# Crown-Generation part 01: Segmentation

Waleed Alzamil

January 27, 2025

# Abstract

Automating dental crown generation is a transformative goal in the dental industry, aiming to reduce the time, effort, and cost associated with traditional manual design. This thesis focuses on the crucial first step in this process: segmentation of dental structures using advanced deep learning models like Point Cloud Transformer (PCT) and Dynamic Graph Convolutional Neural Network (DGCNN). Segmentation is essential for isolating the tooth and its surrounding context, providing a structured input for subsequent crown generation tasks.

The proposed approach evaluates and benchmarks these models on 3D dental datasets, emphasizing their ability to handle complex geometries and variations across different tooth positions. Key contributions include the development of a robust segmentation pipeline, optimized training methodologies, and the exploration of metrics tailored for dental applications.

This segmentation framework lays the groundwork for the second semester's focus on point cloud generation, where the extracted context will be utilized to produce anatomically accurate and aesthetically pleasing dental crowns. By addressing segmentation challenges, this work aims to establish a foundation for end-to-end automation in dental crown design, ultimately enhancing the accuracy and efficiency of dental restorations.

## Acknowledgments

Training such 3D models would not have been possible without fastmesh (a mesh parser in Rust). Fastmesh employs several techniques to parse meshes efficiently, such as dealing with bytes and byte arrays instead of strings, as strings are inherently slow and require extra time for validation. Additionally, Fastmesh uses data-parallelism whenever possible, although this might be a liability when data is relatively small. Fastmesh also utilizes Single-Instruction-Multiple-Data (SIMD) CPU features, which are present in most modern CPUs. Another technique that proved useful for point cloud-based models is lazy parsing, which means that fastmesh lazily parses only the needed mesh elements (vertices, faces, normals, or textures), without eagerly parsing the whole file, leaving the decision to the user. There is also a custom mesh file format, bmesh, that significantly reduced the original mesh files (by about 3X on average) to be later uploaded on training platforms such as Kaggle.

# Contents

# 1 Introduction

## 1.1 Background

Dental crown design is a cornerstone of restorative dentistry, demanding exceptional precision to ensure functionality, durability, and aesthetic harmony. Traditionally, the process of crown generation has relied on labor-intensive manual methods, even with advancements in computer-aided design (CAD) software. However, recent developments in artificial intelligence (AI) and deep learning have introduced promising opportunities for automation, targeting levels of accuracy and efficiency comparable to human expertise. Among the various stages of crown generation, segmentation of dental structures emerges as a pivotal step, facilitating the identification and isolation of the prepared tooth along with its surrounding anatomical context.

## 1.2 Problem Statement

The segmentation of point clouds represents a critical task in dental crown automation, as it provides the essential context for targeted processing. For instance, generating a crown for tooth number *35* requires a pipeline that takes as input the entire point cloud of the neighbouring teeth, such as *35, 36, 34, 25, 24, and 26.* This necessitates classifying every point in the cloud into one of 33 classes, encompassing 32 teeth along with a separate class for gingiva. Despite advancements in deep learning for 3D data, accurately segmenting complex dental structures remains a significant challenge. The intricate geometry of teeth, combined with variability in shapes and occlusions present in scans, complicates this task. Existing methodologies often fail to deliver the level of detail and precision essential for downstream processes such as crown generation, hindering the realization of fully automated dental workflows.

## 1.3 Objective

This thesis seeks to address the challenges associated with dental structure segmentation by leveraging deep learning models, including the Point Cloud Transformer (PCT) and Dynamic Graph Convolutional Neural Network (DGCNN). These models are tailored for processing 3D point cloud data, offering advanced feature extraction capabilities and adaptability to varied geometries. By focusing on segmentation during the initial phase of this work, a strong foundation is established for the subsequent task of crown generation, wherein the segmented data will serve as vital input for creating anatomically accurate and aesthetically pleasing crowns.

## 1.4 Importance of Segmentation

Segmentation is not merely a preparatory step but a fundamental enabler of precise crown generation. It facilitates the extraction of meaningful context, incorporating the prepared tooth, adjacent teeth, and surrounding structures to ensure a functional and well-fitted crown. High-quality segmentation reduces errors in subsequent stages, minimizing the need for manual adjustments and ensuring consistency throughout the automated workflow.

## 1.5 Thesis Structure

This thesis begins with an exploration of related work on segmentation and crown generation techniques, emphasizing challenges and opportunities in the field. The dataset employed in this study is then presented, detailing its characteristics and relevance to the objectives. Subsequently, the proposed methodology is outlined, covering model selection, dataset preparation, and training strategies. Experimental results are analyzed to evaluate segmentation performance, followed by a discussion on the implications, limitations, and potential avenues for improvement. The work concludes by providing an outlook for future efforts focused on crown generation.

# 2 Literature Review

## 2.1 Point Cloud-Based Models

### 2.1.1 Graph-Based Models

**PointNet** PointNet [6] is a pioneering deep learning architecture designed to directly process 3D point clouds without requiring transformation into structured formats (e.g., voxels or images). The model employs multilayer perceptrons (MLPs) and symmetric functions, such as max-pooling, to ensure permutation invariance and handle unordered input data. PointNet captures global features effectively but falls short in local feature extraction, as it lacks hierarchical learning mechanisms. Its key components include point-wise feature extraction, symmetric aggregation, and fully connected layers for classification or segmentation.

- **Strengths:** Direct processing of point clouds, simple yet efficient architecture, scalable to large datasets.
- **Limitations:** Limited capability to capture fine-grained local features and no explicit incorporation of neighbourhood information.

**PointNet++** PointNet++ [7] builds on the success of PointNet by introducing hierarchical feature learning. It employs set abstraction layers and sampling techniques to extract multi-scale local features while preserving the global context. Techniques such as farthest point sampling (FPS), grouping, and MLPs enhance the model's ability to capture fine geometric details, making it more suitable for complex 3D segmentation and classification tasks.

- **Strengths:** Improved local feature extraction through multi-scale and hierarchical learning.
- **Limitations:** Increased computational cost due to hierarchical feature grouping and sampling.

**Dynamic Graph CNN (DGCNN)** Dynamic Graph CNN (DGCNN) [8] introduces graph-based learning for point clouds by constructing a dynamic $k$-nearest neighbours (k-NN) graph in the feature space at each layer. The model employs edge convolutions (EdgeConv) to aggregate local geometric relationships and dynamically update the graph structure. This approach enables DGCNN to outperform PointNet [6] and PointNet++ [7] in tasks requiring rich local geometric feature learning.

- **Strengths:** Effective learning of local structures and spatial relationships via dynamic graph updates.
- **Limitations:** High computational demands for graph construction, particularly for large-scale point clouds.

### 2.1.2 Transformer-Based Models

**Point Cloud Transformer (PCT)** The Point Cloud Transformer (PCT) [4] adapts the self-attention mechanism from transformer architectures for 3D point cloud processing. Instead of relying on MLPs or convolutional operations, PCT uses attention-based modules to capture long-range dependencies and global point relationships. By replacing traditional feature extractors with transformers, PCT enhances the model's ability to learn complex spatial structures in point clouds.

- **Strengths:** Superior ability to capture global context, permutation invariance, and flexibility for classification and segmentation tasks.
- **Limitations:** Computationally expensive due to attention mechanisms, especially for large datasets.

**PoinTr** PoinTr [9] introduces a novel transformer-based framework designed for 3D point cloud completion. Unlike traditional methods, PoinTr effectively generates missing regions of incomplete point clouds by leveraging the global and local context captured through multi-head self-attention mechanisms. This model showcases significant advancements in tasks requiring point cloud reconstruction.

- **Strengths:** Strong ability to reconstruct missing regions, adaptability to various levels of point cloud incompleteness, and state-of-the-art performance in completion tasks.
- **Limitations:** High computational cost due to the use of transformer modules, especially for dense point clouds.

**From Mesh Completion to AI-Designed Crown** This recent work [5] highlights the application of deep learning in the dental domain, specifically for generating AI-designed dental crowns. The pipeline uses advanced mesh completion techniques combined with domain-specific priors to reconstruct missing tooth regions accurately. By leveraging both geometric and anatomical constraints, the system automates the generation of anatomically accurate and aesthetically pleasing crowns.

- **Strengths:** Domain-specific design tailored to dental applications, strong integration of mesh completion and anatomical knowledge, and practical utility in clinical workflows.
- **Limitations:** Dependence on high-quality input data and limited generalizability to non-dental applications.

## 2.2 Self-Supervised Learning: Contrastive Methods

Contrastive methods have emerged as a powerful paradigm in self-supervised learning, enabling models to learn meaningful representations by contrasting positive and negative pairs. Below, we review four prominent methods in this domain: SimCLR, BYOL, Barlow Twins, and VICReg.

**SimCLR** SimCLR [2] introduces a contrastive learning framework where positive pairs are created through data augmentations of the same sample, and negative pairs are sampled from other instances. The method optimizes a contrastive loss to maximize similarity between positive pairs while minimizing similarity between negative pairs.

- **Strengths:** Simplicity, strong performance across diverse datasets, and effective use of augmentations.
- **Limitations:** Dependence on large batch sizes and the need for numerous negative pairs to achieve optimal performance.

**BYOL (Bootstrap Your Own Latent)** BYOL [3] eliminates the need for negative pairs by employing a teacher-student framework. The student network learns by predicting the teacher's representations, which are updated via a moving average of the student's weights.

- **Strengths:** Removes reliance on negative pairs, simplifying training, and demonstrates robustness across diverse tasks.
- **Limitations:** Requires careful tuning of the moving average coefficient and sensitivity to hyperparameters.

**Barlow Twins** Barlow Twins [10] minimizes redundancy between learned features by aligning the cross-correlation matrix of representations to the identity matrix. This encourages decorrelation and maximizes information captured in each feature dimension.

- **Strengths:** Efficient training without the need for negative pairs and low sensitivity to batch size.
- **Limitations:** Limited exploration of augmentation diversity and lower performance on complex datasets compared to other methods.

**VICReg (Variance-Invariance-Covariance Regularization)** VICReg [1] builds on the principles of Barlow Twins by introducing variance, invariance, and covariance regularization terms. This approach enforces invariance across augmented views while encouraging diverse and uncorrelated representations.

- **Strengths:** Balances invariance and diversity, leading to robust representations, and eliminates the need for negative pairs.
- **Limitations:** Increased computational cost due to additional regularization terms.

**Summary of Contrastive Methods** These methods highlight the evolution of self-supervised learning, progressing from contrastive approaches relying on negative pairs (SimCLR) to more sophisticated methods that encourage diversity and invariance without negatives (BYOL, Barlow Twins, VICReg). Their application to 3D point cloud data presents an exciting avenue for future research in unsupervised representation learning.

# 3 Dataset

## Introduction

The dataset utilized in this project is **Teeth3DS**, first introduced at MICCAI 2022. It was developed through a collaboration between *Udini* (France), the *Inria Grenoble Morpheo team* (France), and the *Digital Research Center of Sfax* (Tunisia).

Teeth3DS comprises 1,800[1] 3D intra-oral scans of human teeth, categorized into lower and upper jaws. Each scan provides a detailed representation of dental structures, making the dataset valuable for applications such as dental crown generation, segmentation, and orthodontic analysis. The scans were collected using high-precision intra-oral scanners, ensuring accurate geometric and morphological details.

The dataset includes comprehensive annotations, such as:

- Labels for each point, enabling segmentation tasks.
- Metadata, such as anonymized patient information and jaw type.

In this project, we leveraged this dataset to analyze dental structures and perform 3D segmentation.

## 3.1 Properties of Point Sets in $\mathbb{R}^n$

In the context of processing point clouds, several inherent properties must be considered to develop robust mathematical models and neural networks capable of handling this data effectively. These properties ensure that the underlying structure and semantics of the point cloud data are preserved, enabling neural networks to approximate functions accurately. Below, we discuss these properties in detail and explain their relevance to neural network approximation.

### 3.1.1 Independent Features

Each point in a point cloud is defined by a set of features, typically its spatial coordinates $(x, y, z)$ and sometimes additional handcrafted attributes (e.g., color, intensity or normals). Importantly, these features are treated as independent variables.

**Why It Matters:** Linear Combination in Neural networks rely on the assumption that the features of each point are independent. This independence allows models to approximate complex functions by learning spatial relationships and transformations from raw point features.

**Relation to Neural Networks:**

- Models like PointNet [6] leverage this independence by applying shared weights to all points through symmetric operations, such as max pooling or summation. This ensures that the learned features remain invariant to point permutations.
- By treating features independently, neural networks can extract global patterns, but this can't efficiently capture local information as it processes each point independently.

[1]900 from upper jaw and 900 from lower jaw

### 3.1.2 Unordered Points

Unlike images or structured grids, point clouds are unordered. This means that swapping two points in a cloud does not alter the overall representation.

**Why It Matters:** Neural networks must handle this unordered nature to ensure that the output remains consistent regardless of the input order.

**Relation to Neural Networks:**

- PointNet introduced permutation invariance by applying symmetric functions (e.g., max pooling) to aggregate point features into a global representation.
- Subsequent architectures (e.g., PointNet++, DGCNN, PCT) refine this approach by incorporating local neighbourhood relationships while still respecting the unordered nature of point clouds.

### 3.1.3 Semantic Information (Global Context)

Point clouds capture global semantics, representing the overall structure of an object or scene. For example, the arrangement of points in a cloud defines whether the object is a lower jaw, upper jaw, a car, a chair, or a human.

**Why It Matters:** To approximate functions that describe or classify a point cloud, neural networks must extract global contextual information alongside local details.

**Relation to Neural Networks:**

- Aggregation layers in networks (e.g., max pooling or attention mechanisms) distill global semantic information by summarizing features across all points.
- Multi-scale architectures, such as PointNet++ and DGCNN, combine features at different scales to capture both fine-grained details and global semantics, which is crucial for tasks like segmentation and classification.

### 3.1.4 Fine Detail Structures (Local Context)

Point clouds often encode fine-grained details, such as the curvature of a surface or the sharpness of an edge. Capturing these details is essential for high-resolution tasks like 3D segmentation or shape reconstruction.

**Why It Matters:** Neural networks must preserve and leverage these local structures to approximate functions that depend on small-scale geometric features.

**Relation to Neural Networks:**

- Local operators, such as ball query or k-nearest neighbours (KNN), allow networks to define neighbourhoods and extract fine-grained geometric details.
- Graph-based approaches (e.g., DGCNN, PCT) extend this idea by dynamically connecting points based on learned features, enhancing the model's ability to approximate functions that depend on local geometric variations.

## 3.2 Connection to Neural Network Approximation

The properties of point clouds align closely with the Universal Approximation Theorem for neural networks, which states that a sufficiently large neural network can approximate any continuous function on a compact domain. However, these properties impose specific challenges that must be addressed for point cloud data:

1. **Invariant Function Approximation:** Neural networks must approximate functions that are invariant to point permutations (unordered points). Symmetric operations (e.g., max pooling, average pooling) ensure this invariance.

2. **Global Aggregation:** Networks must also approximate global functions that summarize the overall shape or structure of the point cloud (semantic information). Aggregation mechanisms like pooling or attention allow models to capture these global patterns.

3. **Localized Functionality:** For tasks requiring fine detail (e.g., reconstruction, completion parts), networks must approximate localized geometric details. Neighbourhood-based operations, such as KNN and graph convolutions, enable models to focus on small-scale geometric features.

By addressing these properties, neural networks designed for point clouds ensure accurate and efficient function approximation for this kind of data, enabling state-of-the-art performance in tasks like 3D classification, segmentation, and reconstruction.

## 3.3 Point Cloud Sampling

Sampling is a critical preprocessing step in working with point cloud data. It allows for efficient handling of high-resolution 3D models by reducing the number of points while preserving key geometric, semantic, and detailed information for reconstruction tasks. This reduction helps in computational efficiency without significantly compromising the quality of downstream tasks such as classification, segmentation, or reconstruction. Here are the primary sampling techniques used:

### 3.3.1 Farthest Point Sampling (FPS)

FPS iteratively selects points that are farthest from the already chosen points in the point cloud.

**Example** Sampling of sample in Figure 4 is in Figure 1 with 4096 point to be selected from 121337 point.

- **Why it's useful:**
  - Ensures a uniform distribution of sampled points across the entire point cloud.
  - Retains points that capture the geometric shape and boundaries effectively.
  - Commonly used in models like PointNet++, DGCNN, and PCT to extract hierarchical features.
- **Challenges:**

**Algorithm 1** Farthest Point Sampling (FPS)

```
Require: Point cloud vertices $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$, number of points to sample $m$
Ensure: Subset of sampled points $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m\}$
 1: Initialize $\mathbf{S} = \{\mathbf{s}_1\}$, where $\mathbf{s}_1$ is a random point from $\mathbf{P}$.
 2: Compute distances $\mathbf{d}_i = \infty \; \forall i \in \{1, 2, \ldots, n\}$ (initialize all distances to infinity).
 3: for $j = 2$ to $m$ do
 4:     for $\mathbf{p}_i \in \mathbf{P}$ do
 5:         Update $\mathbf{d}_i = \min(\mathbf{d}_i, \|\mathbf{p}_i - \mathbf{s}_{j-1}\|)$
 6:     end for
 7:     Select $\mathbf{s}_j = \arg\max(\mathbf{d})$
 8:     Add $\mathbf{s}_j$ to $\mathbf{S}$
 9: end for
10: return $\mathbf{S}$
```

  - Computationally expensive for large point clouds due to pairwise distance calculations.

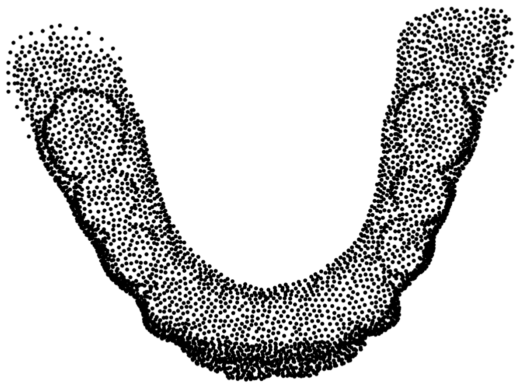

Figure 1: Farthest Point Sampling

### 3.3.2 Voxelization

Voxelization involves partitioning the 3D space into a grid of voxels (3D cubes) and replacing the points within each voxel with a representative point, such as the centroid.

- **Why it's useful:**
  - Reduces the number of points while maintaining overall geometric structure.
  - Facilitates efficient processing by converting irregular point clouds into a structured grid representation.
  - Useful for applications requiring grid-based computations, such as CNNs operating on 3D data.
- **Challenges:**
  - Loss of fine details if the voxel size is too large (trade-off).
  - Requires careful selection of voxel resolution to balance detail retention and computational efficiency.

**Algorithm 2** Voxelization Sampling

```
Require: Point cloud vertices $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$, voxel size $v$
Ensure: Subset of representative points $\mathbf{S}$
1: Initialize an empty voxel grid $G$.
2: for $\mathbf{p}_i \in \mathbf{P}$ do
3:     Compute voxel index for $\mathbf{p}_i$:
           $$\text{index} = \lfloor \mathbf{p}_i / v \rfloor$$
4:     if index $\notin G$ then
5:         Add $\mathbf{p}_i$ to $G$ as the representative point for the voxel.
6:     end if
7: end for
8: Collect representative points $\mathbf{S} = \{\text{representative points in } G\}$.
9: return $\mathbf{S}$
```

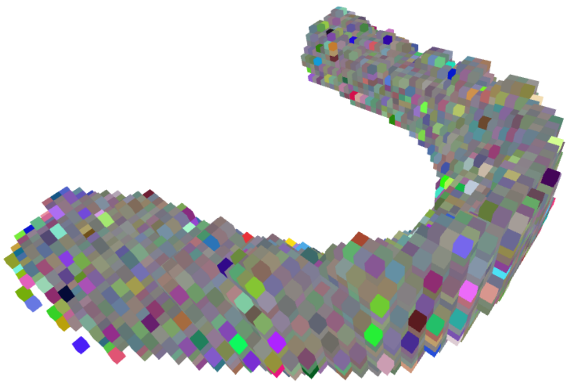

Figure 2: Grid size used here equal to 1

**Example** Sampling of sample in Figure 4 is in Figure 2 with Grid Size[2] equal 1.

### 3.3.3 K-Nearest Neighbors (KNN)

KNN identifies and retains a specified number of neighbouring points for each selected point based on Euclidean distance.

- **Why it's useful:**
  - Captures local neighbourhood information, which is essential for tasks requiring fine detail analysis, like reconstruction.
  - Helps in defining local regions for constructing graph-based models (e.g., DGCNN).
- **Challenges:**
  - Sensitive to the density of points in the cloud, as sparse regions might have less reliable neighbours.
  - Computationally intensive for very large point clouds.

[2]a more reasonable number is 0.25 or at most 0.5 but just for visualization I used grid size equal to 1

**Algorithm 3** K-Nearest Neighbors (K-NN) Sampling

```
Require: Point cloud vertices $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$, centroids $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_m\}$,
    number of neighbors $k$
Ensure: $k$-nearest neighbors for each centroid
 1: Initialize $\mathbf{N}_i = \emptyset\ \forall i \in \{1, 2, \ldots, m\}$.
 2: for $\mathbf{c}_i \in \mathbf{C}$ do
 3:     Compute distances $\|\mathbf{c}_i - \mathbf{p}_j\|\ \forall \mathbf{p}_j \in \mathbf{P}$.
 4:     Sort $\mathbf{P}$ by distance to $\mathbf{c}_i$.
 5:     Select the $k$ nearest points:
            $\mathbf{N}_i = \{\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \ldots, \mathbf{p}_{i_k}\}$
 6: end for
 7: return $\mathbf{N}_i\ \forall i$
```

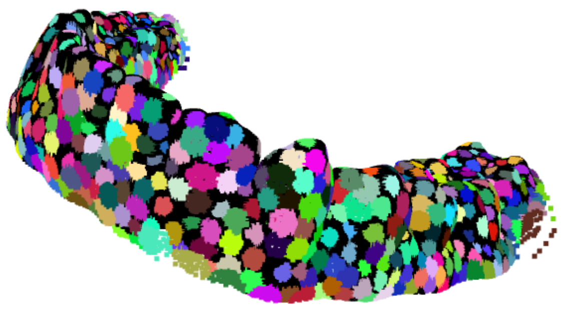

Figure 3: 1024 Centroids are selected using FPS algorithm and then for each point 64 points are selected as nearest neighbors.

**Example** Sampling of sample in Figure 4 is in Figure 3 with 1024 point to be selected from 121337 as a centroid using fps algorithm then using K-NN algorithm selected 64 Nearest points to each one of the centroid.

## 3.4 Summary of Complexities

| **Function** | **Complexity** |
|---|---|
| K-Nearest Neighbors (kd-tree) | $O(n \log n + m \log n + mk)$ |
| Farthest Point Sampling | $O(n \cdot m)$ |
| Voxel Grid Downsampling | $O(n + v)$ |

Table 1: Complexities of Sampling Techniques

**Key Variables:**

- $n$: Number of points in the 3D point cloud.
- $m$: Number of centroids.

- $k$: Number of neighbours per centroid.
- $v$: Number of voxels, dependent on the voxel grid size and voxel size.

## 3.5 Advantages of Combining Voxelization and FPS

1. **Reducing Computational Burden:** Voxelization significantly reduces the number of vertices in the initial point cloud by grouping points into voxels. This avoids the need to process all 117,650 points directly, which can be computationally expensive, especially for large datasets or real-time applications.
2. **Maintaining Uniformity:** FPS ensures that the downsampled points are spread out uniformly, preserving the overall structure of the object.
3. **Controlling the Number of Points:** By setting the voxels size[3] and later using FPS to sample a fixed number of points (e.g., 4,096), you ensure a consistent input size for the neural network.
4. **Avoiding Over-Simplification:** Using voxelization alone can't guarantee a fixed number of points and might lead to loss of important features if the voxel size is too large. By following it with FPS, you ensure that critical geometric details are retained.

## 3.6 Number of Vertices Through Each Step

| **Step** | **Approximate Number of Points** |
|---|---|
| Original Mesh | 117,650 |
| After Voxelization | 20,000 |
| After FPS | 4,096 |

Table 2: Number of Points Through Sampling Steps

This workflow effectively reduces the computational load while retaining essential geometric features. More work, including statistical analysis and benchmarking, will be conducted in the second semester.

[3]need to be fine-tuned to reduce the number of points in the point cloud to a reasonable number (e.g., 20,000)

# 4 Approach

## 4.1 Segmentation Models

Segmenting the 3D teeth models is a critical step in isolating relevant anatomical structures. The following segmentation architectures are utilized in this study: **Dynamic Graph Convolutional Neural Network (DGCNN)** and **Point Cloud Transformer (PCT)**.

### 4.1.1 Dynamic Graph Convolutional Neural Network (DGCNN)

DGCNN employs a dynamic graph structure to capture local and global geometric features. The key components of the DGCNN architecture include:

**Key Features:**

- **EdgeConv Operation:** Constructs a dynamic graph by connecting each point to its $k$-nearest neighbors (k-NN) in feature space. This operation encodes local geometric relationships and updates features by aggregating information from neighbouring points.
- **Dynamic Graph Updates:** The graph is updated at each layer based on feature space distances, allowing the model to adaptively learn relationships between points.
- **Global Feature Aggregation:** A global pooling operation aggregates features across all points, enabling the model to understand the overall shape and structure.

DGCNN achieves robust segmentation performance by effectively leveraging both local geometric structures and global contextual information.

### 4.1.2 Point Cloud Transformer (PCT)

PCT incorporates transformer-based mechanisms to capture long-range dependencies and global relationships within the point cloud. Its architecture is particularly well-suited for handling irregular and unordered data.

**Key Features:**

- **Self-Attention Mechanism:** Computes pairwise attention scores between all points to capture global dependencies and ensure permutation invariance.
- **Position Encoding:** Encodes relative positional information to guide the attention mechanism.
- **Multi-Head Attention:** Utilizes multiple attention heads to model diverse feature interactions, enhancing expressiveness.
- **Integration of Local and Global Features:** Combines local geometric features with global context, enabling fine-grained segmentation.

By leveraging the strengths of transformers, PCT excels in capturing the complex spatial structures of point clouds, achieving great performance in segmentation tasks.

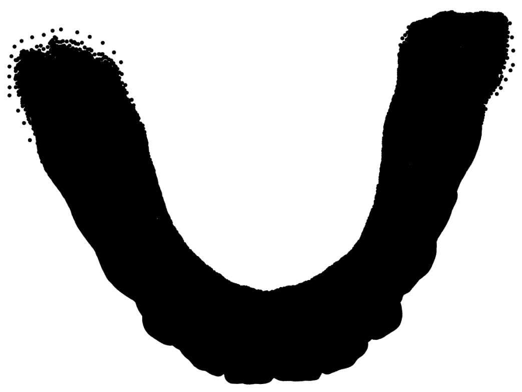
Figure 4: Canonical Space of our dataset

## 4.2 Rigid Transformation and Fixed Canonical Space

To ensure invariance to transformations such as rotation and translation, we apply rigid transformations to map the 3D point clouds into a fixed canonical space[4]. This alignment standardizes the orientation and positioning of point clouds, allowing the models to focus on learning shape-specific features. The two main techniques used are: Self-Supervised Techniques and PCA.

### 4.2.1 Rigid Transformation

Rigid transformations involve applying rotation and translation operations to align point clouds in a common coordinate system while preserving their relative structure. This spatial normalization simplifies comparisons and downstream processing.

### 4.2.2 Fixed Canonical Space

After transformation, all point clouds are mapped into a fixed canonical space as in Figure 4. This provides a unified reference frame, reducing ambiguity introduced by arbitrary orientations in raw data.

## 4.3 Self-Supervised Techniques

Self-supervised learning is employed to learn meaningful representations of point clouds without requiring labelled data. These approaches rely on intrinsic geometric properties to generate supervisory signals.

**Core Concept:** The dataset resides in a fixed canonical space. Random rigid transformations are applied to each model, and a neural network, such as the T-Net introduced in PointNet [6], predicts a $3 \times 3$ transformation matrix Figure 5. The transformed model is compared with the original using a loss function to ensure alignment with the canonical space. Centering the model before applying transformations is crucial to avoid unintended distortions like shearing[5].

[4]canonical just means standard basis vectors. and whenever we say *canonical space* we means by that *canonical orientation*

[5]wich is not a rigid transformation

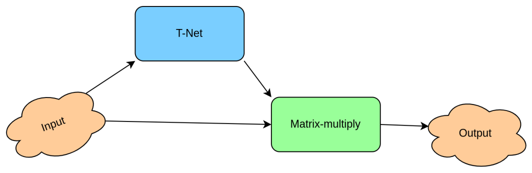

Figure 5: Inference: T-Net architecture for generating a transformation matrix.

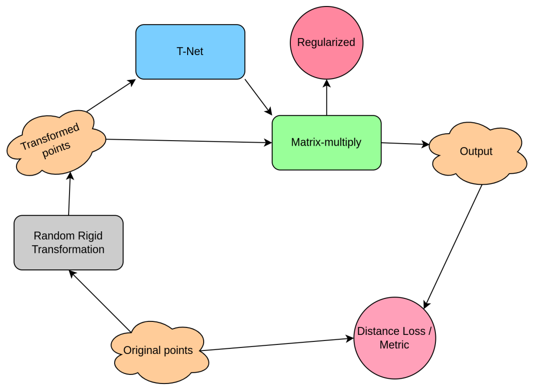

Figure 6: Training

Figure 7: Self-supervised learning: Contrastive method

**Key Strategies:**

- **Contrastive Learning:** Learns invariant representations by contrasting augmented versions of the same point cloud Figure 6.
- **Rotation Prediction:** Models predict rotation applied to a point cloud, enabling them to learn geometric orientation cues effectively.

## 4.4 Principal Component Analysis (PCA)

PCA is used to align point clouds to a fixed canonical orientation. The steps include:

1. Compute the covariance matrix of the point cloud's vertices.
2. Perform eigen decomposition to obtain eigenvalues and eigenvectors.
3. Align the principal axes of the point cloud with global coordinate axes.

While PCA provides a reliable alignment mechanism, the ambiguity in eigenvector directions presents a challenge that requires further resolution. Detailed Algorithm 4

```
Algorithm 4 Principal Component Analysis (PCA)
Require: Point cloud vertices $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$ where $\mathbf{p}_i \in \mathbb{R}^3$
Ensure: Aligned point cloud with principal axes oriented to global coordinate axes
1: Step 1: Compute the covariance matrix
     • Calculate the centroid of the point cloud:
           $$\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i$$
     • Subtract the centroid from each point to center the point cloud:
           $$\mathbf{P}_{\text{centered}} = \{\mathbf{p}_i - \mathbf{c} \mid \mathbf{p}_i \in \mathbf{P}\}$$
     • Compute the covariance matrix:
           $$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{p}_i - \mathbf{c})(\mathbf{p}_i - \mathbf{c})^\top$$
2: Step 2: Perform eigen decomposition
     • Compute the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ and eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ of $\mathbf{C}$ such that:
           $$\mathbf{C}\mathbf{v}_j = \lambda_j \mathbf{v}_j \quad \forall j \in \{1, 2, 3\}$$
     • Sort eigenvalues in descending order and arrange corresponding eigenvectors accordingly.
3: Step 3: Align the principal axes
     • Transform the point cloud to align the principal axes with the global coordinate axes:
           $$\mathbf{P}_{\text{aligned}} = \mathbf{P}_{\text{centered}}\mathbf{V}^\top$$
       where $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3]^\top$ is the matrix of eigenvectors.
4: Step 4: Resolve eigenvector ambiguity (if necessary)
     • Ensure consistent eigenvector directions to handle ambiguity by applying domain-specific constraints or heuristics.
```

# 5 Experiments

This chapter presents the experiments conducted to evaluate the performance of segmentation models under various conditions. The experiments are divided into three main parts, with each part analyzing the robustness of the models and the impact of preprocessing methods.

## 5.1 Experimental Setup

All experiments were conducted on **Kaggle's cloud-based platform**, utilizing their free GPU resources. The hardware and software specifications for the experiments are as follows:

- **GPU:** NVIDIA Tesla P100 with 16 GB VRAM.
- **Python Version:** 3.10.14.
- **PyTorch Version:** 2.4.0.
- **CUDA Version:** 12.3.
- **RAM:** 29 GB.
- **Disk Space:** 19.5 GB (temporary storage for datasets and outputs).
- **Session Time Limit:** 12 hours.

The availability of the Tesla P100 GPU enabled efficient training of both DGCNN and PCT models, with reasonable training times even for complex architectures.

In all experiments, we evaluate the models using both quantitative and qualitative metrics. The results are analyzed using overall accuracy, mean accuracy per class, mean Intersection over Union (mIoU), and qualitative visualizations.

We didn't use a set of files as a validation set from the dataset, so we didn't fine-tune any hyperparameter, but we just evaluated the performance on the test set after each epoch **but we didn't use it to fine-tune or choose anything**[6].

## 5.2 Experiments Without Transformation

This section evaluates the performance of the segmentation models without applying any rigid transformations to the input data.

### 5.2.1 DGCNN

The DGCNN model performed well when no transformations were applied. The results indicate that if the input point clouds are aligned in a fixed canonical space, DGCNN can effectively segment the data.

**Training and Testing Results** Figures 10 13

[6] I am emphasizing about this because some people may think that we used the test set as a validation set because we plotted it with the training set after each epoch and this is not true, we did this just for visualization and simply you can just ignore the test curve and just look at the performance at the last epoch

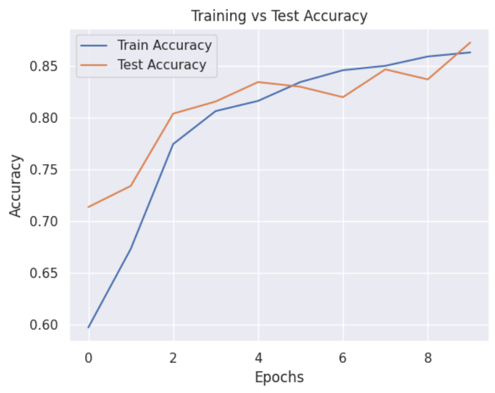

Figure 8: Accuracy for DGCNN without transformation.

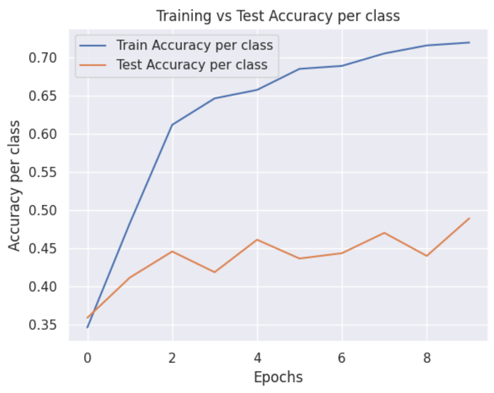

Figure 9: Class-wise accuracy for DGCNN without transformation.

Figure 10: Accuracy metrics

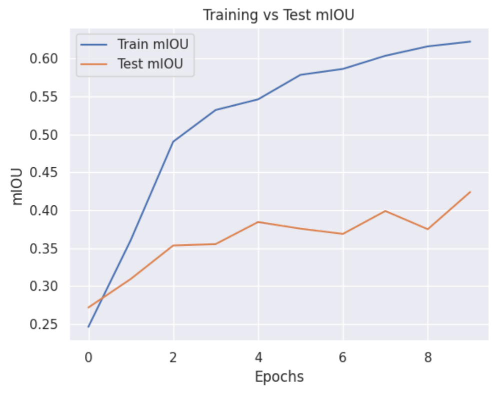

Figure 11: Mean Intersection over Union (mIOU) for DGCNN without transformation.

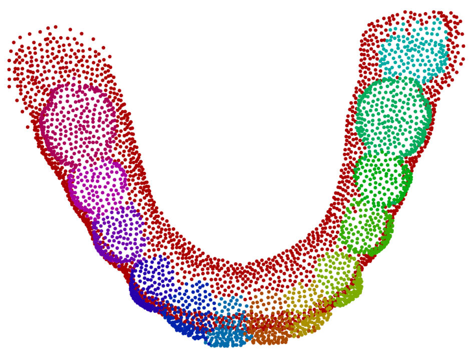

Figure 12: Segmentation output -one sample in the test set- for DGCNN without transformation.

Figure 13: metrics and sample

#### 5.2.2 PCT

The Point Cloud Transformer (PCT) exhibited strong performance in this setting, demonstrating its capacity to process point clouds aligned in a fixed space.

**Training and Testing Results** Figures 16 19

### 5.3 Experiments With Random Rigid Transformations

This section evaluates the robustness of the models when random rigid transformations (rotations and translations) are applied to the input data.

#### 5.3.1 DGCNN

The DGCNN model showed reduced performance when random transformations were applied. The reliance on the T-Net for alignment resulted in suboptimal segmentation.

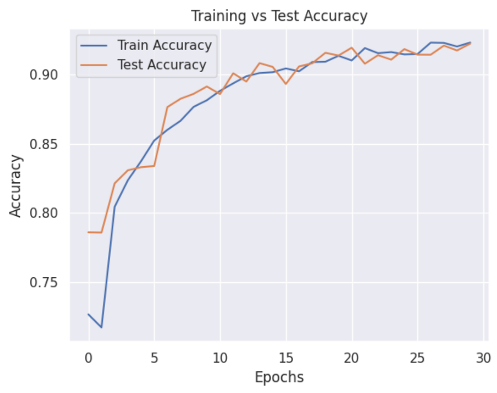

Figure 14: Accuracy for PCT without transformation.

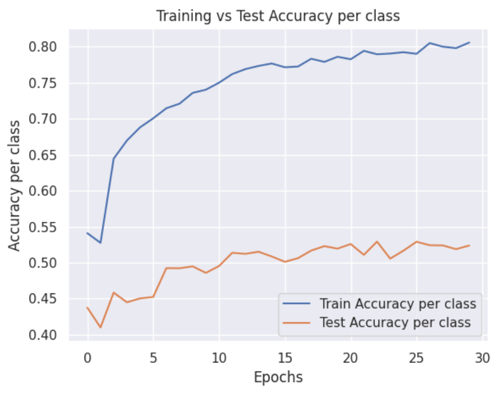

Figure 15: Class-wise accuracy for PCT without transformation.

Figure 16: Accuracy metrics

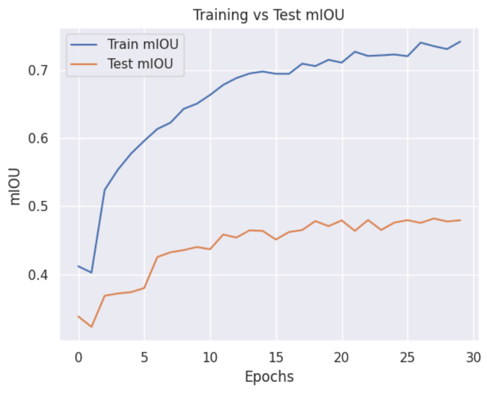

Figure 17: Mean Intersection over Union (mIoU) for PCT without transformation.

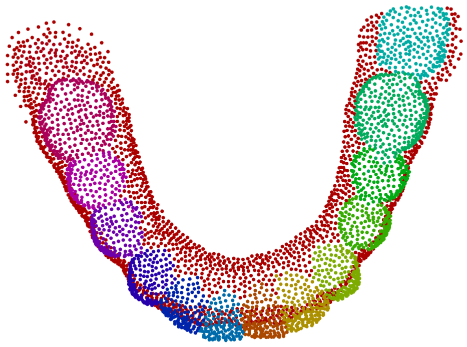

Figure 18: Segmentation output for PCT without transformation.

Figure 19: metrics and sample

**Training and Testing Results** Figures 22 23 26

### 5.3.2 PCT

The PCT model demonstrated resilience to rigid transformations due to its attention mechanism, which effectively captures global relationships in the point cloud.

**Training and Testing Results** Figures 29 30 33

## 5.4 Canonical Space Prediction

This section investigates methods to decouple canonical space prediction from segmentation. Two approaches are explored: **Self-Supervised Learning** and **Principal Component Analysis (PCA)**.

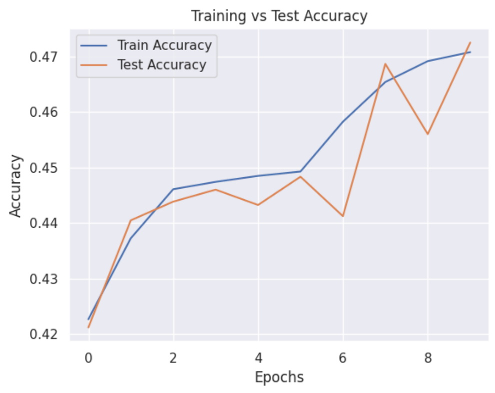

Figure 20: Accuracy for DGCNN with random transformation.

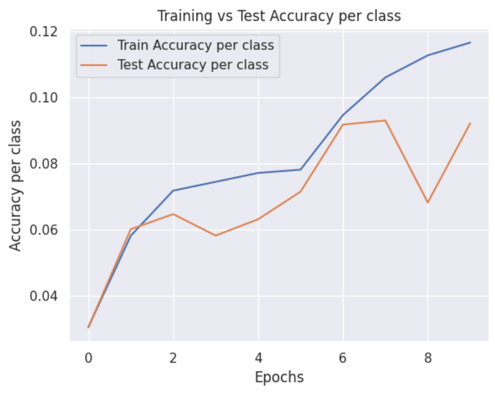

Figure 21: Class-wise accuracy for DGCNN with random transformation.

Figure 22: Accuracy metrics

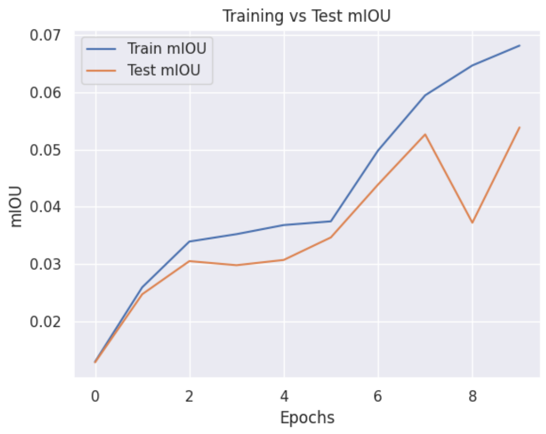

Figure 23: Mean Intersection over Union (mIOU) for DGCNN with random transformation.

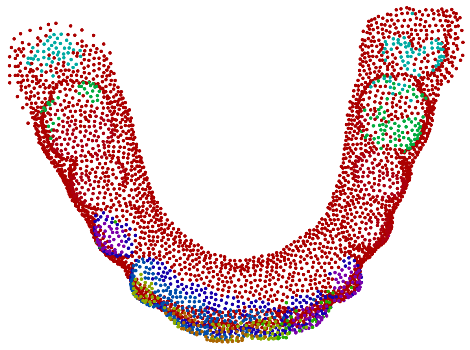

Figure 24: test without random transformation (Identity matrix).

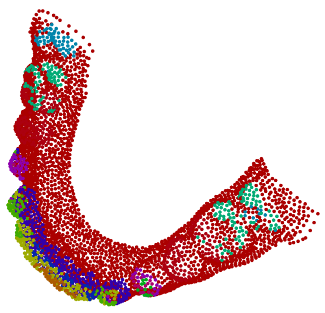

Figure 25: random transformation 1.

Figure 26: Segmentation output for DGCNN with two different random transformations.

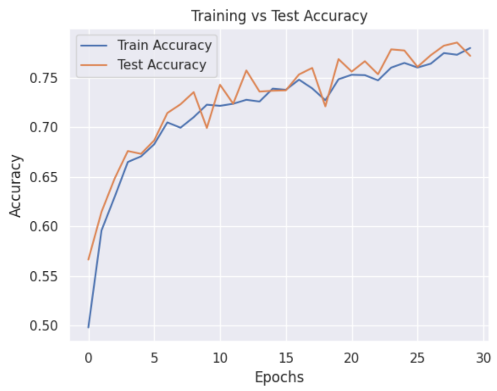

Figure 27: Accuracy for PCT with random transformation.

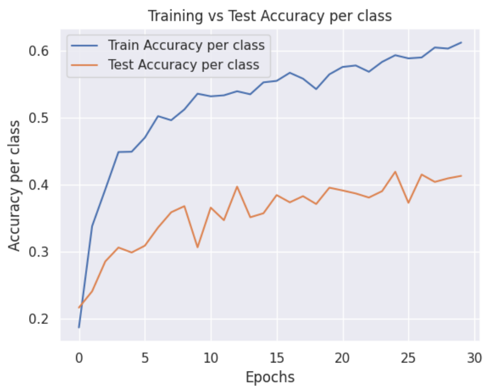

Figure 28: Class-wise accuracy for PCT with random transformation.

Figure 29: Accuracy metrics

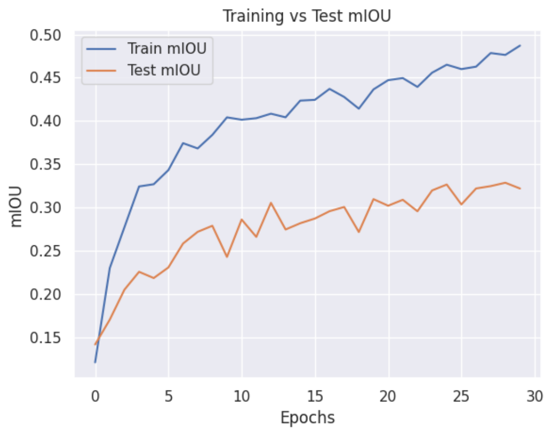

Figure 30: Mean Intersection over Union (mIoU) for PCT with random transformation.

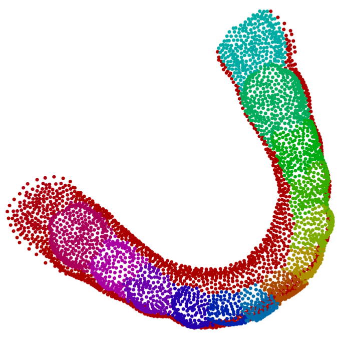

Figure 31: random transformation 1.

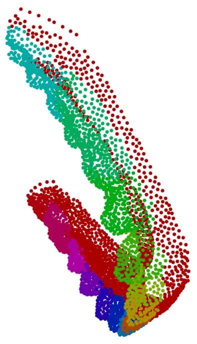

Figure 32: random transformation 2.

Figure 33: Segmentation output for PCT with two different random transformations.

### 5.4.1 Self-Supervised Learning

Contrastive learning was employed to train a neural network (T-Net) to predict a transformation matrix. Due to the simplicity of the T-Net, the results are bad so we need to try another architecture includes some *EdgeConv* layers rather than that existing in the PointNet paper.

### 5.4.2 PCA

PCA demonstrated effectiveness in aligning point clouds to a canonical orientation, although issues with eigenvector direction ambiguity were observed.

**Qualitative Results** Figures 36 39

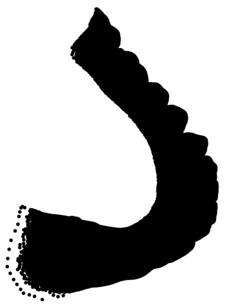
Figure 34: point cloud input (random transformation 1)

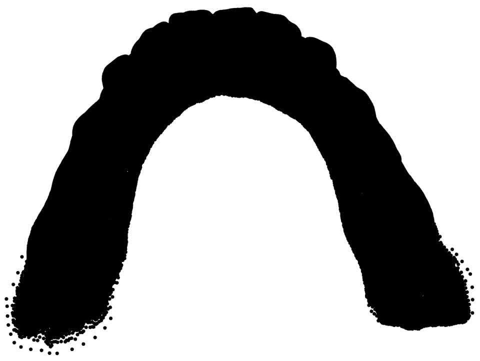
Figure 35: PCA-aligned point cloud output

Figure 36: PCA: input-output

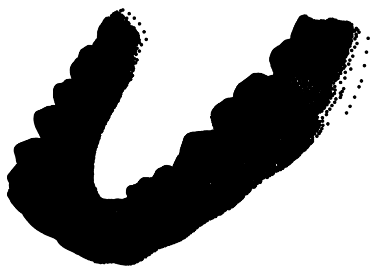
Figure 37: point cloud input (random transformation 1)

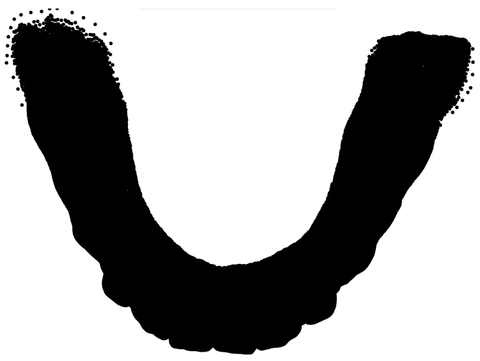
Figure 38: PCA-aligned point cloud output

Figure 39: PCA: input-output

# 6 Discussion and Conclusion

In this thesis, we have explored the application of advanced deep learning models, specifically the Dynamic Graph Convolutional Neural Network (DGCNN) and Point Cloud Transformer (PCT), for the segmentation of dental structures in 3D point clouds. Our experiments on the Teeth3DS dataset demonstrated that both DGCNN and PCT are capable of handling complex geometries and variations inherent in dental scans.

The results indicate that when the input data is aligned in a fixed canonical space, both models perform well, achieving high accuracy and mean Intersection over Union (mIoU). However, the introduction of random rigid transformations significantly affected the performance of DGCNN due to its reliance on T-Net for alignment, highlighting a limitation in its ability to generalize under varying transformations. On the other hand, PCT showed greater resilience to these transformations, attributed to its offset-attention mechanism that effectively captures global relationships within the point cloud.

One notable challenge encountered during our work was the suboptimal performance of the T-Net used for canonical space prediction. Also when training this T-Net with a contrastive method Figure 6, the performance didn't improve much. The simplicity of the T-Net architecture led to poor alignment results, suggesting the need for more sophisticated architectures incorporating additional layers such as EdgeConv to improve its effectiveness as the existing architecture processes each point alone with MLP-shared and then it's followed with max function (symmetric aggregation function) and this doesn't work for our complex point cloud. Principal Component Analysis (PCA) proved effective in aligning point clouds but faced issues with eigenvector direction ambiguity, which requires further refinement.

Despite these challenges, the developed segmentation framework lays a strong foundation for subsequent tasks in crown generation. By isolating the tooth and its surrounding context accurately, the segmented data can serve as crucial input for creating anatomically accurate and aesthetically pleasing crowns. This work thus represents a significant step towards end-to-end automation in dental crown design, ultimately enhancing the accuracy and efficiency of dental restorations.

# 7 Future Work

While the current study has made substantial progress in automating dental crown generation through segmentation, several avenues remain for future exploration:

## 7.1 Model Enhancement

Further improvements can be made by refining existing models or developing hybrid architectures that combine the strengths of DGCNN and PCT. Incorporating additional EdgeConv layers into the T-Net could enhance its alignment capabilities, enabling better generalization under random transformations. Additionally, exploring alternative self-supervised learning methods like Barlow Twins or VICReg may yield improved canonical space predictions.

## 7.2 Data Augmentation and Diversity

Expanding the dataset beyond Teeth3DS to include scans from diverse populations and dental conditions would help test the generalizability of the proposed models. Introducing more varied augmentations during training could also enhance model robustness, ensuring consistent performance across different scenarios.

## 7.3 Crown Generation Pipeline

The next critical phase involves leveraging the segmented data to generate dental crowns. Developing or integrating existing crown generation techniques will enable a comprehensive evaluation of the entire workflow. This includes assessing the anatomical accuracy, aesthetic quality, and practical usability of the generated crowns in clinical settings.

## 7.4 Real-Time Applications

Investigating how the segmentation and crown generation pipeline can be adapted for real-time applications is another promising direction. Real-time processing would significantly reduce turnaround times in dental practices, making the technology more accessible and practical for widespread adoption.

## 7.5 Explainable AI and Adaptive Learning

Integrating explainable AI mechanisms into the models would provide insights into their decision-making processes, building trust among dental professionals. Furthermore, exploring adaptive learning frameworks where the models continuously improve based on new data and user feedback could ensure they stay up-to-date with evolving dental practices.

In conclusion, while this work has established a solid foundation for automated dental crown generation, ongoing research and development hold the promise of further advancements, ultimately transforming the landscape of restorative dentistry.

## References

[1] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations.

[3] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning.

[4] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. PCT: Point cloud transformer. 7(2):187–199.

[5] Golriz Hosseinimanesh, Farnoosh Ghadiri, Francois Guibault, Farida Cheriet, and Julia Keren. From mesh completion to AI designed crown. In Hayit Greenspan, Anant Madabhushi, Parvin Mousavi, Septimiu Salcudean, James Duncan, Tanveer Syeda-Mahmood, and Russell Taylor, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2023*, volume 14228, pages 555–565. Springer Nature Switzerland.

[6] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation.

[7] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space.

[8] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds.

[9] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. PoinTr: Diverse point cloud completion with geometry-aware transformers.

[10] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction.