local pose-conditioned projection heads to project these features into a common, modality-invariant latent space, where the model is tasked with matching point cloud and image features from the same object regions, as illustrated in Figure 4.5. This structural 2D–3D contrastive learning approach offers several advantages over existing methods:

1. It effectively leverages features from pretrained vision foundation models, as opposed to generative cross-modal methods (Zhang and Hou, 2023; Wang et al., 2023).
2. The developed method is lightweight yet effective because it does not use a decoder and employs a frozen 2D backbone. Figure 4.6 shows the size of different pretraining models in relation to linear accuracy on ModelNet40, showcasing the efficiency of our approach.
3. Unlike existing contrastive methods (Afham et al., 2022; Wu et al., 2023) that only learn global shape correspondences, Pic@Point provides guidance at a structural level. By exploiting local correspondences, it provides pose-aware, positional guidance while benefiting from a larger number of contrastive samples.

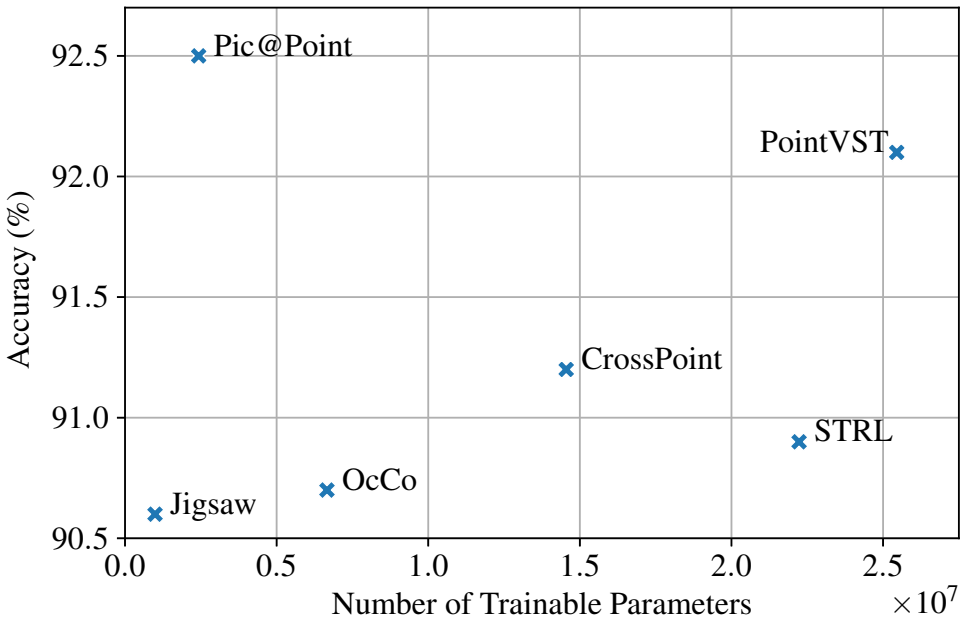


Figure 4.6: Size of pretraining model vs. linear SVM accuracy on ModelNet40. Reported is the number of trainable parameters of self-supervised pretraining models with DGCNN backbone. Figure taken from Herzog and Suwelack, © 2025 PMLR.

The importance of incorporating structural knowledge has been demonstrated in various uni-modal research (Hjelm et al., 2018; Oord et al., 2018; Rao et al., 2020). Our novel cross-modal approach enriches point cloud information with structured, semantic image cues to provide a comprehensive guiding signal for 3D understanding.

Let $D = \{(P_i, I_i)\}_{i=1}^{|D|}$ denote an unlabeled dataset of point clouds $P_i \in \mathbb{R}^{n\times 3}$ and shape renderings $I_i \in \mathbb{R}^{H\times W\times 3}$ of size $H \times W$. Rendering I_i is taken from a random camera viewpoint with view matrix M_{view} and projection matrix M_{proj}. Figure 4.7 depicts the overall architecture of the Pic@Point model. It consists of the following modules:

1. A *3D Backbone* extracts local and global point cloud features, obtained as top-level positional features f^{3d} and final shape embeddings $\mathcal{A}(f^{3d})$, after a global pooling function $\mathcal{A}$.
2. A frozen *2D Backbone* returns top-level local and global image features as extracted by a pretrained vision model (e.g., ResNet, ViT), denoted as f^{2d} and $\mathcal{A}(f^{2d})$.
3. The *Projection Heads* $\mathcal{H}$ project the point cloud and image features into a shared, modality-invariant latent space.

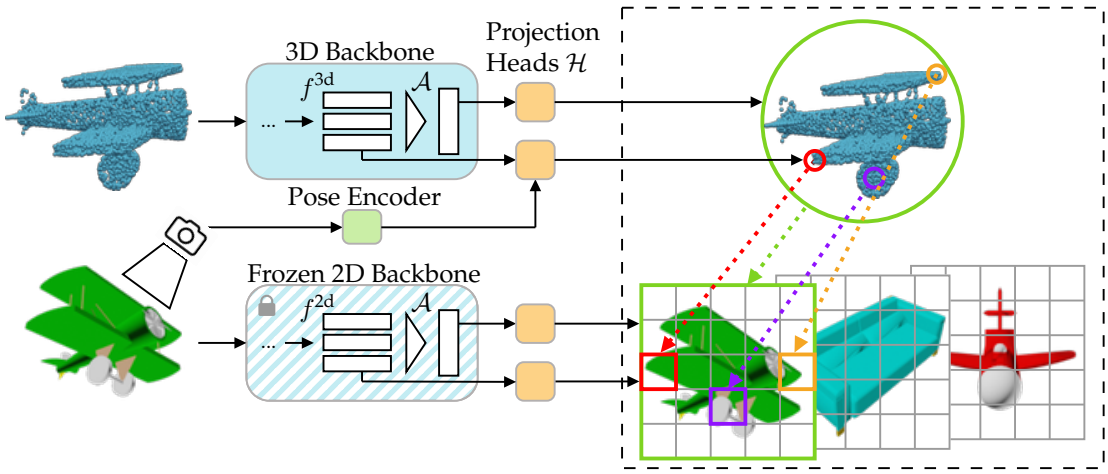


Figure 4.7: Overview of the Pic@Point model architecture. Top-level local and aggregated global point cloud and image features are extracted using a generic 3D backbone and a frozen 2D backbone. Subsequently, the features are processed through global and local, pose-conditioned projection heads $\mathcal{H}$. Figure taken from Herzog and Suwelack, © 2025 PMLR.

Projection into Shared Latent Space

To unify multi-modal knowledge and produce rich, transferable representations, the modal features are projected into a shared latent space, where a contrastive loss is applied between projected features within the mini-batch. The global projection heads $\mathcal{H}^{3d}_{glb}, \mathcal{H}^{2d}_{glb}$ consist of simple two-layer MLPs. The local projection heads $\mathcal{H}^{3d}_{lcl}$, $\mathcal{H}^{2d}_{lcl}$ use Conv1d and Conv2d layers with kernel size 1 and stride 1, respectively, to apply transformations in the channel dimension while preserving the spatial dimensions. All projected features are L2-normalized.

Pose Encoding We facilitate spatially aware correspondence between local point cloud and image features by integrating a pose encoding into the local point cloud projection head $\mathcal{H}^{3d}_{lcl}$. This is necessary because while it is assumed that a complete shape can be uniquely identified in a representation of any modality, this may not hold true for a local shape region. For instance, symmetric features like the red circled plane wing tip depicted in Figure 4.5 could potentially belong to any of the image snippets depicting a plane wing tip, if no additional pose information is provided. The pose encoding is computed via a two-layer MLP that transforms M_{view} into a 64-dimensional vector that is concatenated to the output of the first convolutional layer in $\mathcal{H}^{3d}_{lcl}$.

Cross-Modal Correspondence Task

Point-to-Pixel Mapping For the local cross-modal correspondence task, we begin by establishing the ground truth mapping from each 3D point $(x, y, z) \in \mathbb{R}^3$ to its corresponding normalized image coordinates $(u, v) \in [0, 1]^2$. This is computed via projective transformations using view matrix $M_{view} \in \mathbb{R}^{4\times4}$ and projection matrix $M_{proj} \in \mathbb{R}^{4\times4}$:

$$\begin{pmatrix} u' \\ v' \\ z_{\text{depth}} \\ w \end{pmatrix} = M_{\text{proj}} \cdot M_{\text{view}} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{4.2}$$

Here, u' and v' represent the homogeneous coordinates in the image plane, z_{depth} provides depth information, and w acts as a scaling factor. The final image coordinates (u, v) are obtained by dividing u' and v' by w:

$$u = \frac{u'}{w}, \quad v = \frac{v'}{w} \tag{4.3}$$

Next, we can determine the corresponding image region embedding for this local point cloud embedding centered at (x, y, z) by converting (u, v) to grid positions (i, j). Considering top-level downsampled image regions of size 7×7, as generated by ResNet (He et al., 2016), the indexed position (i, j) of the corresponding image region embedding q_{ij} is calculated as

$$(i, j) = (\lfloor u \cdot 7 \rfloor, \lfloor v \cdot 7 \rfloor). \tag{4.4}$$

With the cross-modal correspondences established, we can apply a contrastive learning objective. For each of the L local point cloud region embeddings $z_1, \ldots, z_L$, we pull close its corresponding image region embedding q^+ of the same object, while pushing away all other image region embeddings in the mini-batch of size m. The set of negative image region embeddings is denoted as $\{q_{ij}^k : (i, j) \in \{1, \ldots, 7\}^2, k \in \{1, \ldots, m\}\} \setminus \{q^+\}$, where q_{ij}^k represents the k-th image region embedding in the mini-batch at indexed spatial location (i, j). To achieve this, we apply the InfoNCE loss function with temperature hyper-parameter τ:

$$\mathcal{L}_{\text{lcl}} = \frac{1}{L}\sum_{l=1}^{L}\mathcal{L}_{\text{lcl}}^{l}, \qquad \text{with} \tag{4.5}$$

$$\mathcal{L}_{\text{lcl}}^{l} = -\log\frac{\exp(z_l^\top q^+/\tau)}{\sum_{i,j,k}\exp(z_l^\top q_{ij}^k/\tau)}. \tag{4.6}$$

Similarly, for global point cloud embedding z, we pull close the global image embedding q^+ of the same object, while pushing away all other global image embeddings $\{q^k : k \in \{1,\dots,m\}\} \setminus \{q^+\}$ in the mini-batch,

$$\mathcal{L}_{\text{glb}} = -\log\frac{\exp(z^\top q^+/\tau)}{\sum_{k}\exp(z^\top q^k/\tau)}. \tag{4.7}$$

The overall loss function is given by $\mathcal{L} = \mathcal{L}_{\text{lcl}} + \mathcal{L}_{\text{glb}}$.

4.4 Experiments

The experiments are divided into two parts. First, we present results from the 90° rotation invariance learning task. In the second part, we provide a comprehensive evaluation of the cross-modal pretraining method Pic@Point in comparison to state-of-the-art self-supervised pretraining methods.

4.4.1 90° Rotation Invariance Learning

This pretext task is integrated into a multi-task framework designed to learn global shape signatures for shape retrieval. A comprehensive quantitative evaluation of this task is conducted as part of the retrieval system in Section 5.4. Below, additional standalone experiments are presented.

Experimental Setup

Following the experimental setup in Section 5.4, we uniformly sample 2048 points from the CAD models and normalize to zero mean within a unit cube $[-1,1]^3 \subset \mathbb{R}^3$. We use pointwise surface normals as additional input features. We employ a PointNet++ backbone, training on a 50% training split of DMUNet and evaluating on the remaining 50% test split. We train for 300 epochs using the Adam optimizer with a base learning rate of $1e^{-3}$ and a batch size of 32.

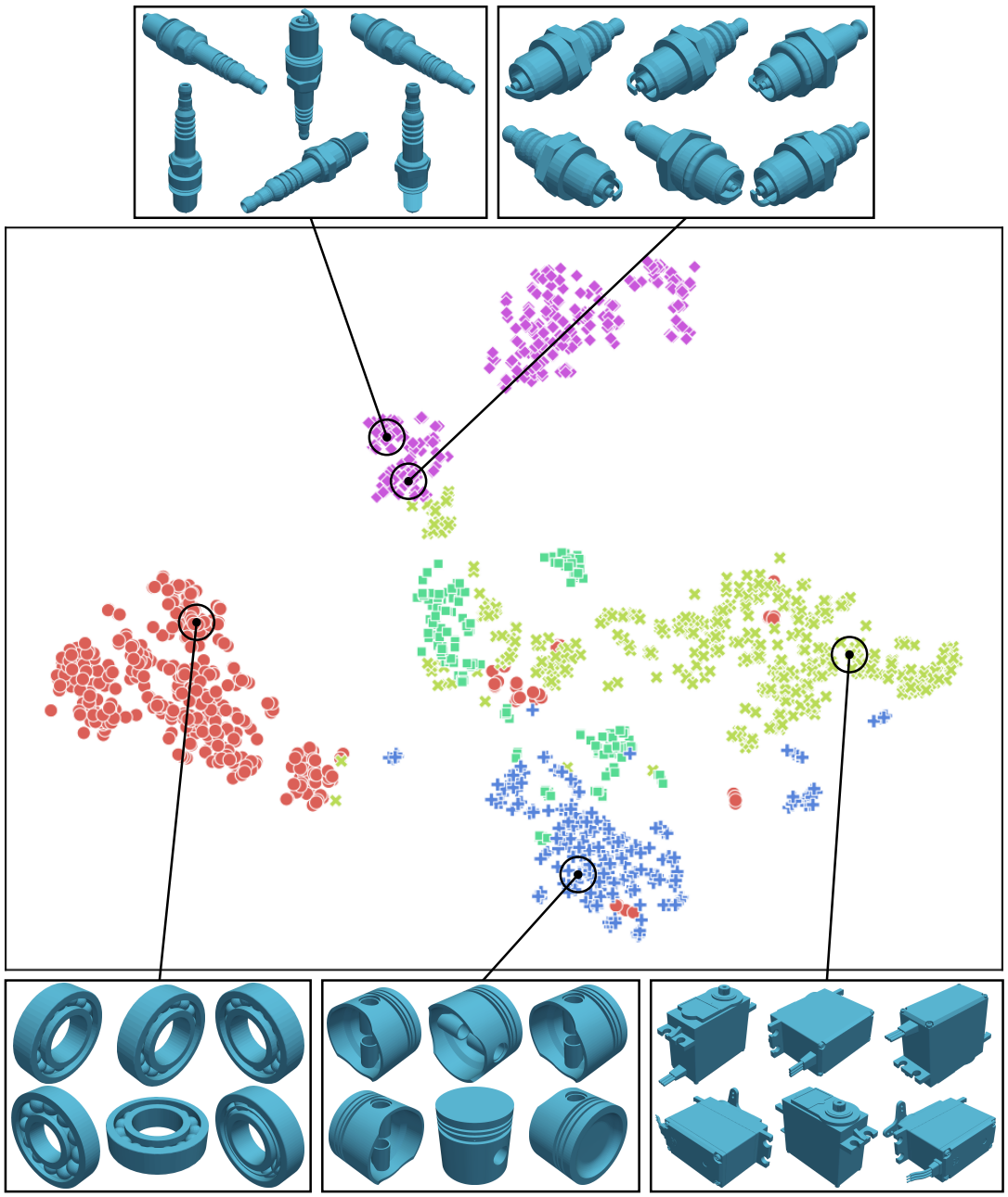


Figure 4.8: Visualization of feature embeddings from 90° rotation invariance learning, dimensionality-reduced via t-SNE. Six 90° rotation-augmented versions of each object are embedded. Example objects (at first position) and their top-5 nearest neighbors in feature space are depicted.

Augmentation	Overall Accuracy (%)
rotation in $SO(3)$	74.0
90° rotations	78.0

Table 4.1: Overall accuracy of linear SVM classification on DMUNet. We compare rotation invariance learning with data augmentation based on random rotations in $SO(3)$ with random rotations by multiples of 90° around the coordinate axes.

Qualitative Results

Figure 4.8 shows the learned feature representations for test objects of five DMUNet classes, visualized using t-SNE for nonlinear dimensionality reduction. Six randomly selected 90° rotation-augmented versions of each object are embedded to assess the consistency with which differently rotated instances of the same object are mapped close in feature space. In addition, the oriented 3D models of example objects and their top-5 nearest neighbors in feature space are depicted. At the top of the figure are two examples of sparkplug objects, and at the bottom are bearing, pulley, and electric motor objects.

It can be observed that visually similar objects are closely embedded in the learned feature space, regardless of rotation. The majority of the nearest neighbors are rotated instances of the same object. In some cases, visually similar yet different objects appear among the closest feature representations. For instance, two of the sparkplug models on the right side do not have insulator ribs. In the case of the electric motor, the third and fifth nearest neighbors show slight variations of the curved cables and an additional side mounting plate.

The distribution of embeddings in the feature space shows distinct clusters based on class categories, demonstrating that the 90° rotation invariance pretext task encourages the neural network to learn class-discriminative feature representations.

Quantitative Results

We evaluate the feature representations learned by applying random 90° rotation augmentation in comparison with augmentations of random rotations in $SO(3)$. Table 4.1 shows the results of linear SVM classification based on the learned feature representations. The results indicate that limiting rotation augmentation to multiples of 90° around the coordinate axes, which exploits

the natural axis alignment of CAD designs, improves class separation in terms of linear SVM classification accuracy in DMUNet by $+4.0\%$.

4.4.2 Cross-Modal Learning

In the following, we present a comprehensive evaluation of Pic@Point pretraining using different point cloud backbones. We conduct experiments on four standard shape analysis benchmarks: ModelNet40 (Wu et al., 2015) and ScanObjectNN (Uy et al., 2019) for object classification, ShapeNetPart (Yi et al., 2016) for part segmentation, and S3DIS (Armeni et al., 2016) for semantic scene segmentation.

Pretraining Setup

Pretraining Dataset Following common practice in related methods, we pretrain on the ShapeNet (Chang et al., 2015) dataset, which consists of more than 50 000 CAD models from 55 semantic categories. We obtain point clouds by randomly sampling 1024 points from each object. Furthermore, we obtain renderings of size 224×224 from 20 different viewpoints placed in a regular dodecahedron around the object, saving projection and view matrices $M_{\text{proj}}, M_{\text{view}}$. We randomly select a single rendering per object at each iteration and apply random rotation as augmentation.

Architectures We conduct experiments using two prominent point cloud backbones: DGCNN (Wang et al., 2019) and PointNeXt (Qian et al., 2022a). DGCNN is a standard architecture used for benchmarking many existing self-supervised methods. PointNeXt is a more recent hierarchical model achieving state-of-the-art results across many point cloud benchmarks. Besides drawing comparison to related contrastive and generative cross-modal methods using the same backbones, we also compare against recent transformer-style methods. For the image backbone we use a light-weight ResNet-18 (He et al., 2016).

During the pretraining phase, the global and local point cloud features are extracted directly from the 3D backbone. Subsequently, the features are projected to shared latent space with dimension $d = 512$. After pretraining, the projection heads are dropped, and the 3D backbone is used exclusively.

Method	ModelNet40	ScanObjectNN
FoldingNet (Yang et al., 2018)	88.4	-
VIP-GAN (Han et al., 2019)	90.2	-
Point-BERT (Yu et al., 2022)	87.4	-
Point-MAE (Pang et al., 2022)	91.0	77.7
Point-M2AE (Zhang et al., 2022a)	92.9	84.1
DGCNN+Jigsaw (Sauder and Sievers, 2019)	90.6	59.5
DGCNN+STRL (Huang et al., 2021)	90.9	77.9
DGCNN+OcCo (Wang et al., 2021)	90.7	78.3
DGCNN+CrossPoint (Afham et al., 2022)	91.2	81.7
DGCNN+PointVST (Zhang and Hou, 2023)	92.1	-
DGCNN+Pic@Point (ours)	**92.5**	**85.7**
DGCNN+Pic@Point (ours, w/ normals)	**92.9**	-

Table 4.2: Linear SVM classification on ModelNet40 and ScanObjectNN (OBJ_BG). We compare against methods using a specialized point cloud architecture (*top*), and methods using a DGCNN backbone (*bottom*). We report the overall accuracy (%) with 1024 points.

Implementation Details The experiments are implemented in PyTorch using the OpenPoints framework (Qian et al., 2022a). We employ the Adam optimizer with Cosine Annealing and a weight decay of $1e^{-6}$ for pretraining. For the point cloud branch we use an initial learning rate of $1e^{-3}$, for the image branch we set a lower learning rate of $5e^{-5}$. We train with a batch size of 32. For the downstream tasks, we follow the training and evaluation settings of Qian et al. (2022a).

In the following, we present experimental results on object classification, part segmentation and scene segmentation.

Object Classification

To evaluate the effectiveness of the proposed Pic@Point pretraining method for object classification, we conduct experiments on the synthetic dataset ModelNet40 (Wu et al., 2015) and the real-world scanned dataset ScanObjectNN (Uy et al., 2019). ModelNet40 consists of 12 331 3D CAD models from 40 object categories. ScanObjectNN contains 15 categories of real-world indoor scans with 2903 unique object instances. We evaluate on all three common variants of the the ScanObjectNN dataset: OBJ_BG contains complete object scans, OBJ_ONLY uses background cropping, and PB_T50_RS contains perturbed

versions of the scans. We sample 1024 points on each object for both training and testing.

We use two transfer learning protocols to evaluate the effectiveness of our proposed Pic@Point model: linear probing with an SVM and fine-tuning on the downstream dataset.

Linear SVM Results We test the representation capability of Pic@Point by fitting a linear SVM on the features extracted from the pretrained point cloud backbone. In Table 4.2 we report our linear classification results on ModelNet40 and ScanObjectNN (OBJ_BG). The upper part of the table shows existing self-supervised methods employing specialized point cloud architectures, including transformer-style methods. These methods focus primarily on point cloud encoding architectures rather than on the design of pretext tasks. The lower part of the table shows architecture-agnostic pretraining methods, which are compared using a DGCNN backbone.

Pic@Point significantly outperforms competing methods on a DGCNN backbone with 92.5% and 85.7% linear classification accuracy on ModelNet40 and ScanObjectNN (OBJ_BG), respectively. By incorporating normals as additional input, we further improve to 92.9% accuracy on ModelNet40. We achieve margins of +0.4% and +4.0% over the second best methods PointVST (Zhang and Hou, 2023) and CrossPoint (Afham et al., 2022), respectively. This underscores the effectiveness of our proposed structural 2D–3D correspondence learning over existing cross-modal approaches such as PointVST, a generative method, and CrossPoint, a contrastive method that relies solely on global correspondences.

Notably, the improvements over existing methods are more pronounced on ScanObjectNN than on ModelNet40. This may be attributed to Pic@Point's enhanced generalization ability through learning modality-invariant features, making it more robust on real-world data with irregular sampling and noise. We outperform methods employing larger transformer-style architectures on ScanObjectNN, achieving a margin of +1.6% over the second best method Point-M2AE (Zhang et al., 2022a), which uses a large multi-scale transformer architecture.

Fine-tuning Results Next, we perform extensive fine-tuning experiments on all three variants of the ScanObjectNN dataset, which has established itself as a favored classification benchmark (Qian et al., 2022a; Ma et al., 2022). The

Method	#Params (M)	OBJ_BG	OBJ_ONLY	PB_T50_RS
PointNet (Qi et al., 2017a)	3.5	73.3	79.2	68.2
DGCNN (Wang et al., 2019)	1.8	82.8	86.2	78.1
PointNeXt-S (Qian et al., 2022a)	1.4	-	-	87.7
PointMLP (Ma et al., 2022)	12.6	-	-	85.4
Pix4Point (Qian et al., 2022b)	22.1	-	-	87.9
ACT (Dong et al., 2022)	22.1	93.3	91.9	88.2
P2P (Wang et al., 2022)	195.8	-	-	89.3
I2P-MAE (Zhang et al., 2023)	12.9	94.2	91.6	90.1
ReCon (Qi et al., 2023)	43.6	95.2	93.6	90.6
[T]ransformer (Vaswani et al., 2017)	22.1	79.9	80.6	77.2
[T]+OcCo (Yu et al., 2022)	22.1	84.9	85.5	78.8
Point-BERT (Yu et al., 2022)	22.1	87.4	88.1	83.1
Point-MAE (Pang et al., 2022)	22.1	90.0	88.3	85.2
Point-M2AE (Zhang et al., 2022a)	15.3	91.2	88.8	86.4
[T]+PointVST (Zhang and Hou, 2023)	22.1	-	-	86.6
[T]+TAP (Wang et al., 2023)	22.1	90.4	89.5	85.7
PointMLP+TAP (Wang et al., 2023)	12.6	-	-	88.5
PointNeXt-S (reproduce)	1.4	91.2	89.9	87.2
+ Pic@Point	1.4	**94.0** (+2.8)	**92.6** (+2.7)	**88.1** (+0.9)

Table 4.3: Real-world 3D classification on ScanObjectNN variants. We report the number of inference model parameters (M) and the overall accuracy (%) with 1024 points.

results are shown in Table 4.3. The top part of the table shows supervised 3D models trained from scratch. The middle part shows 2D-to-3D methods that use large-scale vision foundation models with specialized architectures and downstream adaptations such as visual prompt tuning or multi-stage knowledge distillation. We note that our model does not directly compete with these methods that are not architecture-agnostic and have large memory requirements and 2D-to-3D adaptations. Nevertheless, we include them to provide a comprehensive overview of related research. The lower part of the table lists self-supervised methods employing state-of-the-art 3D point cloud models, many of which are based on transformer architectures.

We perform experiments using the PointNeXt-S (Qian et al., 2022a) backbone. We observe significant improvements compared to training from scratch: +2.8% on OBJ_BG, +2.7% on OBJ_ONLY and +0.9% on PB_T50_RS. Pic@Point surpasses all self-supervised methods with transformer-style point cloud encoders on all three variants of ScanObjectNN, while having a factor of $1/10$th smaller model size. On PB_T50_RS, Pic@Point is outperformed only by TAP

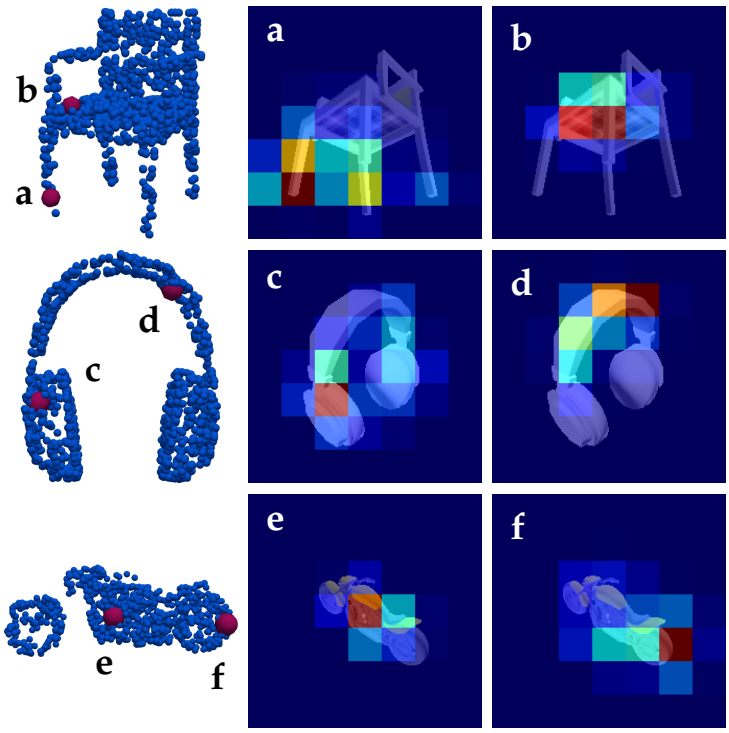


Figure 4.9: Visualization of local 2D–3D correspondences. The first column displays point clouds with two example points highlighted. The second and third columns display heatmaps of the feature distances between one of these points and the image patches of the corresponding object in latent space. Figure taken from Herzog and Suwelack, © 2025 PMLR.

(Wang et al., 2023) using a PointMLP backbone. On OBJ_BG and OBJ_ONLY, Pic@Point outperforms all competing 3D methods.

Visualization In Figure 4.9 we show visualizations of local 2D–3D correspondences learned during Pic@Point pretraining. For each object, the correspondences between the object's image patches and two exemplar points are depicted. Two key observations can be made: Firstly, Pic@Point successfully learns to correlate structures of point cloud regions with image patches depicting semantically similar features. Secondly, it accurately matches corresponding image features by incorporating pose information.

Ablation Studies Table 4.4 presents ablation studies on key components. First, we evaluate the local and global correspondence losses $\mathcal{L}_{\text{lcl}}$ and $\mathcal{L}_{\text{gbl}}$, both of which have equal impact on the results. Second, we confirm that the pose encoding aids in learning from local correspondences. A drop of -0.7% accuracy is observed when pose information is omitted. Lastly, while

$\mathcal{L}_{\text{lcl}}$	$\mathcal{L}_{\text{glb}}$	Pose Cond.	Normals	OA (%)
✗	✓	✓	✓	92.3 (−0.2)
✓	✗	✓	✓	92.3 (−0.2)
✓	✓	✗	✓	91.8 (−0.7)
✓	✓	✓	✓	92.8 (+0.3)
✓	✓	✓	✗	92.5

Table 4.4: Ablation studies on the contributions of local and global correspondence, the impact of pose conditioning, and the use of point normals, evaluated via linear SVM classification on ModelNet40 using a DGCNN backbone.

Method	ins. mIoU (%)	cls. mIoU (%)
PointNet (Qi et al., 2017a)	83.7	80.4
DGCNN (Wang et al., 2019)	85.2	82.3
PointNeXt-S (Qian et al., 2022a)	86.7	84.4
PointMLP (Ma et al., 2022)	86.1	84.6
Transformer (Vaswani et al., 2017)	85.1	83.4
Point-BERT (Yu et al., 2022)	85.6	84.1
Point-MAE (Pang et al., 2022)	86.1	84.2
DGCNN+Jigsaw (Sauder and Sievers, 2019)	85.3	82.3
DGCNN+OcCo (Wang et al., 2021)	85.0	-
DGCNN+CrossPoint (Afham et al., 2022)	85.5	-
DGCNN+PointVST (Zhang and Hou, 2023)	**87.4**	-
DGCNN+Pic@Point (ours)	85.8	**83.0**

Table 4.5: Part segmentation on ShapeNetPart. We report the mean IoU across all instances (ins.) and across all classes (cls.) with 2048 points.

our standard Pic@Point model excludes normals to ensure fair comparison with related methods, we show that incorporating normal information during pretraining and downstream fine-tuning, when available, provides significant benefits.

Part Segmentation

ShapeNetPart (Yi et al., 2016) is a widely used dataset for object part segmentation. It contains 16 881 pre-aligned CAD models from 16 object classes and has a total of 50 part categories. We use the same pretraining setup as for classification, pretraining on 1024 randomly sampled points per object on the ShapeNet dataset. For downstream fine-tuning on ShapeNetPart, we train and test on 2048 points. Table 4.5 reports the mean intersection over union (mIoU)

Method	mIoU (%)	mAcc (%)
PointNet (Qi et al., 2017a)	41.1	49.0
DGCNN (Wang et al., 2019)	47.9	-
PointNeXt-S (Qian et al., 2022a)	63.4	-
PointNeXt-XL (Qian et al., 2022a)	70.5	-
Transformer (Vaswani et al., 2017)	60.0	68.6
Point-BERT (Yu et al., 2022)	60.8	69.9
Point-MAE (Pang et al., 2022)	60.8	69.9
PointNeXt-S (reproduce)	62.9	69.5
+ Pic@Point	**63.6** (+0.7)	**71.1** (+0.6)

Table 4.6: Scene-level semantic segmentation on S3DIS Area 5. We report mean IoU and mean accuracy.

averaged over all instances (ins.) and averaged over all object classes (cls.) with DGCNN backbone.

Overall, ShapeNetPart results are less distinguishable compared to other benchmarks. Both Jigsaw (Sauder and Sievers, 2019) and OcCo (Wang et al., 2021) show no significant improvement over training from scratch using DGCNN. Pic@Point demonstrates slightly better performance than previous pretraining methods on a DGCNN backbone, except for PointVST (Zhang and Hou, 2023), which reports an exceptional result of 87.4%. Compared to CrossPoint (Afham et al., 2022), which uses global 2D correspondences, Pic@Point shows an improvement of +0.3%.

Scene Segmentation

Semantic segmentation on large 3D scenes challenges the understanding of contextual relationships and coherent semantic interpretation. S3DIS (Armeni et al., 2016) is a scene segmentation benchmark consisting of 6 types of large scanned indoor areas with 13 semantic categories. Following common practice, we test on the largest area, Area 5, and fine-tune on the remaining areas. For training, the point clouds are downsampled with a voxel size of 0.04m and sub-sampled to 24 000 points. Testing is conducted on the entire scene.

Pic@Point consistently improves performance over training from scratch, increasing mIoU by +0.7% and mAcc by +0.6%. It outperforms leading transformer-style methods by a margin of +2.8% mIoU. The achieved scene segmentation results indicate that Pic@Point is superior in performing dense

Pic@Point+	#Params M	#Trainable Params M	FLOPs G	Time Per Epoch [mm:ss]
PointNeXt-S	14.3	3.1	5.4	01:12
PointNeXt-S global	13.2	2.0	5.3	00:58
PointNeXt-S local	13.2	2.1	5.4	01:09
DGCNN	14.1	2.9	9.4	13:16
DGCNN global	13.1	1.9	8.5	01:48
DGCNN local	12.8	1.6	9.4	13:09

Table 4.7: Memory and time costs of Pic@Point pretraining. We report the number of (trainable) parameters, floating-point operations, and training time per epoch for pretraining on ShapeNet with a batch size of 32 and 1024 points. We measure on an NVIDIA GeForce RTX 4090 24 GB GPU and a 24-core Intel i9-14900K.

prediction tasks through its integration of rich semantic cues and structural, pose-aware guidance.

Memory Consumption and Training Time

Table 4.7 shows the memory and time costs associated with Pic@Point pretraining. Note that these measurements apply only to the pretraining phase. There is no memory or time overhead during downstream tasks; these depend solely on the 3D backbone model. The differences in training time between Pic@Point pretraining with the PointNeXt-S and DGCNN backbones are mainly attributed to two factors:

1. Processing speed: PointNeXt-S has significantly higher processing speed than DGCNN. Its benchmarks show approximately x5 greater throughput on datasets like ScanObjectNN and ModelNet.
2. Hierarchical aggregation: DGCNN does not incorporate hierarchical aggregation, resulting in a larger number of top-level local features and, consequently, more local comparisons. This causes the computation of local correspondences to take up a larger portion of the total processing time for DGCNN compared to PointNeXt-S.

4.5 Summary

This chapter explored approaches for leveraging CAD model information in the design of pretext tasks, categorized into: (1) native CAD format infor-

mation, (2) design traits, and (3) multimodal data. Due to the limitations of publicly available parametric CAD datasets, the emphasis was placed on leveraging design traits and multimodal data.

Building on the principles of geometric deep learning, we presented an approach that exploits regularities in CAD designs to enhance representation learning. Specifically, we leverage the natural axis alignment of CAD models to encourage invariance to 90° rotations, formulated as a pretext task. This task helps the model learn orientation-robust, discriminative shape representations from unlabeled data, and can be applied using different backbone architectures without modification.

We introduced a novel cross-modal pretraining method that uses unlabeled point cloud and rendering data extracted from CAD models. This method extracts rich structural cues through contrastive learning of point cloud–image correspondences at different levels of shape abstraction. It improves on existing cross-modal methods by providing a rich guiding signal at both global and local scales. Experiments showed that our lightweight Pic@Point model outperforms state-of-the-art self-supervised methods on several 3D benchmark datasets.

CHAPTER 5

Content-based Shape Retrieval for Design Reuse

Reusing existing data is a key strategy for accelerating product development and reducing overall production costs. A central aspect of design reuse is the retrieval of similar CAD models, which are at the heart of the product design and development process. Reusing in-house or purchased standard components facilitates standardization and data integrity throughout the product life cycle. In addition, finding part replacements and alternatives can mitigate supply chain risks and disruptions.

The de facto standard search functionality in product data management and product lifecycle management systems is text search. However, text search is inherently dependent on the quality of the model-related data. In practice, part names, tags, and metadata (e.g., descriptions, dimensions, supplier information) are often incomplete or inconsistent. In contrast, content-based retrieval solutions work directly on the model shape itself.

This chapter explores shape retrieval as a practical application of vector-based representations for CAD models. The retrieval system presented here, as depicted in Figure 5.1, takes a user-centered approach to design-oriented shape retrieval. Users can tailor the search process by defining ROIs on the query shape and adjusting their relevance for similarity comparison to align with functional design requirements. Efficient indexing structures on hierarchical vector-based shape signatures support interactive exploration of retrieved shapes.

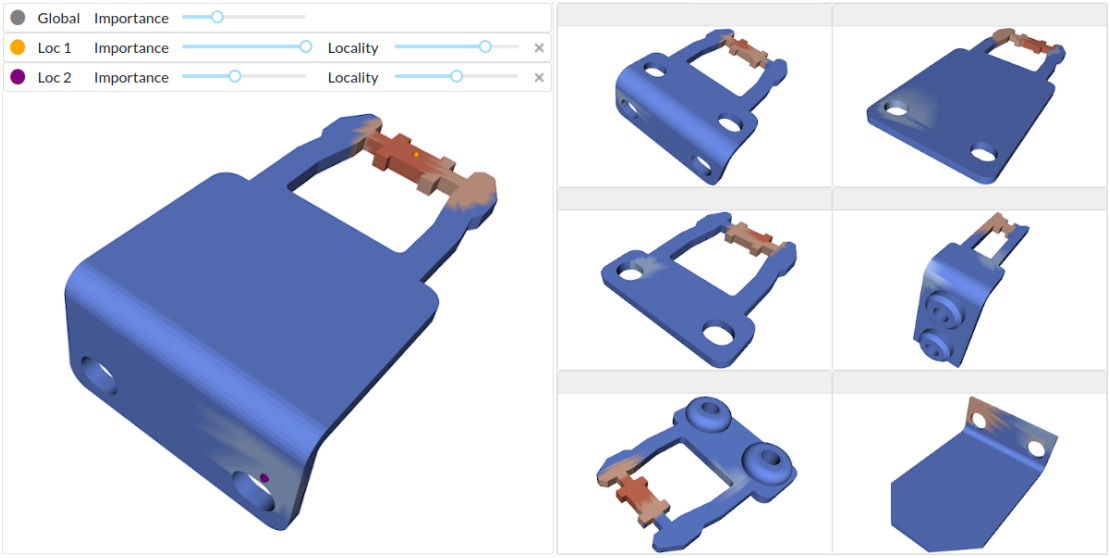


Figure 5.1: User-guided search for safety switches with screw mounting in a collection of diverse mechanical components. The user interactively selects ROIs and adjusts their weighting in similarity comparison. The coloring indicates how strongly a region is considered in similarity matching. Figure taken from Herzog and Suwelack, © 2023 Elsevier.

The chapter is structured as follows: Section 5.1 discusses partial shape retrieval and user guidance mechanisms. Section 5.2 presents hierarchical shape signatures that combine learning-based and traditional shape descriptors. Section 5.3 outlines indexing structures for similarity search and presents the overall search process.

This chapter is based on Herzog and Suwelack (2023).

5.1 Partial Shape Retrieval with User Guidance

Shape retrieval is the process of finding 3D models in a database that are similar to a given query model. Unlike global retrieval, which compares entire models, partial retrieval identifies models based on the similarity of subparts, even if their overall shapes differ. This is relevant when searching with incomplete, occluded, or modified query shapes. In engineering applications, partial retrieval allows to locate models that share certain functional features.

5.1.1 Partial Shape Retrieval Methods

In partial shape retrieval, *part-in-whole matching* methods identify stored models that fully contain the query model. This task involves finding larger model

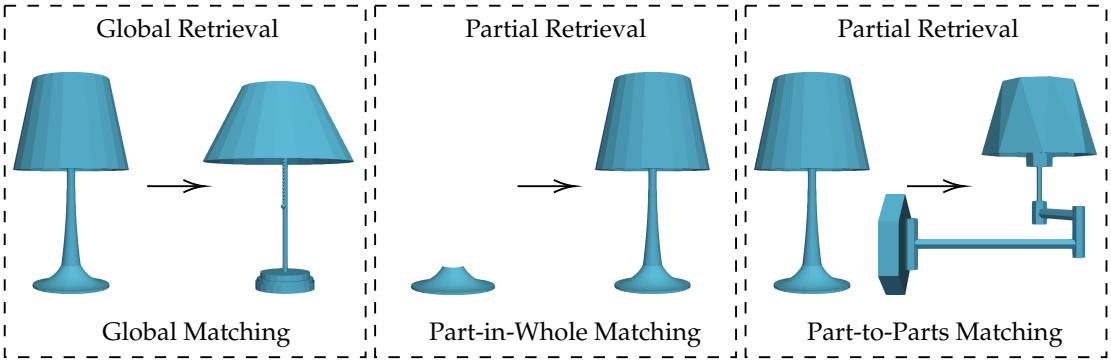


Figure 5.2: Illustration of global and partial shape retrieval methods. The query object is shown on the left and a matched retrieval object is shown on the right.

compositions that contain a particular subpart, as illustrated in Figure 5.2. In this work, we address the more general *part-to-parts matching* problem, which seeks to identify stored models with subparts that resemble some subparts of the query model. This task involves searching for models that share local similarities.

One of the main challenges in partial shape retrieval is the lack of prior knowledge about the position, scale, and orientation of the partial query within a stored model. Exhaustive computation of all possible configurations is computationally prohibitive. Partial retrieval methods are designed to describe 3D models both efficiently and meaningfully.

Segmentation-grouping methods, such as those proposed by Bespalov et al. (2006) and Tao et al. (2013), decompose 3D shapes into subparts using region growing or clustering techniques to divide the model into convex, concave, and planar regions. These subpart decompositions are represented as graphs and are compared using graph matching techniques such as subgraph isomorphism. While segmentation-grouping methods align well with human perception, they are sensitive to topological variations and threshold parameters, and are typically suited to relatively simple shapes (Liu et al., 2013).

Local shape descriptor-based methods, discussed in subsubsection 2.1.3, extract geometric features from local neighborhoods and assess partial similarity by comparing sets of descriptors. These methods provide more flexibility in dealing with complex models and partial shapes. They produce sparse, vector-based descriptors that facilitate similarity measurements using metrics such as Euclidean distance.

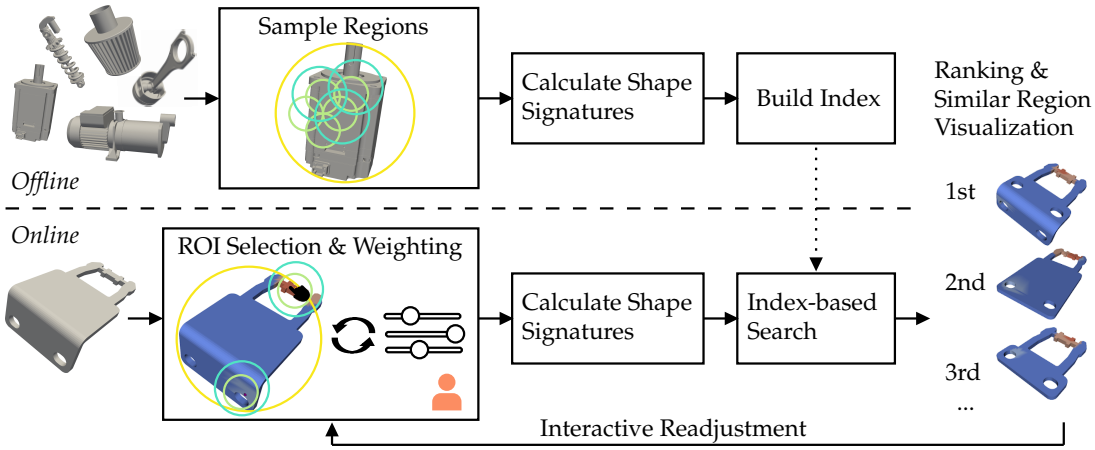


Figure 5.3: Overview of the proposed retrieval process. Figure adapted from Herzog and Suwelack, © 2023 Elsevier.

5.1.2 User Guidance and Feedback Mechanisms

A key challenge of content-based search in CAD data is the ambiguous notion of design similarity, which depends on user intent and application context. For example, one user may be looking for similar pulleys with the same tooth type and shape (e.g., trapezoidal, curved, tire track, flat), while another user may need pulleys with the same type of mounting (e.g., set screw, keyway, split hub, quick-release bushing, taper lock).

This *semantic gap* between geometric information and user interpretation, as described by Smeulders et al. (2000), is a major barrier to the success of content-based retrieval solutions. User guidance through query specification and system interaction can effectively narrow this gap. This is especially relevant when searching in large heterogeneous shape collections, where many globally similar shapes can be found and further specification of relevant features is necessary to obtain appropriate recommendations.

Different mechanisms can be used to implement user guidance. Relevance feedback mechanisms (Elad et al., 2002; Leifman et al., 2005; Giorgi et al., 2010) iteratively take user feedback on the displayed retrieval results and adjust the feature space in an attempt to reduce the gap between geometric information and user interpretation. Shape correspondence methods (Kim et al., 2012; Rustamov et al., 2013; Huang et al., 2014) allow user-defined ROIs for query specification, providing more direct control and supporting interactive exploration of shape collections. These methods calculate shape map correspondences to analyze fine variations within single-class collections.

In our retrieval system, we employ user-defined ROIs to search in large heterogeneous shape collections. For each ROI, users can define a resolution level and assign a relevance weighting in the overall similarity score. The regions are matched to corresponding regions in the stored models by computing the distance between vector-based shape descriptors of the same scale level. The retrieval system visualizes the matched regions in the retrieved models, allowing users to interactively adjust their search based on the retrieved results. Figure 5.3 provides an overview of the retrieval system, detailing both the offline computations and the online search process.

5.2 Hierarchical Shape Signatures

We define shape signatures at multiple scales by combining self-supervised learning-based representations with traditional shape descriptors. At the global shape level, we densely encode nuanced semantic and geometric relationships of the objects using self-supervised learning-based representations. To describe local shape regions, we are less concerned with high-level semantics and instead focus on accurately capturing local features. To achieve this, we use traditional local shape descriptors that are easy to extract and require no training.

We capture each shape at three resolution levels: the global shape level g and the local shape levels l_2 and l_1. The global shape level g captures the entire shape, while l_2 and l_1 correspond to local regions sampled as spheres with radii $r = 0.4$ and $r = 0.2$, respectively. During the offline phase, we uniformly sample 128 regions on l_2 and 512 regions on l_1 for each stored model. All shapes are normalized to fit into a unit cube $[-1, 1]^3 \subset \mathbb{R}^3$. Shape descriptors are standardized to zero mean and unit variance.

5.2.1 Global Shape Descriptor

Self-supervised learning uses deep neural networks to learn discriminative, dense vector-based shape representations without the need for manual labels during training. Guided by the geometric representation learning principles of Bronstein et al. (2021), we designed a multi-task neural network with three pretext tasks focused on exploiting geometric priors of transformation invariance and hierarchical structure:

Self-Reconstruction In self-reconstruction, the input shape is compressed into a feature vector and subsequently reconstructed using a decoder model. Following Rao et al. (2020), we adopt a FoldingNet (Yang et al., 2018) decoder, which learns a folding operation from a canonical 2D grid onto the underlying 3D surface of a point cloud. The reconstruction loss is measured via Chamfer distance between an input point cloud P and its reconstruction P': $\text{CD}(P, P')$, as defined in Equation 2.3.

90° Rotation Invariance We further enforce rotation-invariant shape representations through a metric learning objective. The model learns a nonlinear embedding space, where the distance between a sample object and its rotation-augmented version is optimized to be close, while the distances to other objects are maximized. As mechanical components are commonly axis-aligned, we augment with 90° rotations. The design of this pretext task is detailed in subsection 4.2.1.

Hierarchical Reasoning In addition to the aforementioned tasks, we employ the local-to-global reasoning task by Rao et al. (2020), which aims at recognizing global shapes by seeing only a small part. Similar to the rotation invariance task, this pretext task is formulated as a metric learning objective, where local shape features and the corresponding global shape representation are projected into a shared feature space, where they are pulled close. The design of this pretext task is discussed in subsubsection 3.2.2.

We employ a PointNet++ backbone (Qi et al., 2017a) that processes point clouds with pointwise surface normals, comprising 2048 points sampled from CAD models. Shape representations are extracted from the global abstraction level of the neural network, resulting in a 512-dimensional feature vector.

5.2.2 Local Shape Descriptor

We describe local regions using the Point Feature Histogram (PFH) descriptor proposed by Rusu et al. (2008). PFH is a pose-invariant descriptor for point clouds with pointwise surface normals. It captures fine details while being largely noise resistant. PFH encodes the neighborhood of each point as triplets (α, β, γ) of angular values between all pairs of point normals. For each pair of points p_s and p_t with normals n_s and n_t in the neighborhood, a local Darboux

frame is defined with origin in p_s: $u = n_s$, $v = u \times (p_t - p_s)/\|p_t - p_s\|_2$, and $w = u \times v$. The angular values are then defined as:

$$\begin{aligned} \alpha &= v \cdot n_t, \\ \beta &= u \cdot \frac{(p_t - p_s)}{\|p_t - p_s\|_2}, \\ \gamma &= \operatorname{atan2}(w \cdot n_t, u \cdot n_t). \end{aligned} \tag{5.1}$$

This definition follows the Point Cloud Library (PCL) implementation of PFH. While the original proposal by Rusu et al. (2008) uses the distance between p_s and p_t as a fourth value, the PCL implementation does not consider this feature discriminative enough and uses only the three angular values.

All sets of triplets are jointly binned into a three-dimensional histogram with 5 bins per angle, resulting in a $5 \times 5 \times 5 = 125$-dimensional feature vector. The point clouds are obtained by sampling 10 000 points with surface normals from CAD models.

5.3 Index-based Similarity Search

Index-based similarity search in vector embeddings has gained significant momentum due to large-scale applications in multimedia search and recommender systems. Similarity search libraries such as *Faiss* (Douze et al., 2024) are able to perform approximate k-Nearest Neighbor (k-NN) search in millions to billions of high-dimensional vectors in milliseconds to seconds (Johnson et al., 2019; Ren et al., 2020). This progress has led us to adopt a purely vector-based approach to scalable partial shape matching.

5.3.1 Indexing Structures for k-NN Search

Given a query vector, k-NN search finds the k nearest elements in a data pool. To avoid the cost of brute force search, which iterates over all stored vectors at search time, indexing structures can be built on top of the database, reducing memory usage and accelerating search time through pre-computation and approximation.

We place the signatures of all data pool shapes into index-based datastores per resolution level. The number of vectors stored per resolution level is given by the number of pool objects times the number of sampled regions. For each

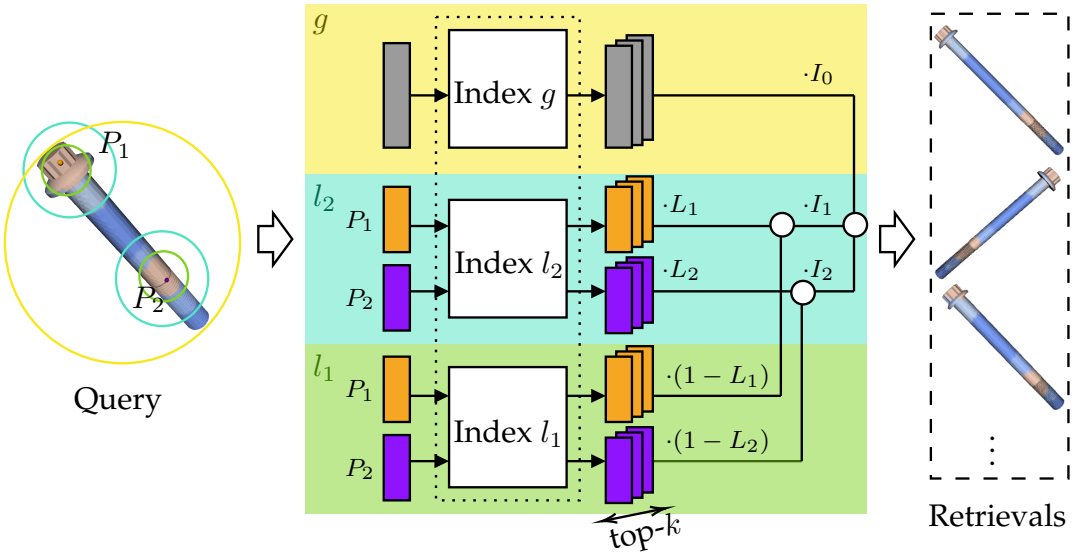


Figure 5.4: Illustration of the index-based search for a partially threaded 12-point screw. Figure adapted from Herzog and Suwelack, © 2023 Elsevier.

store, we build efficient indexing structures using Product Quantization (PQ) and Inverted File Index (IVF) techniques implemented in Faiss.

The product quantizer reduces the storage space by compressing the stored vectors via a codebook approach. Each stored vector of dimension d is split into m sub-vectors of dimension $d^* = d/m$. k-means clustering is performed separately on each sub-vector to decompose each sub-space into $k^* = 2^{n_{\text{bits}}}$ cells. The cell centroids serve as prototypes for the possible values of the sub-vectors and are stored explicitly. The floating-point values of the sub-vectors are replaced by the index of the closest cell, thus reducing the memory footprint.

The product quantizer is combined with IVF, a non-exhaustive search technique that reduces the search scope by clustering the search space at indexing time. It uses k-means clustering to partition the space into n_{list} cells. The vectors in each cell are stored contiguously in an inverted file. At search time, the query vector is compared against the cell centroids and only the n_{probe} nearest cells are visited.

5.3.2 Search Process

The following describes the process of searching for similar objects using the previously defined index-based datastores. Figure 5.4 shows an example where a query object containing two custom ROIs is used for the search. Given a query object with ROIs $i = 1, \ldots, m$, where $i = 0$ denotes the global shape

region, the user specifies each ROI by selecting a point on the query object and adjusting two sliders that control the scale and weighting of that region. These adjustments are controlled by the following parameters:

- importance factor $I_i \in [0,1], i \geq 0$ defines the relative importance of region i,
- locality factor $L_i \in [0,1], i \geq 1$ defines the scale of region i.

The subsequent top-k retrieval process consists of three steps:

1. For every ROI, extract resolution level-wise shape descriptors.
2. For every extracted descriptor, retrieve the top-$\bar{k}$ nearest pool feature vectors.
3. Combine the retrievals into a joined retrieval list.

For resolution level $j \neq g$, there is no guarantee that the top-k retrieved feature vectors belong to different objects. In fact, due to overlapping sampled regions, the most similar feature vectors typically belong to different regions of the same few objects. We adopt a simple solution to filter out duplicate retrieval objects. Let n_j denote the number of sampled regions at resolution level j. Then we calculate the $\bar{k} = n_j \cdot k$ nearest neighbors and keep the most similar feature vectors of the top-k unique objects. Note that $\bar{k}$ depends only on the granularity of the shape resolutions; it is constant in the number of pool objects.

The joined retrieval score s_h of a retrieved object h is calculated as

$$s_h = I_0 \cdot d_{0,g} + \sum_{i=1}^{m} I_i \cdot (L_i \cdot d_{i,l_2} + (1 - L_i) \cdot d_{i,l_1}), \tag{5.2}$$

where $d_{i,j}$ denotes the retrieved distance of region i at resolution level j. The score of ROI i is defined as a linear interpolation of distances at local resolution levels l_1, l_2 with interpolation factor L_i. The scores of all regions are combined into a weighted sum with importance factors I_i. Missing distance values, i.e. an object is very similar on some resolution level but not in the top-$\bar{k}$ on others, are set to the upper bound of retrieved distances. The top-k retrieval list is ordered by ascending joined retrieval score.

5.4 Experiments

5.4.1 Experimental Setup

Datasets We evaluate the performance of our interactive retrieval system on three datasets of mechanical components: the ESB dataset (Jayanti et al., 2006), the DMUNet (Dekhtiar et al., 2018), and the ABC dataset (Koch et al., 2019a). Existing retrieval benchmarks mainly cater to the evaluation of global retrieval methods, where similarity between objects is defined as belonging to the same class. For a partial, multi-faceted evaluation of design similarity, we additionally define fine-granular retrieval tasks based on the similarity of local functional features on subsets of the DMUNet and the ABC dataset.

Interactive Queries The evaluation setup for interactive queries is as follows. For each query object, a point of interest is manually selected and up to three adjustments to the ROI settings (I, L) are interactively made based on the retrieval results. This amounts to up to $1 + 3 = 4$ mouse clicks. Due to the manual effort involved in evaluating an interactive system, all queries are performed by a single user. A comparison to multiple first-time users is conducted in subsubsection 5.4.3.

Evaluation Metrics We evaluate each query object by using the remaining objects to form the retrieval pool. Since our proposed system is designed to be interactive and exploratory, we put a strong emphasis on the relevance of the top few retrieved items. These are the ones that users look at and interact with.

We use the following metrics with micro averaging:

- Precision@k (P@k): the ratio of relevant items in the top k retrieved.
- Nearest Neighbor (NN): the ratio of closest relevant matches (NN = P@1).

Index Parameters On datastores with $\geq$ 1M vectors we use IVF($n_{list} = 1024$), PQ($m = 64$, $n_{bits} = 8$) and $n_{probe} = 128$ to reduce search time and storage space. On datastores with $<$ 1M vectors we use flat indexes. The largest number of vectors stored in a single datastore in our experiments belongs to the l_1 index on the largest used ABC subset, which amounts to $10\,150 \times 512 \approx 5$ million vectors.

Method	NN
TLC* (Bai et al., 2015)	88.2
MFSD (Tatsuma and Aono, 2009)	87.5
densenet-AE* (Bickel et al., 2023)	87.4
PANORAMA* (Papadakis et al., 2010)	86.5
DESIRE (Vranic, 2005)	86.0
LFD (Chen et al., 2003)	85.5
SHD (Kazhdan et al., 2003)	83.5
ours (global descriptor)	87.6

Table 5.1: Performance of our global descriptor in comparison to global retrieval methods on ESB. (*) marked methods use deviated ESB subsets. The results for SHD, LFD and DESIRE are stated as reported by Tatsuma and Aono (2009).

Training Parameters The self-supervised neural network for the global shape descriptor is trained for 300 epochs using the Adam optimizer with a base learning rate of $1e^{-3}$ and a batch size of 32. It is trained on a 50% training split of the DMUNet dataset, in addition to 800 ESB samples for the ESB dataset and 600 screws for the ABC+Screw dataset. Note that since the ESB does not provide a separate test set, the network is trained on the data pool, while for the other datasets, the training samples are separate from the query and pool samples.

5.4.2 Results on ESB

The ESB dataset is a well established benchmark dataset for the retrieval of mechanical components. Following Tatsuma and Aono (2009), we consider 800 samples from 42 classes. First, we compare the performance of our proposed retrieval system using only the global signature to well-known global retrieval methods. After that, we measure the improvement by incorporating user-defined ROIs on several classes.

Performance with Global Signature Only

Table 5.1 shows the nearest neighbor retrieval performance with our global signature only in comparison to well-known global retrieval methods on ESB. Our global signature yields results comparable to established methods, with only a slight performance gap observed for TLC. In further experiments,

Class	
Handles	18
Contact switches	8
Gear-like parts	36
Spoked wheels	15

Table 5.2: Categories of query models on ESB for interactive partial retrieval evaluation.

retrieval based on the global signature serves as a baseline to measure the gain by incorporating user-defined ROIs.

Performance with User-defined ROIs

Table 5.3 shows the improvement by incorporating user-defined regions in comparison to retrieval based on the global shape signature only. Figure 5.5 shows exemplary retrieval results. We carried out interactive queries for each object of four classes, as listed in Table 5.2. The selection of classes is based on the following criteria:

1. the class represents a functional (e.g., handles, contact switches) rather than a shape-based (e.g., 'U-shaped parts') category,
2. the class is well-defined, as in being loosely consistent with the crowdsourcing categorization performed by Jagadeesan et al. (2009).

Overall, a significant improvement can be observed by enabling the user to select a ROI. The performance gain compared to global retrieval varies between classes, with a 13.8% P@5 error reduction for the spoked wheels up to a 78.3% P@5 error reduction for the contact switches. For the gear-like parts the ROI was placed at the gear teeth. In Figure 5.5 it can be observed that this feature is also present in the spoked wheels class, resulting in some retrieval failures.

Comparison to APC Part-in-Whole Retrieval

To further test our local similarity measure, we draw comparison to part-in-whole retrieval via autoencoder-based part clustering (APC) by Muraleedharan et al. (2019), shown in Figure 5.6. We compare with part-in-whole queries by manually selecting a ROI corresponding to the query part and setting the importance weighting to only the local signature ($I_0 = 0, I_1 = 1$). In alignment with the APC setup, we allow the query object to be part of the retrieval data pool.

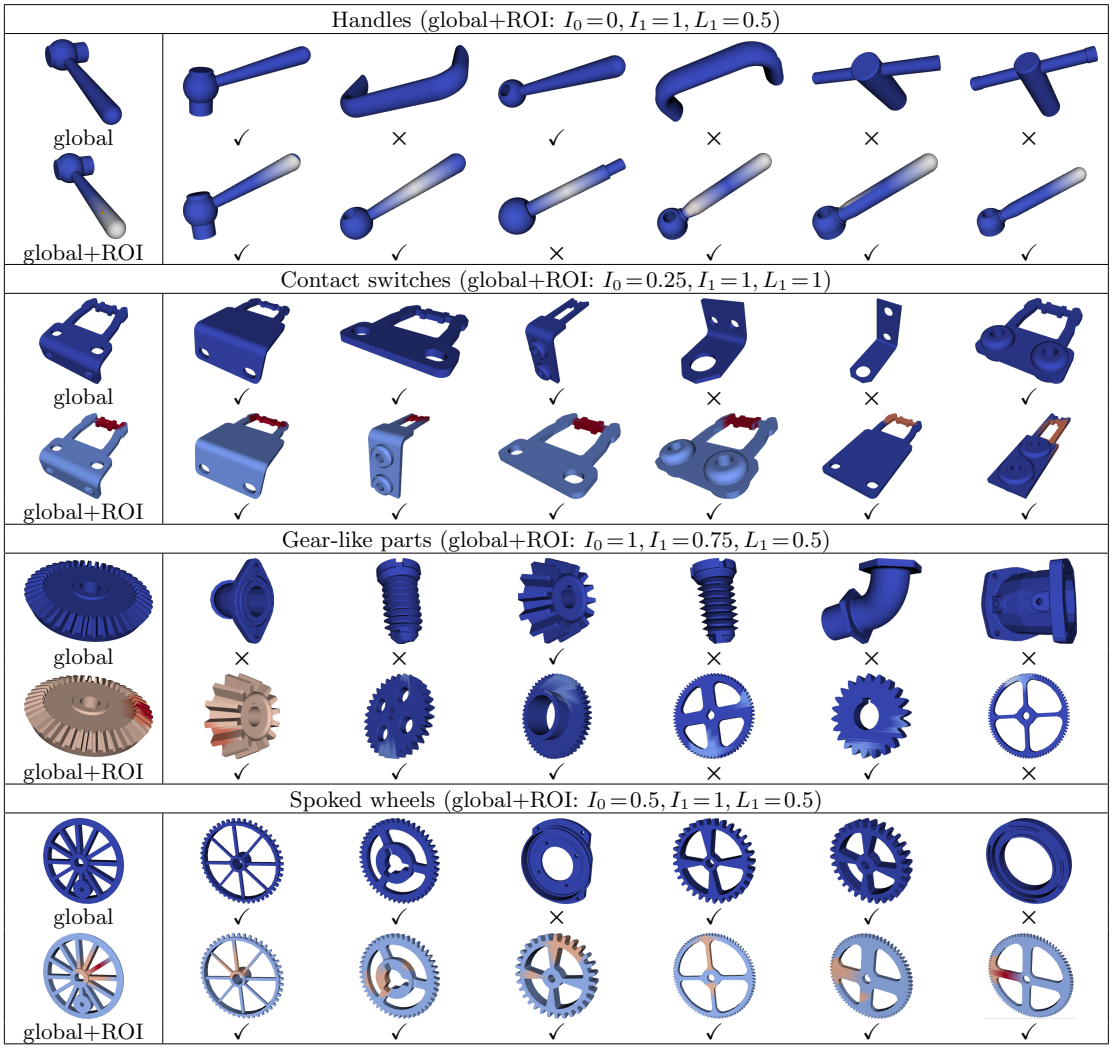


Figure 5.5: Exemplary retrievals on ESB. For user-defined ROIs, the importance-locality settings (I, L) are indicated. The coloring ranges from cool to warm, indicating low to high ROI importance. Figure taken from Herzog and Suwelack, © 2023 Elsevier.

Class	Signature	NN	P@3	P@5
Handles	global	94.4	92.6	87.8
	global+ROI	**96.3**	**95.7**	**91.5**
Contact switches	global	75.0	54.2	42.5
	global+ROI	**100.0**	**95.8**	**87.5**
Gear-like parts	global	91.7	88.9	86.1
	global+ROI	**93.5**	**90.7**	**89.3**
Spoked wheels	global	80.0	66.7	58.7
	global+ROI	**86.7**	**80.0**	**64.4**

Table 5.3: Retrieval performance with user-defined ROIs (global+ROI) in comparison to pure global retrieval (global) on ESB.

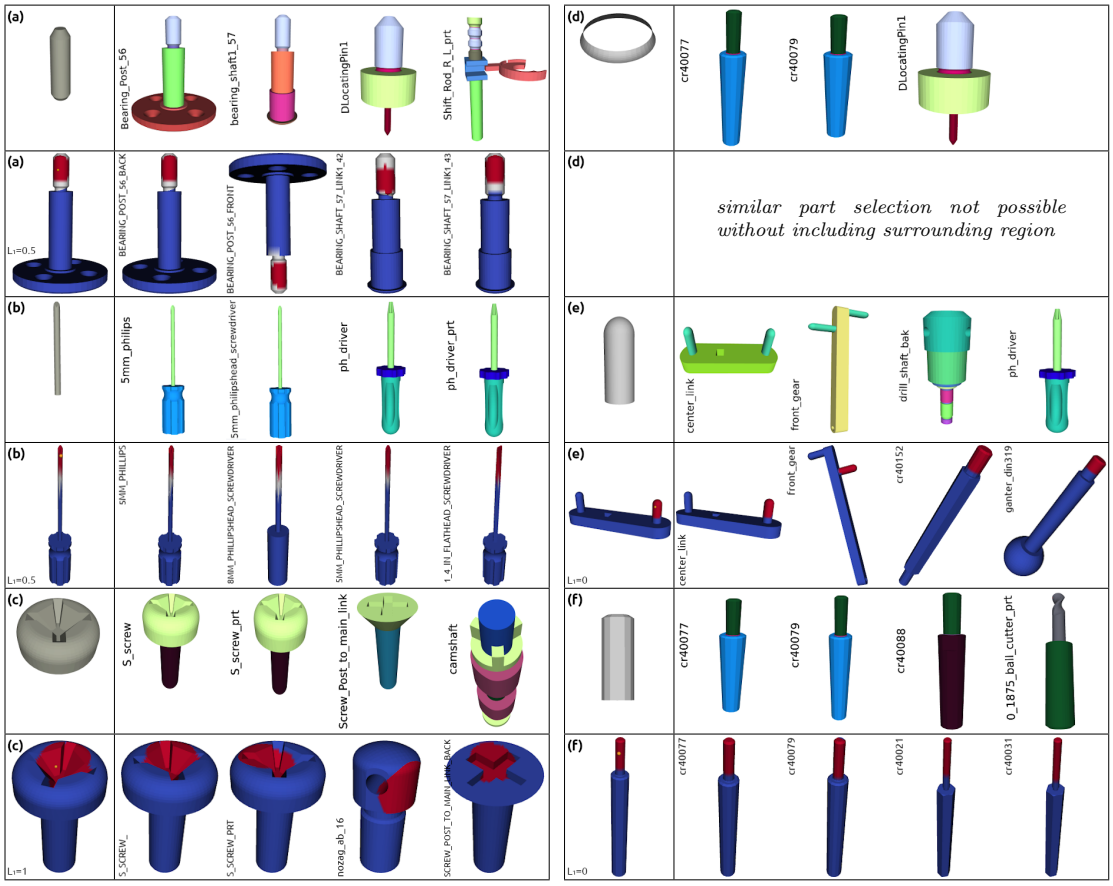


Figure 5.6: Comparison of local part retrieval based on our user-defined ROIs only (second row of query a-f) to APC part-in-whole retrieval (first row of query a-f). Figure taken from Herzog and Suwelack, © 2023 Elsevier.

The first retrieved object consistently corresponds to the same whole model containing the query region. For query (a), our system using only the local region signature correctly retrieves other bearing shaft pins, including on models with other orientation. APC retrieves on rank three and four pin-shaped parts of different model types. For query (c), we retrieve Phillips round head screws like the query part on the first two ranks. In third place, we retrieve a different model type with a rounded head, as the rounded shape appears to prevail over the cross-shaped indentation. APC on the other hand retrieves a Phillips flat head screw. Query (d) cannot be selected similarly to the query segment of APC without including the surrounding region via our sphere-shaped ROI selection. For query part (f), a long 14-sided prism, we correctly retrieve all visually similar parts, all from the same model type. APC matches on rank four a cylindrical part with a notch instead.

5.4.3 Results on DMUNet

In addition to the class-based retrieval evaluation on ESB, we construct fine-granular similarity tasks on the DMUNet. We search for objects that are not only of the same overall class, but also possess specific design features, as detailed in Table 5.4. Following the original DMUNet implementation by Dekhtiar et al. (2018), the data pool consists of a 50% test split with 958 mechanical components from 30 classes.

We define the following tasks:

1. retrieving pulleys with a keyway,
2. retrieving shockabsorbers with a gas canister,
3. retrieving switches based on their connection type (solder eye / solder pin).

Table 5.5 shows the performance of our proposed system. Figure 5.7 shows exemplary retrieval results. The improvement with user-defined ROIs compared to pure global retrieval ranges from a P@5 error reduction of 17.4% for shockabsorbers with gas canister to 32.8% for pulleys with keyway. Each task presents a unique challenge. Locating similar pulleys with a keyway is challenging due to the small scale of the keyway feature, which requires highly detailed shape signatures. Shockabsorbers have a distinct global shape, yet the gas canister can vary considerably in shape. Lastly, the switches vary significantly in global shape and soldering features are present in many other components.

Class		Feature			
Pulleys	53	with keyway	18		
Shockabsorbers	38	with gas canister	10		
Switches	38	with solder eye	14	with solder pin	6

Table 5.4: Categories of query models on DMUNet for interactive partial retrieval evaluation.

Class	Signature	NN	P@3	P@5
Pulleys with keyway	global	33.3	40.7	35.6
	global+ROI	**74.1**	**67.3**	**56.7**
Shockabsorbers with gas canister	global	80.0	63.3	50.0
	global+ROI	**100.0**	**76.7**	**58.7**
Switches with solder eye/pin	global	50.0	41.7	33.0
	global+ROI	**85.0**	**68.3**	**51.3**

Table 5.5: Retrieval performance on fine-grained DMUNet tasks.

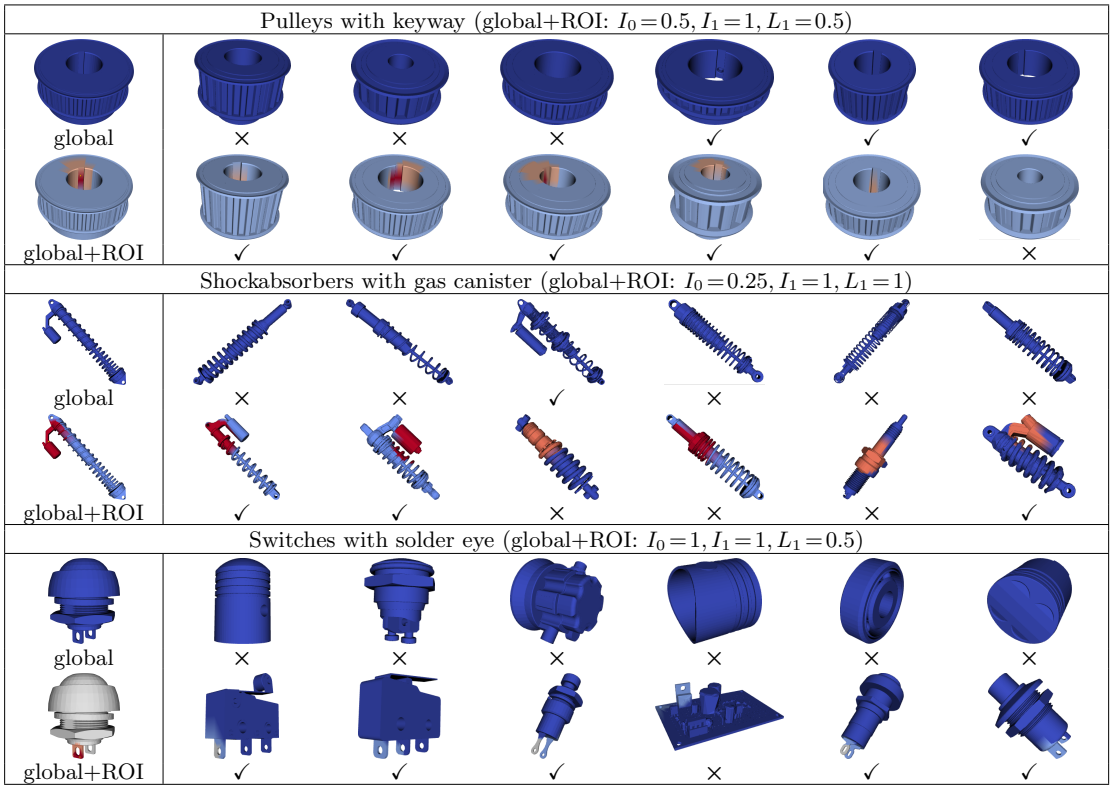


Figure 5.7: Exemplary fine-grained retrievals on the DMUNet. Figure taken from Herzog and Suwelack, © 2023 Elsevier.

Class	$\mathcal{L}_{\text{rec}}$	$\mathcal{L}_{\text{rot}}$	$\mathcal{L}_{\text{l2g}}$	NN	P@3	P@5
Pulleys with keyway	✓	✓		**74.1**	57.4	47.8
	✓		✓	**74.1**	62.3	53.3
		✓	✓	72.2	63.0	**57.0**
	✓	✓	✓	**74.1**	**67.3**	56.7
Shockabsorbers with gas canister	✓	✓		66.7	57.8	45.3
	✓		✓	93.3	66.7	53.3
		✓	✓	83.3	62.2	46.7
	✓	✓	✓	**100.0**	**76.7**	**58.7**
Switches with solder eye/pin	✓	✓		78.3	51.7	40.3
	✓		✓	78.3	56.7	47.7
		✓	✓	80.0	60.6	44.3
	✓	✓	✓	**85.0**	**68.3**	**51.3**

Table 5.6: Ablation study for the pretext tasks of the global descriptor on DMUNet.

Ablation Study

To examine the effectiveness of the self-supervision tasks of the global shape descriptor, we perform an ablation study in Table 5.6. It can be observed that each self-supervision loss contributes to the retrieval performance and by removing it from the system the precision rate decreases. For the pulleys with keyway task, the change in performance is less significant than for the other tasks. This can be explained by the overall lower global signature performance impact for this task, as indicated by the global vs. global+ROI performance in Table 5.5.

First-Time User Study

For an interactive system the quality of search results can vary between individual users. To test the intuitive usability and measure the variations between different human subjects who operate the system, we conducted the same fine-grained DMUNet retrieval tasks with 5 first-time users. The results of the user study are plotted in Figure 5.8. The study participants have been given two non task-related example geometries up front as an introduction to the graphical user interface and have used the same interactive query setup.

For the shockabsorber and switch retrieval tasks, the performance between first-time users is consistent, and the non first-time reference user performing the interactive queries for the global+ROI experiments performs in the upper

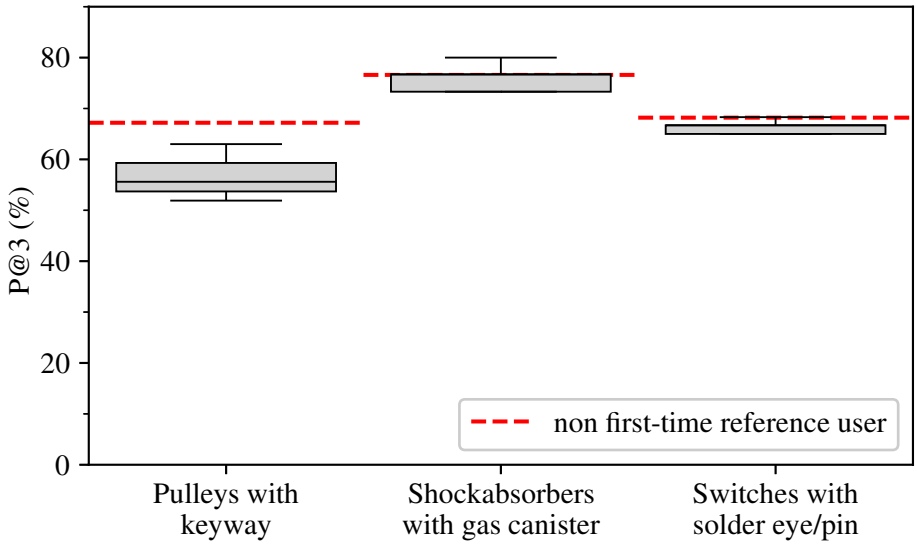


Figure 5.8: Retrieval performance of five first-time users on DMUNet, compared to the baseline of a reference user who is familiar with the retrieval system, represented by the red dashed line. Figure adapted from Herzog and Suwelack, © 2023 Elsevier.

quartile or slightly above for these tasks. The pulley retrieval task is more sensitive to the ROI placement and shows more variation between the first-time users, as well as to the reference user who performs significantly higher.

5.4.4 Results on ABC Dataset with Screws

Lastly, we test our retrieval system on subsets of the ABC dataset, an unannotated, heterogeneous collection of mechanical components. We define retrieval tasks by mixing in 150 screws, highly diverse in terms of size, type and orientation into a data pool of 0, 1000 and 10 000 non-screw parts from the ABC dataset. The data pools are referred to as Screws, Screws+ABC1k and Screws+ABC10k, respectively. The screws are highly diverse in terms of size, type and orientation, as depicted in Figure 5.9. The types of screw features evaluated are listed in Table 5.7.

We conduct experiments on three tasks:

1. locating screws with the same head type,
2. locating screws with partial threading,
3. locating partially threaded screws with the same head type.

Table 5.8 shows the performance of locating screws by design feature in the ABC dataset. For retrieval by head type, the average performance over all

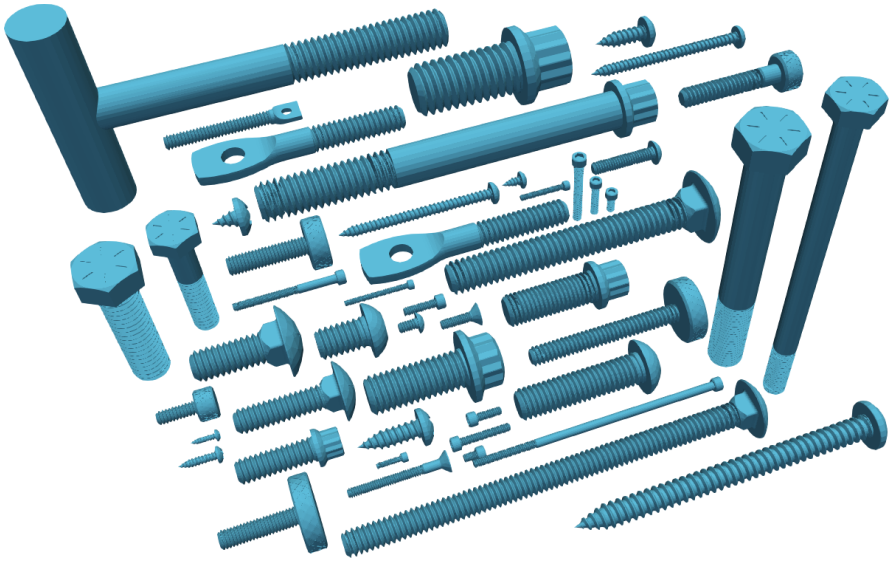


Figure 5.9: Rendering of a random subset of the 150 screws mixed into the ABC dataset. Figure taken from Herzog and Suwelack, © 2023 Elsevier.

Feature	Type									
Head Type	Thumb	12	Tee	5	Socket	12	Button	15	Carriage	30
	Hex	15	Flat	18	Eye	3	12-Point	16	Pan	24
Threading	Full	116	Partial	34						

Table 5.7: Feature types of screw query models for interactive partial retrieval evaluation on Screws+ABC.

head types is reported. On the largest data pool with more than 10 000 mixed mechanical components, we achieve a P@5 rate of 93.7% for locating screws with the same head type. This amounts to an error reduction of 65.2% compared to global retrieval. For locating screws with partial threading, we achieve a P@5 rate of 83.7%, which amounts to an error reduction of 63.0%.

It can be observed that locating screws with the same head type performs stronger than locating screws with partial threading. We presume this is due to the screw heads presenting a more distinctive feature than a change in threading and also being more prominently captured on the global scale. Locating screws based on the combined design features of head type and partial threading performs lower than each of the features on their own. In this task setting, even less correct retrieval candidates are available. On average, there are only 4.5 partially threaded screws per head type present in the data pool with up to 10 150 shapes.

		Screws		Screws+ABC1k		Screws+ABC10k	
Class	Signature	NN	P@5	NN	P@5	NN	P@5
by head type	global	97.3	89.8	95.3	86.6	95.3	81.9
	global+ROI	**99.1**	**97.8**	**99.7**	**97.5**	**97.6**	**93.7**
with partial threading	global	91.2	80.0	88.2	70.6	**88.2**	55.9
	global+ROI	**96.1**	**88.2**	**93.1**	**88.4**	**88.2**	**83.7**
by head type with partial threading	global	84.8	72.2	81.8	68.5	81.8	60.0
	global+ROI	**91.9**	**87.7**	**88.9**	**84.3**	**88.9**	**77.3**

Table 5.8: Performance of locating screws with specific design features in ABC dataset.

Execution Time

We reach an average search time of 853 ms on the Screws+ABC10k dataset with over 10 000 objects. The search time refers to the accumulated index search times up until acquiring the joined retrieval list and is averaged over 100 queries, measured under Intel Core i7-8565U QuadCore 3.6 GHz with 16GB RAM. Note that we optimized the index parameters for accuracy rather than speed, and merely set a constraint of queries under one second to preserve interactivity. Recent studies on index-based similarity search show that search times within a second can be preserved even at a scale of billions of vectors (Johnson et al., 2019; Ren et al., 2020).

5.4.5 Discussion

Our experimental results on three established mechanical component datasets consistently show significant improvements by incorporating user-defined ROIs. Beyond the quantitative retrieval performance, the true value of interactive, user-guided shape search is not fully revealed by the fixed evaluation settings of retrieval benchmarks. Interactive retrieval facilitates an intuitive understanding of search results by allowing the user to immediately see the effects of parameter settings and adjust accordingly. The use of color-coding to indicate the relevance of matched regions in search results improves understanding of which features of an object were considered similar. Existing retrieval benchmarks fail to emulate the locality and ambiguity of design similarity that is found in application-oriented scenarios. As a step towards partial, design-oriented retrieval evaluation, we have formulated fine-grained retrieval tasks on the DMUNet and the ABC dataset.

Limitations and Failure Cases The independent similarity computation for ROIs at different scales with subsequent weighted score aggregation makes our proposed system efficient on a large scale and flexible regarding the positioning of features on different objects. However, failure cases may occur due to difficulties in finding a good weighting between the individual ROIs and the global shape. In addition, very small design features can be missed by our descriptors at the currently defined levels of shape resolution. Lastly, sphere-shaped ROIs may not be appropriate for selecting certain design features such as elongated or ring-like features.

5.5 Summary

This chapter explored an industrial application of vector-based CAD model representations in partial shape retrieval for design reuse. We developed an interactive retrieval system that introduces a novel approach to shape retrieval in heterogeneous CAD model collections. The system allows the definition of weighted, user-defined ROIs to support an ambiguous notion of design similarity with respect to design intent and context. The shape signatures combine traditional and learning-based descriptors at multiple scales. For global shape descriptors, we trained self-supervised representations based on self-reconstruction, rotation invariance, and hierarchical reasoning to exploit the fundamental regularities of geometric data. For local shape regions, we used simple, pose-invariant PFH descriptors. The system facilitates fast, scalable part-to-parts matching through index-based similarity search structures, implemented using the Faiss similarity search library. Extensive experiments demonstrate the effectiveness of our user-guided retrieval system on three datasets of mechanical components. The chapter concluded with a discussion highlighting the qualitative advantages of an interactive retrieval system, while also addressing limitations related to the weighting of ROIs and the selection of design features with elongated or ring-like shapes.

CHAPTER 6

Conclusion

In this thesis, we developed approaches to derive fundamental geometric understanding from unlabeled CAD models to support machine learning applications in product engineering.

6.1 Scientific Contributions

The following sections summarize the scientific contributions of the thesis.

Data-Efficient Learning on CAD Data

In light of the data situation in product engineering, we developed a concept to enhance the data efficiency of machine learning tasks in mechanical engineering. We used transfer learning techniques to transfer knowledge from pretrained feature representations to downstream tasks with limited data. We conducted an experimental study to evaluate the impact of self-supervised and supervised transfer learning strategies on the classification of mechanical components. The results show that self-supervised pretraining achieves a level of performance comparable to supervised pretraining, with particularly strong generalization capabilities across large domain gaps.

Self-Supervised Tasks for CAD Models

A central part of this thesis was the development of self-supervised representation learning techniques specifically tailored to CAD models, extending the current research field beyond methods designed for raw point clouds. We provided a systematic overview of strategies for exploiting CAD-specific information and characteristics in the design of pretext tasks. We demonstrated how the axis alignment of CAD models can be exploited to formulate a 90° rotation invariance pretext task to build rotation-invariant feature representations that exhibit class-discriminative boundary modeling behavior. We introduced a novel cross-modal self-supervised pretraining method that integrates point cloud and multi-view data derived from CAD models to provide a holistic object view. This is achieved by projecting point cloud and image modalities into a shared feature space, where the model is trained on cross-modal correspondences. Our method improves on existing cross-modal methods by incorporating structural correspondences at the local shape level to capture rich contextual cues. Experimental results demonstrate the superior performance of our developed method compared to state-of-the-art self-supervised pretraining methods on several 3D shape benchmarks.

Content-based Shape Retrieval for Design Reuse

We investigated the practical application of self-supervised representation learning in the context of shape retrieval for design reuse. We developed an interactive shape retrieval system for searching in large heterogeneous shape collections. This system employs user-defined, weighted ROIs to support an ambiguous notion of design similarity with respect to user intent and application context, addressing a major shortcoming of current content-based retrieval solutions. The system uses self-supervised representation learning techniques to capture nuanced shape relationships in dense, vector-based global shape signatures. This enables the use of index-based similarity search techniques for achieving interactive, scalable part-to-parts matching. We conducted quantitative evaluations of the developed system on several retrieval datasets. We also discussed qualitative benefits of adjustable, interactive shape search in terms of improved comprehensibility, as well as limitations associated with finding optimal ROI weightings and selecting precise non-spherical regions.

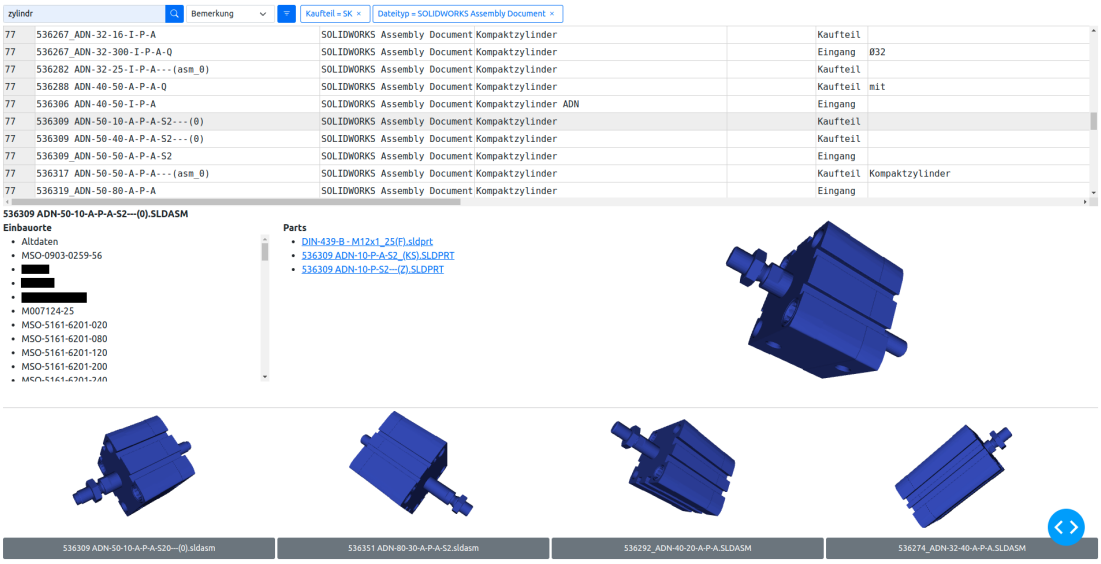


Figure 6.1: Carry-over part search tool that combines text-based search and filtering functionalities with geometric similarity. Geometrically similar parts to a selected component are displayed at the bottom.

6.2 Industrial Applications

This section outlines industrial projects and prototypes that employ self-supervised representation learning techniques for CAD models, demonstrating the relevance and practical utility of such techniques in industrial applications.

Management of Carry-Over Parts

In mechanical engineering and automotive, the term *carry-over parts* is used to describe components that can be reused across multiple assemblies, projects, or product generations with minimal modification. The use of carry-over parts is encouraged to promote standardization and cost efficiency. The management of carry-over parts has become increasingly challenging due to growing product complexity, expanding product variants, and the involvement of numerous suppliers. Poorly maintained legacy data that lacks proper categorization and textual descriptions further complicates the search for relevant parts. Figure 6.1 depicts a prototype search tool developed for a leading manufacturer of industrial connection technology. The tool combines text-based search and filter functionalities with geometric similarity using vector-based shape representations. This dual approach is indented to compensate for incomplete or inaccurate database entries by linking poorly documented parts to geometrically similar

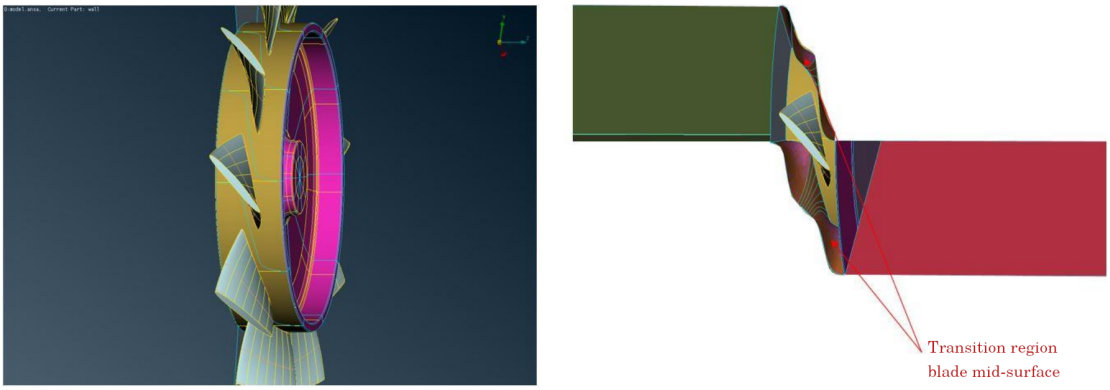


Figure 6.2: Segmentation of an axial fan and definition of the block-structured flow space for CFD simulation. Figure adapted from Heinrich et al. (2022), BMBF project ASTRADIN.

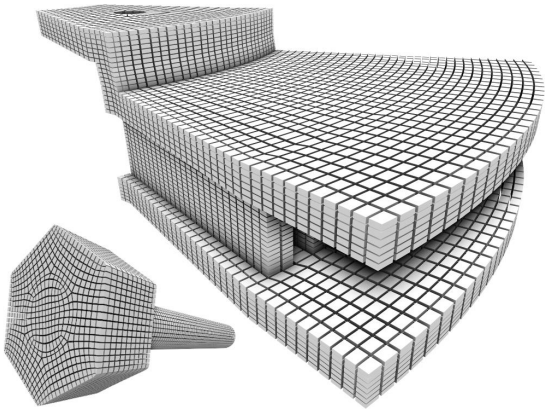


Figure 6.3: Automatically generated sweep hex meshes for a rotational slice of a brake disc and a simple screw.

ones. For fuzzy text search results, it displays geometrically similar alternatives, enabling the discovery of part variants that would otherwise remain hidden due to inaccurate metadata.

Design Optimization for Flow-induced Acoustics

Aeroacoustic design optimization for turbomachinery is a time-consuming process involving numerous iterations of numerical simulations and geometry adjustments. In a research project aimed at automating the design optimization of axial fans, a fully integrated and automated simulation-based process was developed, illustrated in Figure 6.2. This process converts a customer's axial fan geometry into a fully parameterized 3D model, generates geometry variations through Design of Experiments (DoE), and performs CFD simulations for each variation. To parameterize the customer's model, the axial fan is segmented,

and key features are identified. Due to the limited availability of labeled industrial fan data for training, self-supervised pretraining on publicly available axial fan models was employed to improve the data efficiency of the segmentation task.

Data-driven Sweep Hex Meshing

Hexahedral volumetric meshes are highly valued in structural mechanics simulations for their computational efficiency and reliability. A widely used method for creating high-quality hex meshes is sweep meshing, in which a 2D quad mesh is generated on a source surface and extruded along a path to a target surface. This process requires significant manual effort, such as defining appropriate source-target sweep volumes. In a hybrid approach combining geometric and data-driven techniques, an automated sweep hex meshing process was developed for crash and stress analysis, as shown in Figure 6.3. This process incorporates a machine learning-based module to segment objects into sweep volumes that is trained on specific component types, such as brake discs, screws, or seals. Self-supervised pretraining enhances the model's ability to generalize across various component types, reducing the reliance on labeled data for fine-tuning on specific components.

6.3 Outlook

We conclude this thesis with an outlook on potential extensions and future directions of research.

Multimodal Learning with Textual Metadata

This thesis has focused on using information directly embedded within 3D CAD models. A promising avenue for future research is the combination of geometric data with related product information (e.g., part names, descriptions, manufacturing specifications) stored independently in PLM systems. Unlike cross-modal learning, where multi-view features are integrated to improve 3D understanding, this aims at naturally handling and combining different data sources in a single model. Large foundation models, such as GPT-4, are increasingly incorporating multimodal inputs to achieve comprehensive representations of real-world environments (Achiam et al., 2023). Progress

has also been made toward unified representations for language and point clouds (Zhang et al., 2022b; Xue et al., 2023, 2024). For CAD models, this approach offers the potential to learn representations that directly link textual descriptions to specific design elements, supporting applications such as text-based shape manipulation, multimodal feature-based retrieval, and automated compliance checking.

Representation of Shape and Functionality

Current research in 3D representation learning focuses primarily on capturing the geometric shape and visual appearance of objects. However, for many applications in engineering, it is equally important to model the functional aspects of CAD models that control their operational behavior and interactions with other parts. The functional role of a CAD model stems from the designer's understanding of its intended purpose. Elements such as kinematic joints, machining features, and assembly structures each contribute different insights into how a part operates within a larger system. In our proposed shape retrieval system, we enabled users to specify design intent by interactively defining relevant features based on their contextual knowledge. The broader goal is to incorporate functional aspects directly into CAD model representations. This approach aims to create representations that reflect not only the geometry of a part but also its operational behavior within a system.

Creating Parametric CAD Datasets through Reconstruction Techniques

In this thesis, we have examined the limitations of current parametric CAD datasets and identified gaps in both shape complexity and product information richness. To address these gaps, a critical next step is to develop larger, more complex parametric CAD benchmark datasets. This is essential for advancing self-supervised methods that can learn from high-level parametric features to capture underlying design principles and patterns. One potential approach for facilitating the creation of large, complex parametric CAD datasets is the use of CAD model reconstruction techniques. Recent advances in methods for generating code-like CAD modeling operation sequences from point clouds and meshes (Willis et al., 2021; Li et al., 2023b) have improved the automatic conversion of non-parametric data formats, such as STL files or sensor data, into fully parametric models. Looking ahead, these reconstruction methods could help create diverse, complex parametric models at scale.

Applications in Industrial Robotics

Self-supervised representation learning for CAD models also presents opportunities in industrial robotics, at the interface between digital design and physical execution. CAD models provide precise, high-level descriptions of objects that capture abstract geometric features, hierarchical structures, and design characteristics, such as sharp edges and planar surfaces, that are less apparent in raw sensor data. Learning geometric priors from CAD models can potentially enhance tasks such as learning object affordances, grasp synthesis, or assembly planning, which rely on a high-level understanding of geometric and structural object properties and domain-specific knowledge. By encapsulating these high-level geometric and functional design traits, CAD-based representations can improve generalization across similar objects and reduce reliance on noisy, incomplete, or limited sensor data.

List of Symbols

$\mathbb{R}^d$	d-dimensional Euclidean space
$\lvert \cdot \rvert$	Cardinality of a set
$\lVert \cdot \rVert_p$	p-Norm
$.^\top$	Transpose
$\oplus$	Concatenation of vectors
$SO(3)$	Special orthogonal group
$P = \{p_1, \ldots, p_n\}$	Point cloud with n points
f_θ	Neural network f with set of learnable parameters θ
$.^{(t)}$	At iteration t

List of Figures

1.1 Artificial intelligence applications in product engineering 2
1.2 Contributions of the thesis 5

2.1 Solid representation schemes 10
2.2 Mesh generation methods 13
2.3 Illustrations of solid model and face adjacency graph 15
2.4 PFH support region and histograms by Rusu et al. (2009) 17
2.5 Formation of UV-grids by Jayaraman et al. (2021) 22
2.6 MVCNN architecture by Su et al. (2015) 24
2.7 Simplified PointNet architecture 27
2.8 Point cloud autoencoder 32
2.9 Illustration of Triplet Loss 35
2.10 Intra- and cross-modal correspondences by Afham et al. (2022) . . 38

3.1 Supervised, unsupervised and self-supervised learning paradigms 43
3.2 Self-supervised representation learning workflow 44
3.3 Public CAD dataset sizes 46
3.4 Data acquisition and usage concept for product engineering . . . 47
3.5 Supervised and self-supervised transfer strategies 50
3.6 Transfer learning results on DMUNet for different source datasets 55
3.7 Visualization of 2D t-SNE embeddings 56

4.1 Pretext task design for CAD models 60
4.2 Examples of brake discs 61
4.3 Statistics on parametric CAD datasets 62

4.4 Generative and contrastive cross-modal pretraining 67
4.5 Structural point cloud–picture correspondence 68
4.6 Size of pretraining model vs. linear SVM accuracy 69
4.7 Overview of the Pic@Point model architecture 70
4.8 T-SNE embeddings of 90° rotation invariance learning 73
4.9 Visualization of local 2D–3D correspondences 79

5.1 User-centered shape retrieval solution 86
5.2 Global and partial shape retrieval methods 87
5.3 Overview of the proposed retrieval process 88
5.4 Illustration of the index-based search process 92
5.5 Retrieval results on ESB 97
5.6 Comparison of part-in-whole retrieval 98
5.7 Retrieval results on DMUNet 100
5.8 First-time user retrieval performance on DMUNet 102
5.9 Visualization of Screws+ABC dataset 103

6.1 Search tool for carry-over parts 109
6.2 Segmentation of an axial fan for automated design optimization . 110
6.3 Automatically generated sweep hex mesh 110

List of Tables

2.1 Comparison of input representations for 3D object recognition . . 21
2.2 Overview of self-supervised learning methods on point clouds . . 32

3.1 Datasets of transfer learning experiments 52
3.2 Transfer learning results on DMUNet by downstream dataset size 53
3.3 Transfer learning results on MCB-B by downstream dataset size . 53
3.4 Transfer learning results on DMUNet by source dataset size . . . 55

4.1 Rotation invariance task linear SVM results 74
4.2 Linear SVM classification on ModelNet40 and ScanObjectNN . . 76
4.3 Classification on ScanObjectNN variants 78
4.4 Ablation studies of Pic@Point 80
4.5 Part segmentation on ShapeNetPart 80
4.6 Scene-level semantic segmentation on S3DIS 81
4.7 Memory and time costs of Pic@Point pretraining 82

5.1 Global retrieval performance on ESB 95
5.2 Categories of query models on ESB 96
5.3 Partial retrieval performance on ESB 98
5.4 Categories of query models on DMUNet 100
5.5 Partial retrieval performance on DMUNet 100
5.6 Ablation of the pretext tasks of the global descriptor 101
5.7 Feature types of screw query models 103
5.8 Partial retrieval performance on ABC 104

List of Algorithms

1 90° Rotation Invariance Learning with Memory Bank 64

Bibliography

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*. *Cited on page 111.*

Afham, M., Dissanayake, I., Dissanayake, D., Dharmasiri, A., Thilakarathna, K., and Rodrigo, R. (2022). Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9902–9912. *Cited on pages 32, 38, 67, 68, 76, 77, 80, 81, and 119.*

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., and Savarese, S. (2016). 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1534–1543. *Cited on pages 75 and 81.*

Armstrong, C. G., Fogg, H. J., Tierney, C. M., and Robinson, T. T. (2015). Common themes in multi-block structured quad/hex mesh generation. *Procedia Engineering*, 124:70–82. *Cited on page 14.*

Bai, X., Bai, S., Zhu, Z., and Latecki, L. J. (2015). 3d shape matching via two layer coding. *IEEE transactions on pattern analysis and machine intelligence*, 37(12):2361–2373. *Cited on page 95.*

Baumgart, B. G. (1974). *Geometric modeling for computer vision.* Stanford University. *Cited on page 11.*

Beaini, D., Passaro, S., Létourneau, V., Hamilton, W., Corso, G., and Liò, P. (2021). Directional graph networks. In *International Conference on Machine Learning*, pages 748–758. PMLR. *Cited on pages 21 and 23.*

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828. *Cited on page 43.*

Bespalov, D., Regli, W. C., and Shokoufandeh, A. (2003). Reeb graph based shape retrieval for cad. In *International design engineering technical conferences and computers and information in engineering conference*, volume 36991, pages 229–238. *Cited on page 16.*

Bespalov, D., Regli, W. C., and Shokoufandeh, A. (2006). Local feature extraction and matching partial objects. *Computer-Aided Design*, 38(9):1020–1037. *Cited on page 87.*

Biasotti, S., Marini, S., Spagnuolo, M., and Falcidieno, B. (2006). Sub-part correspondence by structural descriptors of 3d shapes. *Computer-Aided Design*, 38(9):1002–1019. *Cited on page 16.*

Bickel, S., Schleich, B., and Wartzack, S. (2023). A novel shape retrieval method for 3d mechanical components based on object projection, pre-trained deep learning models and autoencoder. *Computer-Aided Design*, 154:103417. *Cited on page 95.*

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. (2022). Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668. *Cited on pages 21 and 23.*

Braid, I. C. (1975). The synthesis of solids bounded by many faces. *Communications of the ACM*, 18(4):209–216. *Cited on page 11.*

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*. *Cited on pages 19, 63, and 89.*

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42. *Cited on page 19.*

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*. *Cited on page 22.*

Cao, W., Robinson, T., Hua, Y., Boussuge, F., Colligan, A. R., and Pan, W. (2020). Graph representation of 3d cad models for machining feature recognition with deep learning. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 84003, page V11AT11A003. American Society of Mechanical Engineers. *Cited on pages 20 and 21.*

Carter, I. M. (1990). Applications and prospects for ai in mechanical engineering design. *The Knowledge Engineering Review*, 5(3):167–179. *Cited on page 2.*

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*. *Cited on pages 46, 47, and 75.*

Chen, B., Xia, Y., Zang, Y., Wang, C., and Li, J. (2023a). Decoupled local aggregation for point cloud learning. *arXiv preprint arXiv:2308.16532*. *Cited on page 21.*

Chen, D.-Y., Tian, X.-P., Shen, Y.-T., and Ouhyoung, M. (2003). On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pages 223–232. Wiley Online Library. *Cited on pages 16 and 95.*

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR. *Cited on page 36.*

Chen, Y., Liu, J., Zhang, X., Qi, X., and Jia, J. (2023b). Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21674–21683. *Cited on page 24.*

Choy, C., Gwak, J., and Savarese, S. (2019). 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084. *Cited on pages 21 and 24.*

Cicirello, V. and Regli, W. C. (2001). Machining feature-based comparisons of mechanical parts. In *Proceedings International Conference on Shape Modeling and Applications*, pages 176–185. IEEE. *Cited on page 15.*

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. (2018). Spherical cnns. *arXiv preprint arXiv:1801.10130*. *Cited on page 63.*

Colligan, A. R., Robinson, T. T., Nolan, D. C., Hua, Y., and Cao, W. (2022). Hierarchical cadnet: Learning from b-reps for machining feature recognition. *Computer-Aided Design*, 147:103226. *Cited on pages 20 and 21.*

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. (2020). Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271. *Cited on page 21.*

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29. *Cited on page 22.*

Dekhtiar, J., Durupt, A., Bricogne, M., Eynard, B., Rowson, H., and Kiritsis, D. (2018). Deep learning for big data applications in cad and plm – research review, opportunities and case study. *Computers in Industry*, 100. *Cited on pages 52, 54, 94, and 99.*

Delaunay, B. et al. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2. *Cited on page 13.*

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. *Cited on page 48.*

Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., and Li, H. (2021). Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 1201–1209. *Cited on page 24.*

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. *Cited on page 31.*

Dong, R., Qi, Z., Zhang, L., Zhang, J., Sun, J., Ge, Z., Yi, L., and Ma, K. (2022). Autoencoders as cross-modal teachers: Can pretrained 2d image transform-

ers help 3d representation learning? *arXiv preprint arXiv:2212.08320*. *Cited on pages 37 and 78.*

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. (2024). The faiss library. *arXiv preprint arXiv:2401.08281*. *Cited on page 91.*

El-Mehalawi, M. and Miller, R. A. (2003). A database system of mechanical components based on geometric and topological similarity. part i: representation. *Computer-Aided Design*, 35(1):83–94. *Cited on page 15.*

Elad, M., Tal, A., and Ar, S. (2002). Content based retrieval of vrml objects—an iterative and interactive approach. In *Multimedia 2001: Proceedings of the Eurographics Workshop in Manchester, United Kingdom, September 8–9, 2001*, pages 107–118. Springer. *Cited on page 88.*

Elinson, A., Nau, D. S., and Regli, W. C. (1997). Feature-based similarity assessment of solid models. In *Proceedings of the fourth ACM symposium on Solid modeling and applications*, pages 297–310. *Cited on page 15.*

Engelmann, F., Kontogianni, T., Hermans, A., and Leibe, B. (2017). Exploring spatial context for 3d semantic segmentation of point clouds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 716–724. *Cited on page 26.*

Ericsson, L., Gouk, H., Loy, C. C., and Hospedales, T. M. (2022). Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62. *Cited on pages 41 and 42.*

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. (2018). Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68. *Cited on page 63.*

Fei, J. and Deng, Z. (2024). Rotation invariance and equivariance in 3d deep learning: a survey. *Artificial Intelligence Review*, 57(7):168. *Cited on page 65.*

Fey, M., Lenssen, J. E., Weichert, F., and Müller, H. (2018). Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877. *Cited on page 22.*

George, J. A. (1971). *Computer implementation of the finite element method*. Stanford University. *Cited on page 13.*

Giorgi, D., Frosini, P., Spagnuolo, M., and Falcidieno, B. (2010). 3d relevance feedback via multilevel relevance judgements. *The Visual Computer*, 26:1321–1338. *Cited on page 88.*

Goyal, P., Mahajan, D., Gupta, A., and Misra, I. (2019). Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6391–6400. *Cited on page 49.*

Graham, B., Engelcke, M., and Van Der Maaten, L. (2018). 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232. *Cited on page 24.*

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284. *Cited on page 36.*

Guo, H.-X., Liu, Y., Pan, H., and Guo, B. (2022). Implicit conversion of manifold b-rep solids by neural halfspace representation. *ACM Transactions on Graphics (TOG)*, 41(6):1–15. *Cited on pages 20 and 21.*

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S.-M. (2021). Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199. *Cited on pages 21 and 29.*

Guo, W., Wang, J., and Wang, S. (2019). Deep multimodal representation learning: A survey. *Ieee Access*, 7:63373–63394. *Cited on page 37.*

Hamdi, A., Giancola, S., and Ghanem, B. (2021). Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1–11. *Cited on pages 21 and 24.*

Han, X., Tang, Y., Wang, Z., and Li, X. (2024). Mamba3d: Enhancing local features for 3d point cloud analysis via state space model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 4995–5004. *Cited on page 21.*

Han, Z., Shang, M., Liu, Y.-S., and Zwicker, M. (2019). View inter-prediction gan: Unsupervised representation learning for 3d shapes by learning global

shape memories to support local view predictions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 8376–8384. *Cited on page 76.*

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., and Cohen-Or, D. (2019). Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12. *Cited on pages 21 and 23.*

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009. *Cited on page 33.*

He, K., Girshick, R., and Dollár, P. (2019). Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4927. *Cited on page 48.*

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. *Cited on pages 71 and 75.*

Heinrich, J., Herzog, V., Pantle, I., and Demer, J. (2022). *Schlussbericht für das Verbundvorhaben "ASTRADIN" - Automatisierte Strömungsaksustische Designoptimierung für die Industrie: BMBF-Förderprogramm "KMU-Innovativ". Berichtszeitraum: von: 01.07.2019 bis: 31.03.2022*. CAIQ GmbH. *Cited on page 110.*

Herzog, V. and Suwelack, S. (2022). Data-efficient machine learning on three-dimensional engineering data. *Journal of Mechanical Design*, 144(2):021709. *Cited on pages 6, 45, 46, 47, 50, 52, 55, and 56.*

Herzog, V. and Suwelack, S. (2023). Bridging the gap between geometry and user intent: Retrieval of cad models via regions of interest. *Computer-Aided Design*, page 103573. *Cited on pages 6, 64, 86, 88, 92, 97, 98, 100, 102, and 103.*

Herzog, V. and Suwelack, S. (2025). Pic@Point: Cross-modal learning by local and global point-picture correspondence. In Nguyen, V. and Lin, H.-T., editors, *Proceedings of the 16th Asian Conference on Machine Learning*, volume 260 of *Proceedings of Machine Learning Research*, pages 703–718. PMLR. *Cited on pages 6, 66, 68, 69, 70, and 79.*

Hilaga, M., Shinagawa, Y., Kohmura, T., and Kunii, T. L. (2001). Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212. *Cited on page 16.*

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*. *Cited on pages 35 and 69.*

Ho-Le, K. (1988). Finite element mesh generation methods: a review and classification. *Computer-aided design*, 20(1):27–38. *Cited on page 14.*

Hou, J., Luo, C., Qin, F., Shao, Y., and Chen, X. (2023). Fus-gcn: Efficient b-rep based graph convolutional networks for 3d-cad model classification and retrieval. *Advanced Engineering Informatics*, 56:102008. *Cited on pages 20 and 21.*

Hu, S.-M., Liu, Z.-N., Guo, M.-H., Cai, J.-X., Huang, J., Mu, T.-J., and Martin, R. R. (2022). Subdivision-based mesh convolution networks. *ACM Transactions on Graphics (TOG)*, 41(3):1–16. *Cited on pages 21 and 23.*

Huang, Q., Wang, F., and Guibas, L. (2014). Functional map networks for analyzing and exploring large shape collections. *ACM Transactions on Graphics (ToG)*, 33(4):1–11. *Cited on page 88.*

Huang, Q., Wang, W., and Neumann, U. (2018). Recurrent slice networks for 3d segmentation of point clouds. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2635. *Cited on page 26.*

Huang, S., Xie, Y., Zhu, S.-C., and Zhu, Y. (2021). Spatio-temporal self-supervised representation learning for 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6535–6545. *Cited on pages 32, 36, and 76.*

Huang, T., Ding, Z., Zhang, J., Tai, Y., Zhang, Z., Chen, M., Wang, C., and Liu, Y. (2023). Learning to measure the point cloud reconstruction loss in a representation space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12208–12217. *Cited on page 34.*

Iyer, N., Jayanti, S., Lou, K., Kalyanaraman, Y., and Ramani, K. (2005). Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509–530. *Cited on page 14.*

Jagadeesan, P., Wenzel, J., Corney, J., Yan, X., Sherlock, A., Torres-Sanchez, C., and Regli, W. (2009). Validation of purdue engineering shape benchmark clusters by crowdsourcing. In *Proceedings of the international conference on product lifecycle management*. *Cited on page 96.*

Jayanti, S., Kalyanaraman, Y., Iyer, N., and Ramani, K. (2006). Developing an engineering shape benchmark for cad models. *Computer-Aided Design*, 38(9):939–953. *Cited on page 94.*

Jayaraman, P. K., Sanghi, A., Lambourne, J. G., Willis, K. D., Davies, T., Shayani, H., and Morris, N. (2021). Uv-net: Learning from boundary representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11703–11712. *Cited on pages 20, 21, 22, 61, and 119.*

Jing, L., Zhang, L., and Tian, Y. (2021). Self-supervised feature learning by cross-modality and cross-view correspondences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1581–1591. *Cited on pages 32 and 38.*

Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449. *Cited on page 17.*

Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547. *Cited on pages 91 and 104.*

Jones, B., Hildreth, D., Chen, D., Baran, I., Kim, V. G., and Schulz, A. (2021). Automate: A dataset and learning approach for automatic mating of cad assemblies. *ACM Transactions on Graphics (TOG)*, 40(6):1–18. *Cited on page 60.*

Kanezaki, A., Matsushita, Y., and Nishida, Y. (2018). Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5010–5019. *Cited on page 21.*

Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. (2003). Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164. *Cited on pages 16 and 95.*

Kazmi, I. K., You, L., and Zhang, J. J. (2013). A survey of 2d and 3d shape descriptors. In *2013 10th International Conference Computer Graphics, Imaging and Visualization*, pages 1–10. IEEE. *Cited on page 14.*

Kim, S., Chi, H.-g., Hu, X., Huang, Q., and Ramani, K. (2020). A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. *European Conference on Computer Vision*. *Cited on page 52.*

Kim, V. G., Li, W., Mitra, N. J., DiVerdi, S., and Funkhouser, T. (2012). Exploring collections of 3d models using fuzzy correspondences. *ACM Transactions on Graphics (TOG)*, 31(4):1–11. *Cited on page 88.*

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*. *Cited on pages 21 and 22.*

Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., and Panozzo, D. (2019a). Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611. *Cited on pages 47 and 94.*

Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., and Panozzo, D. (2019b). Abc dataset normal estimation benchmark. *Cited on page 52.*

Kornblith, S., Shlens, J., and Le, Q. V. (2019). Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2661–2671. *Cited on page 48.*

Lahav, A. and Tal, A. (2020). Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)*, 39(6):1–13. *Cited on pages 21 and 23.*

Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X., and Jia, J. (2022). Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8500–8509. *Cited on pages 21 and 30.*

Lambourne, J. G., Willis, K. D., Jayaraman, P. K., Sanghi, A., Meltzer, P., and Shayani, H. (2021). Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12773–12782. *Cited on pages 20, 21, and 61.*

Lan, S., Yu, R., Yu, G., and Davis, L. S. (2019). Modeling local geometric structure of 3d point clouds using geo-cnn. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 998–1008. *Cited on page 29.*

Leifman, G., Meir, R., and Tal, A. (2005). Semantic-oriented 3d shape retrieval using relevance feedback. *The Visual Computer*, 21:865–875. *Cited on page 88.*

Li, J., Liu, Z., Li, L., Lin, J., Yao, J., and Tu, J. (2023a). Multi-view convolutional vision transformer for 3d object recognition. *Journal of Visual Communication and Image Representation*, 95:103906. *Cited on page 21.*

Li, P., Guo, J., Zhang, X., and Yan, D.-M. (2023b). Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16826. *Cited on page 112.*

Li, P., He, F., Fan, B., and Song, Y. (2023c). Tpnet: A novel mesh analysis method via topology preservation and perception enhancement. *Computer Aided Geometric Design*, 104:102219. *Cited on pages 21 and 23.*

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018). Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31. *Cited on pages 26 and 29.*

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., and Tang, J. (2021). Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering*, 35(1):857–876. *Cited on page 32.*

Liu, Z.-B., Bu, S.-H., Zhou, K., Gao, S.-M., Han, J.-W., and Wu, J. (2013). A survey on partial retrieval of 3d shapes. *Journal of Computer Science and Technology*, 28(5):836–851. *Cited on page 87.*

Loncaric, S. (1998). A survey of shape analysis techniques. *Pattern recognition*, 31(8):983–1001. *Cited on page 14.*

Lupinetti, K., Pernot, J.-P., Monti, M., and Giannini, F. (2019). Content-based cad assembly model retrieval: Survey and future challenges. *Computer-Aided Design*, 113:62–81. *Cited on page 60.*

Ma, L., Huang, Z., and Wang, Y. (2010). Automatic discovery of common design structures in cad models. *Computers & Graphics*, 34(5):545–555. *Cited on page 16.*

Ma, X., Qin, C., You, H., Ran, H., and Fu, Y. (2022). Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*. *Cited on pages 21, 28, 77, 78, and 80.*

Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE. *Cited on pages 21, 23, and 26.*

McWherter, D., Peabody, M., Regli, W. C., and Shokoufandeh, A. (2001). Transformation invariant shape similarity comparison of solid models. In *International design engineering technical conferences and computers and information in engineering conference*, volume 80241, pages 303–312. American Society of Mechanical Engineers. *Cited on page 15.*

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124. *Cited on page 22.*

Muraleedharan, L. P., Kannan, S. S., and Muthuganapathy, R. (2019). Autoencoder-based part clustering for part-in-whole retrieval of cad models. *Computers & Graphics*, 81:41–51. *Cited on page 96.*

Newell, A. and Deng, J. (2020). How useful is self-supervised pretraining for visual tasks? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7354. *Cited on page 49.*

Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer. *Cited on page 35.*

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*. *Cited on pages 35, 36, and 69.*

Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D. (2002). Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832. *Cited on page 16.*

Owen, S. J. (1998). A survey of unstructured mesh generation technology. *IMR*, 239(267):15. *Cited on page 14.*

Pang, Y., Wang, W., Tay, F. E., Liu, W., Tian, Y., and Yuan, L. (2022). Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision*, pages 604–621. Springer. *Cited on pages 32, 33, 76, 78, 80, and 81.*

Papadakis, P., Pratikakis, I., Theoharis, T., and Perantonis, S. (2010). Panorama: A 3d shape descriptor based on panoramic views for unsupervised 3d object retrieval. *International Journal of Computer Vision*, 89:177–192. *Cited on page 95.*

Park, J., Lee, S., Kim, S., Xiong, Y., and Kim, H. J. (2023). Self-positioning point-based transformer for point cloud understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21814–21823. *Cited on page 30.*

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660. *Cited on pages 26, 27, 78, 80, 81, and 90.*

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30. *Cited on pages 27, 49, and 51.*

Qi, Z., Dong, R., Fan, G., Ge, Z., Zhang, X., Ma, K., and Yi, L. (2023). Contrast with reconstruct: Contrastive 3d representation learning guided by generative pretraining. In *International Conference on Machine Learning*, pages 28223–28243. PMLR. *Cited on pages 37 and 78.*

Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M., and Ghanem, B. (2022a). Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35:23192–23204. *Cited on pages 21, 28, 75, 76, 77, 78, 80, and 81.*

Qian, G., Zhang, X., Hamdi, A., and Ghanem, B. (2022b). Pix4point: Image pretrained transformers for 3d point cloud understanding. *arXiv preprint arXiv:2208.12259*. *Cited on pages 37 and 78.*

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR. *Cited on pages 37 and 66.*

Rao, Y., Lu, J., and Zhou, J. (2020). Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5376–5385. *Cited on pages 32, 36, 49, 51, 69, and 90.*

Ren, J., Zhang, M., and Li, D. (2020). Hm-ann: Efficient billion-point nearest neighbor search on heterogeneous memory. *Advances in Neural Information Processing Systems*, 33:10672–10684. *Cited on pages 91 and 104.*

Riegler, G., Osman Ulusoy, A., and Geiger, A. (2017). Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586. *Cited on pages 21 and 24.*

Rostami, R., Bashiri, F. S., Rostami, B., and Yu, Z. (2019). A survey on data-driven 3d shape descriptors. In *Computer Graphics Forum*, volume 38, pages 356–393. Wiley Online Library. *Cited on page 17.*

Ruan, L., Ma, Y., Yang, H., He, H., Liu, B., Fu, J., Yuan, N. J., Jin, Q., and Guo, B. (2023). Mm-diffusion: Learning multi-modal diffusion models for joint audio and video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10219–10228. *Cited on page 66.*

Rustamov, R. M., Ovsjanikov, M., Azencot, O., Ben-Chen, M., Chazal, F., and Guibas, L. (2013). Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 32(4):1–12. *Cited on page 88.*

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE. *Cited on pages 17 and 119.*

Rusu, R. B., Marton, Z. C., Blodow, N., and Beetz, M. (2008). Persistent point feature histograms for 3d point clouds. In *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, pages 119–128. *Cited on pages 17, 90, and 91.*

Salti, S., Tombari, F., and Di Stefano, L. (2014). Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264. *Cited on page 17.*

Sanghi, A. (2020). Info3d: Representation learning on 3d objects using mutual information maximization and contrastive learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020,*

Proceedings, Part XXIX 16, pages 626–642. Springer. *Cited on pages 32, 36, and 64.*

Sauder, J. and Sievers, B. (2019). Self-supervised deep learning on point clouds by reconstructing space. *Advances in Neural Information Processing Systems*, 32. *Cited on pages 32, 35, 76, 80, and 81.*

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823. *Cited on pages 34 and 35.*

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. (2022). Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294. *Cited on page 46.*

Shapiro, V. (2002). Solid modeling. *Handbook of computer aided geometric design*, 20:473–518. *Cited on pages 10 and 18.*

Sharp, N., Attaiki, S., Crane, K., and Ovsjanikov, M. (2022). Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3):1–16. *Cited on pages 21 and 23.*

Shimada, K. (2006). Current trends and issues in automatic mesh generation. *Computer-Aided Design and Applications*, 3(6):741–750. *Cited on page 14.*

Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D. J., and Sinop, A. K. (2023). Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pages 31613–31632. PMLR. *Cited on page 21.*

Smeulders, A. W., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on pattern analysis and machine intelligence*, 22(12):1349–1380. *Cited on page 88.*

Smirnov, D. and Solomon, J. (2021). Hodgenet: Learning spectral geometry on triangle meshes. *ACM Transactions on Graphics (TOG)*, 40(4):1–11. *Cited on page 21.*

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29. *Cited on page 36.*

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953. *Cited on pages 21, 24, 26, and 119.*

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852. *Cited on pages 48 and 49.*

Sun, H., Wang, Y., Wang, P., Cai, X., and Li, D. (2023). Viewformer: View set attention for multi-view 3d shape understanding. *arXiv preprint arXiv:2305.00161*. *Cited on pages 21 and 24.*

Sun, J., Ovsjanikov, M., and Guibas, L. (2009). A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library. *Cited on page 17.*

Tao, F., Qi, Q., Liu, A., and Kusiak, A. (2018). Data-driven smart manufacturing. *Journal of Manufacturing Systems*, 48:157–169. *Cited on page 46.*

Tao, S., Huang, Z., Ma, L., Guo, S., Wang, S., and Xie, Y. (2013). Partial retrieval of cad models based on local surface region decomposition. *Computer-Aided Design*, 45(11):1239–1252. *Cited on page 87.*

Tatsuma, A. and Aono, M. (2009). Multi-fourier spectra descriptor and augmentation with spectral clustering for 3d shape retrieval. *The Visual Computer*, 25:785–804. *Cited on page 95.*

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420. *Cited on page 29.*

Tran, B., Hua, B.-S., Tran, A. T., and Hoai, M. (2022). Self-supervised learning with multi-view rendering for 3d point cloud analysis. In *Proceedings of the Asian Conference on Computer Vision*, pages 3086–3103. *Cited on page 67.*

Ullman, D. G. (2003). *The mechanical design process*. *Cited on page 4.*

Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, T., and Yeung, S.-K. (2019). Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597. *Cited on pages 75 and 76.*

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11). *Cited on page 56.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30. *Cited on pages 29, 78, 80, and 81.*

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*. *Cited on page 21.*

Voelcker, H. and Requicha, A. (1977). Constructive solid geometry. *Cited on page 10.*

Vranic, D. V. (2005). Desire: a composite 3d-shape descriptor. In *2005 IEEE international conference on multimedia and expo*, pages 4–pp. IEEE. *Cited on pages 16 and 95.*

Wang, H., Liu, Q., Yue, X., Lasenby, J., and Kusner, M. J. (2021). Unsupervised point cloud pre-training via occlusion completion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9782–9792. *Cited on pages 32, 33, 76, 80, and 81.*

Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., and Tong, X. (2017). O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11. *Cited on pages 21 and 24.*

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12. *Cited on pages 21, 28, 75, 78, 80, and 81.*

Wang, Z., Yu, X., Rao, Y., Zhou, J., and Lu, J. (2022). P2p: Tuning pre-trained image models for point cloud analysis with point-to-pixel prompting. *Advances in neural information processing systems*, 35:14388–14402. *Cited on pages 37 and 78.*

Wang, Z., Yu, X., Rao, Y., Zhou, J., and Lu, J. (2023). Take-a-photo: 3d-to-2d generative pre-training of point cloud models. In *Proceedings of the IEEE/CVF*

International Conference on Computer Vision, pages 5640–5650. *Cited on pages 32, 39, 67, 68, 78, and 79.*

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*. *Cited on page 46.*

Wei, X., Yu, R., and Sun, J. (2020). View-gcn: View-based graph convolutional network for 3d shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1850–1859. *Cited on pages 21 and 24.*

Willis, K. D., Jayaraman, P. K., Chu, H., Tian, Y., Li, Y., Grandi, D., Sanghi, A., Tran, L., Lambourne, J. G., Solar-Lezama, A., et al. (2022). Joinable: Learning bottom-up assembly of parametric cad joints. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15849–15860. *Cited on page 60.*

Willis, K. D. D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J. G., Solar-Lezama, A., and Matusik, W. (2021). Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4). *Cited on page 112.*

Wright, P. K. and Bourne, D. A. (1988). *Manufacturing intelligence*. Addison-Wesley Longman Publishing Co., Inc. *Cited on page 2.*

Wu, T., Pan, L., Zhang, J., Wang, T., Liu, Z., and Lin, D. (2021). Density-aware chamfer distance as a comprehensive metric for point cloud completion. *arXiv preprint arXiv:2111.12702*. *Cited on page 34.*

Wu, W., Qi, Z., and Fuxin, L. (2019). Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630. *Cited on page 29.*

Wu, Y., Liu, J., Gong, M., Gong, P., Fan, X., Qin, A., Miao, Q., and Ma, W. (2023). Self-supervised intra-modal and cross-modal contrastive learning for point cloud understanding. *IEEE Transactions on Multimedia*. *Cited on page 68.*

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920. *Cited on pages 21, 23, 26, 47, 52, 75, and 76.*

Xiao, A., Huang, J., Guan, D., and Lu, S. (2022). Unsupervised representation learning for point clouds: A survey. *arXiv preprint arXiv:2202.13589*. *Cited on page 31.*

Xu, C., Yang, S., Galanti, T., Wu, B., Yue, X., Zhai, B., Zhan, W., Vajda, P., Keutzer, K., and Tomizuka, M. (2022). Image2point: 3d point-cloud understanding with 2d image pretrained models. In *European Conference on Computer Vision*, pages 638–656. Springer. *Cited on page 37.*

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018a). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*. *Cited on page 23.*

Xu, M., Ding, R., Zhao, H., and Qi, X. (2021). Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3173–3182. *Cited on page 29.*

Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y. (2018b). Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European conference on computer vision (ECCV)*, pages 87–102. *Cited on page 29.*

Xue, L., Gao, M., Xing, C., Martín-Martín, R., Wu, J., Xiong, C., Xu, R., Niebles, J. C., and Savarese, S. (2023). Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1179–1189. *Cited on page 112.*

Xue, L., Yu, N., Zhang, S., Panagopoulou, A., Li, J., Martín-Martín, R., Wu, J., Xiong, C., Xu, R., Niebles, J. C., et al. (2024). Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27091–27101. *Cited on page 112.*

Yan, S., Yang, Z., Li, H., Song, C., Guan, L., Kang, H., Hua, G., and Huang, Q. (2023). Implicit autoencoder for point-cloud self-supervised representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14530–14542. *Cited on pages 32 and 34.*

Yang, Y., Feng, C., Shen, Y., and Tian, D. (2018). Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215. *Cited on pages 32, 33, 76, and 90.*

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. (2016). A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12. *Cited on pages 75 and 80.*

Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., and Lu, J. (2022). Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322. *Cited on pages 32, 33, 76, 78, 80, and 81.*

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. *Advances in neural information processing systems*, 30. *Cited on page 26.*

Zhang, Q. and Hou, J. (2023). Pointvst: Self-supervised pre-training for 3d point clouds via view-specific point-to-image translation. *IEEE Transactions on Visualization and Computer Graphics*. *Cited on pages 32, 39, 66, 68, 76, 77, 78, 80, and 81.*

Zhang, R., Guo, Z., Gao, P., Fang, R., Zhao, B., Wang, D., Qiao, Y., and Li, H. (2022a). Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training. *Advances in neural information processing systems*, 35:27061–27074. *Cited on pages 76, 77, and 78.*

Zhang, R., Guo, Z., Zhang, W., Li, K., Miao, X., Cui, B., Qiao, Y., Gao, P., and Li, H. (2022b). Pointclip: Point cloud understanding by clip. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8552–8562. *Cited on pages 37 and 112.*

Zhang, R., Wang, L., Qiao, Y., Gao, P., and Li, H. (2023). Learning 3d representations from 2d pre-trained models via image-to-point masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21769–21780. *Cited on page 78.*

Zhang, Y., Jiang, H., Miura, Y., Manning, C. D., and Langlotz, C. P. (2022c). Contrastive learning of medical visual representations from paired images and text. In *Machine Learning for Healthcare Conference*, pages 2–25. PMLR. *Cited on page 66.*

Zhang, Z., Hua, B.-S., Rosen, D. W., and Yeung, S.-K. (2019). Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International conference on 3d vision (3DV)*, pages 204–213. IEEE. *Cited on page 63.*

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. (2021). Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268. *Cited on page 30.*

Zhou, Y. and Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499. *Cited on page 24.*

List of Publications

Journal Articles

- Vencia Herzog and Stefan Suwelack. "Data-Efficient Machine Learning on Three-Dimensional Engineering Data". In: *Journal of Mechanical Design*, 144.2 (2022). DOI: 10.1115/1.4052753.
- Vencia Herzog and Stefan Suwelack. "Bridging the Gap between Geometry and User Intent: Retrieval of CAD Models via Regions of Interest". In: *Computer-Aided Design*, 163 (2023). DOI: 10.1016/j.cad.2023.103573.

Articles in Conference Proceedings

- Vencia Herzog and Stefan Suwelack. "Pic@Point: Cross-Modal Learning by Local and Global Point-Picture Correspondence". In: *Proceedings of the 16th Asian Conference on Machine Learning*, 260 (2025). URL: https://proceedings.mlr.press/v260/herzog25a.html.