university of groningen

faculty of science and engineering

Self-Supervised Representation Learning in Point Clouds for Hierarchical Graph-Based Anatomical Structure Identification

Niels Rocholl

nedap

university of groningen / faculty of science and engineering

University of Groningen

Self-Supervised Representation Learning in Point Clouds for Hierarchical Graph-Based Anatomical Structure Identification

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at University of Groningen under the supervision of
Dr. Matthia Sabatelli (Artificial Intelligence, University of Groningen)
and
Dr. Thomas Markus (Nedap Healthcare)

Niels Rocholl (s3501108)

August 30, 2024

Contents

Acknowledgments

During this project I was primarily supported by two supervisors, Dr. Matthia Sabatelli from the University of Groningen, and Dr. Thomas Markus from Nedap Healthcare. I want to thank Dr. Matthia Sabatelli for teaching me to think critically about research results and to always scrutinize the validity of scientific statements. Our meetings were always fruitful and intellectually stimulating, consistently encouraging me to think from a scientist's perspective. I also want to thank Dr. Thomas Markus for the many enjoyable conversations where we could explore and get lost in ideas that delved deeply into the technicalities of AI. From Thomas, I learned that a creative mind can find a solution to almost any problem. I have gained a great deal of both technical and scientific knowledge from our discussions. I want to thank Nedap Healthcare for this research opportunity and the exceptional resources I was provided. Especially, the access to two H100 GPUs has enabled me to experiment with and train advanced models on large datasets. This thesis would not have been possible without these resources.

Abstract

This thesis presents **G**raph **R**epresentation of **A**dvanced **P**art **E**ncodings (GRAPE), a novel framework for anatomical structure identification in three-dimensional point clouds of human bodies. By combining self-supervised learning and graph neural networks, GRAPE constructs graph-based representations of 3D objects, where each node is enriched with a latent feature derived from their corresponding object parts. These features, generated by a Masked Autoencoder (Point-MAE), capture geometrical and spatial information within the point cloud.

GRAPE-GNN, the model developed using this framework, leverages both the node embeddings (latent Point-MAE features) and the topological information within the hierarchical graph created by GRAPE for accurate identification of anatomical structures. Evaluated on our Anatomy-GRAPE dataset, which was specially annotated for this thesis and validated by medical experts, GRAPE-GNN demonstrates high precision and recall across various anatomical structures. Our experiments show that GRAPE-GNN retains good performance in scenarios with limited annotated training data. We identify areas for improvement, such as enhancing performance under point cloud occlusion. This method not only demonstrates the effectiveness of graph-based anatomical structure identification but also the potential of integrating self-supervised learning based point cloud features with graph-based representations in general.
[1]

[1]Code available at GitHub - nedap/GRAPE

1 Introduction

This thesis proposes a novel approach for the identification of anatomical structures in three-dimensional point cloud representations of the human body. The methodology primarily focuses on the recognition of muscular structures, as illustrated in Figure 1, but possesses the potential for extension to skeletal elements and other anatomical components. A high-level overview of the main steps of the anatomical structure identification pipeline is shown in Figure 2. This figure highlights three distinct steps. Step one involves patch generation and encoding, where the point cloud is divided into patches and encoded using a model that has been pre-trained in a self-supervised fashion. Step two is graph generation, where these encoded patches are integrated into a predefined anatomical graph. Step three involves structure identification using a graph neural network, which classifies the nodes in the graph to identify anatomical structures. The following section provides an introduction to the core concepts underpinning this research. Section 1.1 explores the idea of anatomical structures and elaborates on anatomical structure identification. Section 1.2 introduces self-supervised learning of point cloud spatial features. Lastly, Section 1.3 introduces graph-based point cloud representations and the subsequent prediction of nodes. Each section introduces one research question centered around the section's respective concept.

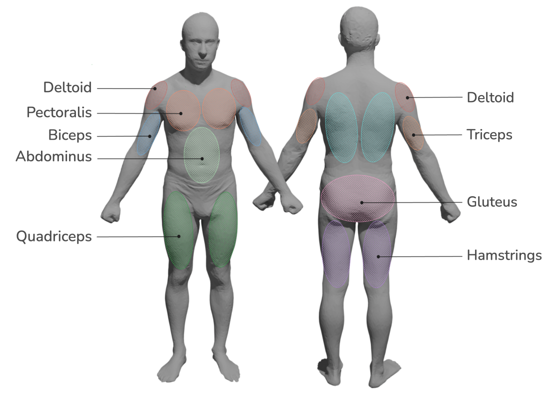


Figure 1: Areas that roughly correspond to representative anatomical structures in a 3D model of a male body.

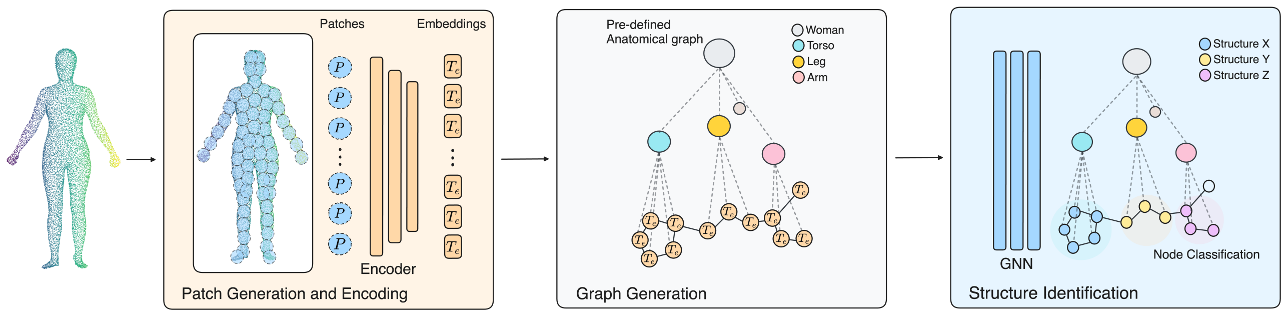


Figure 2: Overview of the Anatomical Structure Identification Pipeline.

1.1 Anatomical Structure Identification

Accurate identification and communication of anatomical structures are crucial in the medical field. When communicating medical conditions, doctors need to be exceptionally precise in referring to the affected location. Using exact anatomical terms instead of general descriptions ensures clarity and avoids ambiguity. The term *anatomical structure* is used in the medical field and literature to refer to any distinct and identifiable part of the human body [1]. The importance of identifying anatomical structures becomes clear when considering practical cases. Think for example of radiology, where radiologists must find and identify precise anatomical locations to diagnose tumors, fractures, or infections. Similarly, in physical therapy, knowing the exact muscle or tendon is vital in deciding the correct treatment. Finally, in general practice, pinpointing the exact location is essential for under-

standing the severity of symptoms. Pain in one area might be harmless, while the same pain just a few centimeters deeper might indicate a serious issue.
Precise anatomical identification requires highly trained medical experts who are already in short supply globally [2, 3, 4]. Additionally, determining the correct structure or location can be time-consuming, worsening the problem during a workforce shortage. This situation provides a prime opportunity for Artificial Intelligence (AI) to alleviate pressure on the workforce. By effectively using AI for tasks where it genuinely enhances quality and efficiency, we cannot only reduce the pressure on the healthcare system but also improve the quality of care delivered to patients. This leads to the central objective of this thesis, which is dedicated to exploring the potential of AI for accurate and useful anatomical structure identification. The first research question serves to answer this objective:

Q1. Can machine learning be used for accurate and robust anatomical structure identification in humans based on exterior surface data alone?

To achieve this objective we have to design a system, that on a high level works similar to a doctor identifying an anatomical structure based on a patient's visual indications. With Q1 we aim to determine whether it is possible to make a system that is able to predict the anatomical structures underlying a specific location on the body only using exterior surface data. Exterior surface data means that the model is only provided with an exterior view of the human during inference, with no CT, MRI, or other type of imaging that can provide an internal view. Additionally, we evaluate whether the system can achieve high precision and recall, and be robust to partial occlusion.
Using images as input for this proposed model might seem the default and obvious choice. However, we argue that using three-dimensional input data, such as point clouds, will provide valuable benefits, relating to geometry and spatial relationships. This approach also addresses issues like skin tone biases and variations in lighting conditions, assuming near-perfect point cloud extraction and the use of colorless point clouds. Additionally, many locations on the body can be hard to discriminate due to the visual uniformity of the skin. Local and global spatial and geometrical cues can potentially provide beneficial information allowing for a better understanding of the precise body locations. Recent advancements in point cloud processing have demonstrated the potential of capturing these geometric and spatial features via self-supervised pre-training approaches like Point-MAE and Point-M2AE [5, 6]. These models are able to learn high-quality latent representations from unlabelled point cloud data. These representations can in turn be used for downstream tasks such as anatomical structure identification by fine-tuning on a smaller task-specific labelled dataset.
The next section will introduce the concept of self-supervised learning, and discuss self-supervised learning of point cloud spatial Features.

1.2 Self-Supervised Learning of Point Cloud Spatial Features

The paradigm of using labeled data to train AI systems is referred to as supervised learning. This method of training has produced numerous high-quality specialized models. However, supervised learning comes with an inherent bottleneck, which becomes especially evident when considering its application to the medical domain. A major challenge when applying supervised learning to the medical domain is the lack of high-quality large-scale annotated data [7, 8]. Medical data often requires highly specialized professionals for reliable annotation. An example is the need for radiologists to label tumors in CT scans. For those not well-informed on AI, it can be hard to justify radiologists spending weeks on data labeling, given their critical role in direct patient care. Furthermore, compared to the hourly rate of annotators in other domains, such as general image recognition or natural language processing, specialized medical experts are extremely expensive, making it even harder to find sufficient resources to create such a dataset. Self-Supervised Learning (SSL) is a paradigm that addresses this data bottleneck. SSL defines a learning objective centered around the inherent structure of the data, allowing models to learn from orders of magnitude more data. Recent work also provides evidence that SSL models can learn representations that are more robust to adversarial examples, label corruption, and input perturbations and are more fair compared to their supervised counterparts [9, 10]. These advantages make SSL a promising approach for the pre-training phase of the anatomical structure identification pipeline, addressing the issue of labeled data scarcity while simultaneously providing the aforementioned benefits of robust and fair representations. Q2 is dedicated to exploring the effectiveness of using SSL to produce the envisioned representations:

Q2. Can self-supervised learning models accurately capture and represent fine-grained geometric features and spatial relationships of the human body based on surface-level point cloud data?

With Q2, we aim to determine if an SSL model can learn representations that capture the geometric details of the human body. These details include features such as the shape of a hand, arm, or leg. Additionally, we aim to determine if an SSL model can capture spatial relationships. On a local level, certain areas across the human body can be very similar, but the relative position between these areas can be informative. Transformer-based SSL models such as Point-MAE [6] have demonstrated potential in learning complex geometric features and spatial relationships within point cloud data. The effectiveness of this model for self-supervised learning of point cloud spatial features will be explored in detail in this thesis.
SSL serves as the pretraining step in the anatomical structure identification pipeline, and it is followed by a supervised graph-based fine-tuning step for the downstream structure identification task. Section 1.3 will introduce the concept of graph-based anatomical structure identification. This approach aims to integrate the spatial and geometrical features from the SSL step and the relational and hierarchical characteristics of the human anatomy, for accurate and robust structure identification.

1.3 Graph-Based Representation and Identification

The human anatomy is extremely complex, consisting not of a single two-dimensional layer, but rather an intricate three-dimensional structure composed of multiple interconnected layers and systems. These layers range from surface-level skin and muscles to deeper organs and skeletal structures. A model that only uses exterior surface data has no information about this interior anatomical structure, making the predictive task of the model extremely challenging. Even though we do not have direct access to the internals of the human body, we can make assumptions about the human anatomy that can potentially be highly useful for the final anatomical structure identification model. Under the assumption of a standardized anatomical model, we can disregard individual anatomical variations, including sex-based differences and congenital or acquired abnormalities. Versions of such a model have already been developed in different form factors. Examples of such models include SNOWMED-CT [11], and Elseviers Complete Anatomy [12]. Taking inspiration from SNOWMED-CT, anatomy can naturally be expressed as a graph, containing relational and hierarchical information between anatomical structures. Working under the assumption that this information is static and consistent across humans, it can be used to inform the anatomical structure identification model about the human interior. Graphs come in the form of graph-structured data, which is different from fixed-size and/or regular-structured data such as images or videos. Therefore, specialized algorithms that can understand and learn from graph-structured data are needed. A promising branch of AI that is focused on this type of data is Graph Neural Networks (GNNs). Instead of considering each individual entity in isolation, GNNs use of the underlying graph to make informed predictions. The final research question is dedicated to exploring the benefit of introducing graph-based representations into our anatomical structure identification system.

Q3. Can the integration of graph-based representations enhance the accuracy of the anatomical structure identification?

With Q3, we aim to determine if we can effectively combine the spatial and geometrical features generated by the SSL model from the first step, with the relational and hierarchical characteristics of the human anatomy. By modeling this information as a graph, we intend to exploit both types of information using a GNN.

1.4 Thesis Outline

This thesis is organized into six chapters. Chapter 1 introduced the core concepts underpinning this research. Chapter 2 presents a review of three areas: machine learning in medical imaging and anatomy, self-supervised learning, and graph neural networks. The first section on medical imaging and anatomy is intentionally general and is intended to explore a wide-ranging set of techniques that could potentially serve as inspiration for the methodology of this research. The section on SSL explores various self-supervised learning methods, aimed at informing the reader how different self-supervised learning methods can be used to learn rich representations from unlabeled data. The third section explores graph neural networks, a type of neural network that can naturally operate on graph-structured data. Concepts such as message passing are discussed, and popular methods for updating node embeddings are presented. In Chapter 3, the methodology is presented in four sections. The first section covers the datasets used to train and validate the methodology. The second section describes the self-supervised learning architecture. The third section explains the graph construction process. Finally, in the fourth section, the custom GNN for structure identification is presented. In Chapter 4 the experimental setup is detailed, and five main experiments are introduced. Chapter 5 presents the experimental results for these five experiments. Lastly, Chapter 6 presents the discussion, conclusion, and ends with a note on future work.

2 Literature Review

Many medical imaging techniques have been invented to provide immense benefits to patients and doctors, resulting in a diverse range of modalities such as functional magnetic resonance imaging, computed tomography, and X-rays. AI has positioned itself as one of the most promising paradigms in medical imaging research [13]. Each modality requires specialized machine learning models that have been trained on modality-specific data. For tasks such as tumor segmentation or nodule detection, there has already been a substantial amount of foundational research to build upon [14, 15]. The objective of this project is less established, and despite extensive research, we were unable to find well-established methods that address the identification of anatomical structures using only exterior point cloud data. Finding the final methodology requires a wide-ranging review of various approaches found not only in AI-based medical imaging but AI in general. This general review is aimed at identifying techniques and methodologies that could be transferred to this project's specific application. Section 2.1 introduces various (anatomical) landmark detection techniques, a popular problem that is related but distinct from anatomical structure identification, focusing on the detection of singular points instead of regions corresponding to structures hidden inside the body. Additionally, the section introduces notable work on medical image segmentation. Section 2.2 reviews the concept of SSL and details the four families of SSL, finishing with masked modeling, the SSL family central to the pre-training stage described in this thesis. Lastly, Section 2.3 reviews the field of graph neural networks, starting with foundational concepts such as message passing, and ending with prominent architectures such as GraphSAGE.

2.1 Machine Learning for Medical Imaging and Anatomical Tasks

Localization and identification of anatomical landmarks, though related to anatomical structure identification, focus on specific points rather than comprehensive structures. These landmarks are critical within medical imaging as they serve as important reference points when planning surgery or radiation therapy [16, 17]. But even outside medicine, facial landmark localization is essential for tasks such as biometric authentication or emotion recognition. Early work on facial landmark detection, which tries to detect anatomical landmarks in the human face, focused mainly on regression and template fitting methods [18, 19, 20]. With the advent of deep learning, Convolutional Neural Networks (CNN) became a powerful tool to effectively capture spatial hierarchies and patterns in images through convolutional layers. Work by Sun et al. [21] uses multiple CNN-based networks to capture complex features, subsequently fusing their output in a cascaded fashion. These networks aim to predict the location of the landmarks, essentially performing a regression task aided by the CNN. Later work aimed to address the shortcomings of traditional regression and template fitting methods for the problem of facial landmark detection. Particularly regarding occlusion and pose variation, through a deep learning system based on a variation of multi-task learning [22]. In contrast to traditional multi-task learning, where the goal is to maximize the performance over all tasks, Zhang et al. aimed to maximize the main task of facial landmark detection r, with auxiliary tasks $a \in A$. They formulate their learning objective to:

$$\underset{\mathbf{W}^r, \{\mathbf{W}^a\}_{a \in A}}{\arg\min} \sum_{i=1}^{N} \ell^r(y_i^r, f(x_i; \mathbf{W}^r)) + \sum_{i=1}^{N} \sum_{a \in A} \lambda^a \ell^a(y_i^a, f(x_i; \mathbf{W}^a)), \tag{1}$$

The elements of this objective function are defined as follows:

- $\underset{\mathbf{W}^r,\{\mathbf{W}^a\}_{a\in A}}{\arg\min}$ is the command that finds the set of parameters $\mathbf{W}^r$ for the main task r and $\{\mathbf{W}^a\}_{a\in A}$ for the auxiliary tasks $a \in A$ that minimize the given function.
- $\sum_{i=1}^{N} \ell^r(y_i^r, f(x_i;\mathbf{W}^r))$ is the first summation term representing the loss for the main task r over N samples.
  - ℓ^r is the loss function for the main task r.
  - y_i^r is the true label for the main task r for the i-th sample.
  - $f(x_i;\mathbf{W}^r)$ represents the model's prediction for the input x_i using parameters $\mathbf{W}^r$.
  - N is the total number of samples.
- $\sum_{i=1}^{N}\sum_{a\in A} \lambda^a \ell^a(y_i^a, f(x_i;\mathbf{W}^a))$ is the second summation term representing the weighted sum of losses for the auxiliary tasks $a \in A$ over N samples.
  - λ^a denotes the importance coefficient for the a-th task's error.
  - ℓ^a is the loss function for each auxiliary task a.
  - y_i^a is the true label for the auxiliary task a for the i-th sample.
  - $f(x_i;\mathbf{W}^a)$ represents the model's prediction for the input x_i using parameters $\mathbf{W}^a$.
  - A is the set of auxiliary tasks.

Together with a task-wise early stopping scheme, this method was shown to be more robust in scenarios of severe occlusion compared to the early cascaded CNN-based methods. For anatomical landmark detection in CT and MRI scans, quite different methods based on Deep Reinforcement Learning (DRL) have also been considered [23, 24]. These methods work by designing an agent that can be trained to find the optimal path to the anatomical landmark, both in 2D and 3D. These agents are inspired by the work in [25] which introduced Deep Q-learning. A more recent method of localizing anatomical landmarks in medical images is presented in [26]. This paper proposes a method that consists of two stages, focusing on global and local localization respectively. Both stages consist of a ResNet [27] style CNN with two prediction heads. One head is designed to predict the probability of a landmark in a given image patch, the other is designed to predict displacement vectors pointing towards the landmark.

A task closely related to anatomical structure identification is medical image segmentation. This often involves segmenting an anatomical structure, tumor, or cavity. Anatomy-aided segmentation models have led to breakthroughs in recent years [28, 29]. The idea behind these models is that rather than only focusing on pixel-level information, they incorporate anatomical information such as shape, motion, or context to guide the model. Even more recent methods are based on the popular Transformer architecture. [30] presents the Fully Convolutional Transformer, which integrates convolutional operations directly into the transformer architecture, replacing linear projections in the Multi-Head Self-Attention (MHSA) with depthwise convolutions to preserve spatial context crucial for medical imaging. A state-of-the-art (SOTA) method called MedSAM [31] uses a Vision Transformer (ViT) based architecture comprising an image encoder for feature extraction, a prompt encoder for promptable segmentation, and a mask decoder for generating segmentation results. The base ViT consists of 12 transformer layers which process 1024×1024 input images into 16×16 patches. The mask decoder consists of a transformer and transposed convolutional layer, and produces the segmentation masks. MedSAM is intended as a SOTA foundational model for medical image segmentation.

It was trained on a large-scale medical image dataset with 1,570,263 image-mask pairs, covering 10 imaging modalities and over 30 cancer types.
This subsection reviewed a wide range of advances in medical image analysis and anatomical tasks. The next section will continue with a review of work in the field of SSL.

2.2 Self-Supervised Learning

Self-Supervised Learning is a sub-field of AI that has enjoyed an enormous amount of attention and research over the past years [32, 33, 34, 35]. The key advantage of SSL is its ability to learn from large amounts of unlabeled data [36, 37]. This has been an essential part of scaling modern large-scale AI systems such as BERT, GPT, and SEER. Compared to traditional supervised methods, which are trained on a specific predefined task, SSL methods learn more generic representations that can be useful across a variety of tasks. Medicine is a domain where SSL can be especially useful due to the fact that labeling is often costly, and tasks are not always known beforehand [38, 39]. While there have been many proposed SSL methods [40, 41, 42, 43], the general technique used is to predict any hidden part from any visible part. A schematic example that is intended to visualize this can be seen in Figure 3. This schematic shows a high-level example of a typical supervised and self-supervised objective. It shows two examples for the SSL case, the top one shows a model trained to predict the hidden pixels in an image of a fruit. The bottom example shows a model trained to predict missing words in English sentences. The schematic also shows that the SSL step is usually followed by a downstream task, image classification in this case. In this step, the pre-trained SSL model is adjusted to facilitate classification and fine-tuned on a labeled dataset. The idea is that the SSL model has learned to produce high-quality representations of the data that are useful for the downstream task. This approach has been shown to achieve comparable or even superior performance to its purely supervised counterpart using a smaller labeled dataset [32, 44, 33].

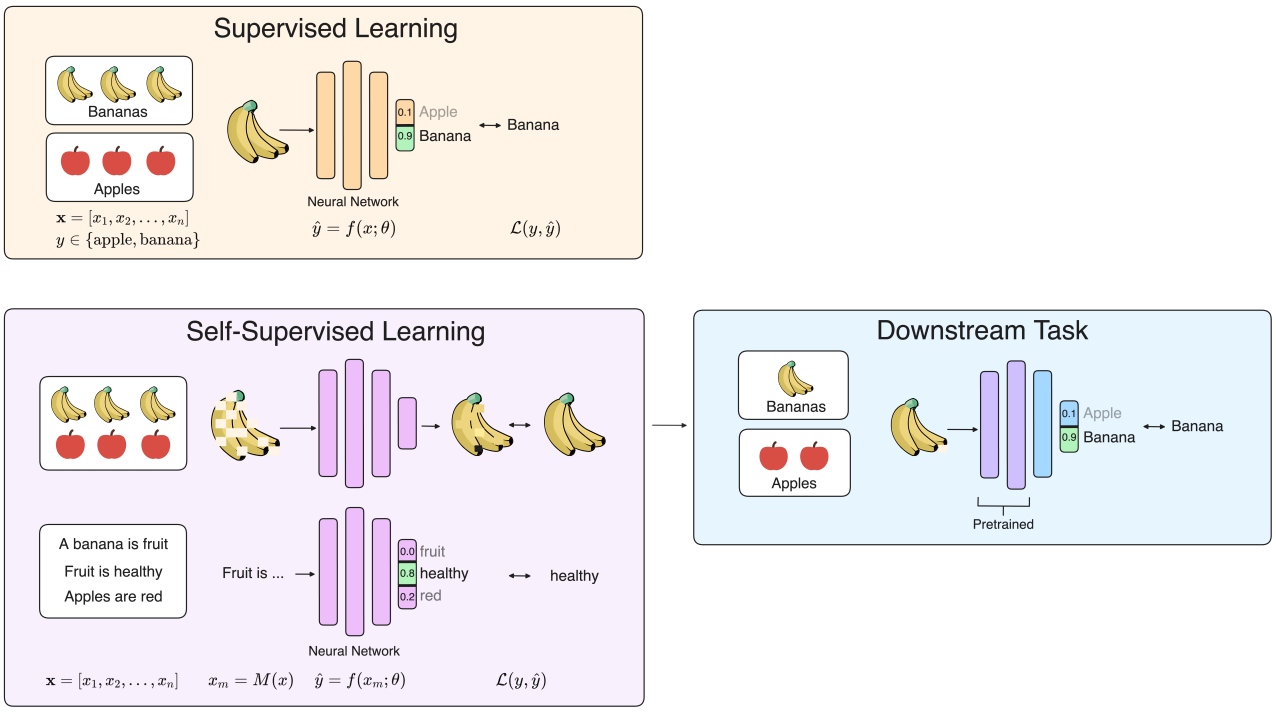


Figure 3: Supervised Learning vs. Self-Supervised Learning, a high-level comparison.

SSL can broadly be categorized into four families [45]. These families are Deep Metric Learning, Self-Distillation, Canonical Correlation Analysis, and Masked Image Modeling. A short technical explanation of each family will follow, intended to show the key concepts and differences.

Deep Metric Learning Deep Metric Learning (DML) originates from the idea of contrastive loss, which was first introduced in [46], and later formally defined in [47]. In DML a network is trained to predict if two samples are from the same class based on the proximity between their respective embeddings. To achieve this, variants of the inputs are created through transformations that preserve semantic meaning. These variants are referred to as negative and positive pairs. During training, the network makes the positive pairs more similar in embedding space, and the negative pairs dissimilar. A method proposed by Oord et al. [40] defines the Information Noise-Contrastive Estimation loss or InfoNCE for short. This loss aims to maximize the mutual information between different views or augmentations of the same data point. Given a set $X = \{x_1, \dots, x_N\}$ of N random samples containing one positive sample from $p(x_{t+k}|c_t)$ and $N-1$ negative samples from the 'proposal' distribution $p(x_{t+k})$, InfoNCE optimizes for:

$$\mathcal{L}_N = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right] \tag{2}$$

The elements of this loss function are defined as follows:

- $\mathop{\mathbb{E}}_{X}$: The expectation over the set of samples X.
- x_{t+k}: The positive sample drawn from the conditional distribution $p(x_{t+k}|c_t)$.
- x_j: The j-th sample in the set X, which includes one positive sample and $N-1$ negative samples.
- $f_k(x_{t+k}, c_t)$: The scoring function that measures the compatibility between x_{t+k} and c_t.
- $\sum_{x_j \in X} f_k(x_j, c_t)$: The normalization term that sums the scores of all samples in the set X with the context c_t.
- c_t: The context or representation of the original data point.

While the original contrastive loss penalizes negative pairs based on a margin, InfoNCE loss leverages a probabilistic approach by treating the contrastive learning problem as a binary classification task. It aims to distinguish the positive sample from the negative samples in X by maximizing the similarity in embedding space between positive pairs while also minimizing it for negative pairs.
SimCLR [32] is one of the most popular methods utilizing InfoNCE. SimCLR is designed to learn visual representations by encouraging similarity between two augmented views of an image. Augmentations include resizing, cropping, color jittering, and random blurring. The embeddings created by SimCLR have been shown to increase the performance of downstream tasks.

Self-Distillation Self-Distillation methods work by constructing two encoders, called the teacher and student, and feeding two different views of the same image to these encoders. A tertiary predictor network is then tasked to map the output of one encoder to the other. This can easily lead to the collapse of the encoders, a failure mode where the encoder produces uninformative or degenerate representations. One of the core methods to prevent this collapse is to update the weights of the first encoder, with a running average of the second encoder's weights. Popular self-distillation methods are Bootstrap Your Own Latent (BOYL) and self-distillation with no labels (DINO) [34, 41]. BOYL was first introduced to deal with the problem of collapse. In BOYL, the student and teacher receive different views from the same image via transformations such as random resizing, cropping, color jittering, and brightness alterations. The student network is updated via gradient descent, while the teacher is updated based on an exponential moving average of the weights of the student network. This asymmetry in updates is essential in preventing collapse. The loss of BOYL is defined as:

$$\mathcal{L}_{\text{BYOL}}(\theta_s,\gamma) = \mathbb{E}_{(x,t_1,t_2)\sim(X,T_1,T_2)}\left[\left\|\text{renorm}(p_\gamma(f_{\theta_s}(t_1(x)))) - \text{renorm}(f_{\theta_t}(t_2(x)))\right\|_2^2\right] \tag{3}$$

The elements of this loss function are defined as follows:

- $\mathbb{E}_{(x,t_1,t_2)\sim(X,T_1,T_2)}$: The expectation over the set of samples X and the transformations T_1 and T_2. Here, x represents a data point, t_1 and t_2 are augmentations applied to x.
- x: A sample from the dataset X.
- t_1, t_2: Transformations applied to x. These transformations create different augmented views of the same data point.
- $f_{\theta_s}(t_1(x))$: The output of the online network f with parameters θ_s when the transformation t_1 is applied to x.
- $f_{\theta_t}(t_2(x))$: The output of the target network f with parameters θ_t when the transformation t_2 is applied to x.
- p_γ: A predictor network parameterized by γ, which maps the online network's output to the target space.
- renorm: A re-normalization function that ensures the outputs of the networks have unit norm, used to stabilize training.
- $\|\cdot\|_2^2$: The squared Euclidean norm, which measures the squared distance between the normalized outputs of the online network's predictor and the target network.

Canonical Correlation Canonical correlation was first introduced through the Canonical Correlation Framework (CCA) in [43]. CCA aims to find linear combinations, called canonical variates, of the variables in two sets of data such that the correlation between them is maximized. During the process of finding these projections, CCA projects the data into a shared latent space. This shared latent space can be seen as a feature space where the most important relationships between the views are captured. These features can be used for downstream tasks. The CCA loss is defined as:

$$\begin{gathered}\mathcal{L} = -\sum_{n=1}^{N}\langle U_n, V_n\rangle, \text{ such that} \\ \frac{1}{N}\sum_{n=1}^{N} U_n = 0, \quad \frac{1}{N}\sum_{n=1}^{N} V_n = 0, \quad \text{(zero-mean representations)} \\ \frac{1}{N}U^T U = \frac{1}{N}V^T V = I, \quad \text{(identity covariance representations)}\end{gathered} \tag{4}$$

The elements of this loss function are defined as follows:

- $X \in \mathbb{R}^D$: The random variable X in a D-dimensional real space.
- $Y \in \mathbb{R}^D$: The random variable Y in a D-dimensional real space.
- $U = f_x(X)$: The transformation of X using the function f_x to obtain U.
- $V = f_y(Y)$: The transformation of Y using the function f_y to obtain V.
- $\mathcal{L}$: The loss function to be minimized.
- $\sum_{n=1}^{N} \langle U_n, V_n \rangle$: The sum of the inner dot products between the transformed variables U and V overall N samples.

The loss function should satisfy two constraints. First, it should satisfy the zero-mean representations, which ensures that the learned representations are centered around the origin. This aims to remove bias from the data. Secondly, it should satisfy the identity covariance representations, ensuring the learned representations have unit variance and are uncorrelated. While CCA originated from the field of multivariate statistics, it has been successfully extended to deep learning in well-known approaches such as VICReg and BarlowTwins [48, 49].

Masked Image Modeling Early SSL pre-training strategies relied on image degradation techniques such as noise, or shuffling patches [50, 51, 52]. Modern methods that fall in the family of Masked Image Modelling (MIM) are based on the principle of masking. The high-level idea of masking is represented in the previously mentioned Figure 3, where a mask is applied to the model input in order to hide portions of the input image. The model is subsequently trained to predict the missing pixels. The success of this method builds upon two major innovations, the Vision Transformer and the idea of a learnable mask token introduced in BERT [44]. BERT is a language model that replaces the hidden text tokens with learnable mask tokens and teaches the model to recover the original text. This basic principle remains very popular in the modern landscape of Large Language Model pre-training. Similarly, Masked Autoencoders (MAEs) and SimMIM extend this principle to the visual domain [33, 42]. MAEs work on two core principles, the first is an asymmetric encoder-decoder architecture, with the encoder only operating on visible patches. The lightweight decoder tries to reconstruct the original image from the latent representation and mask tokens. The second principle is that masking a high portion (75%) creates a nontrivial and meaningful self-supervisory task. After the SSL training stage, the MAE encoder is able to produce highly informative latent representations of the images. For downstream tasks, the decoder can be replaced by a new prediction head designed for that specific task, such as image classification.
Masked Autoencoders have also been extended to the 3D domain. Given that this project focuses on 3D point cloud data, these adaptations are highly relevant. [6] introduces Point-MAE for point cloud self-supervised learning. Extending the MAE to point clouds is not straightforward, and it involves some unique steps. On a high level, PointMAE works by breaking the point cloud up into patches via farthest point sampling and k-nearest neighbors. After this, each patch is embedded, and a high portion is masked. From there the approach is similar to the traditional MAE, namely to predict the masked points. The loss function needs to capture the difference between predicted points and ground truth. This can be achieved by creating a loss function based on the Chamfer Distance [53], which is defined as:

$$\mathcal{L} = \frac{1}{|P_{\text{pre}}|} \sum_{a \in P_{\text{pre}}} \min_{b \in P_{\text{gt}}} \|a-b\|_2^2 + \frac{1}{|P_{\text{gt}}|} \sum_{b \in P_{\text{gt}}} \min_{a \in P_{\text{pre}}} \|a-b\|_2^2 \tag{5}$$

Here P_{pre} are the predicted point patches, with P_{gt} being the ground truth. Chamfer Distance essentially looks at two sets of points. For each point within each set, the algorithm finds the nearest neighbor in the other set and sums the squared distances up. A similar approach called Point-M2AE [5], aims to capture both fine-grained and high-level semantics of 3D shapes by modifying the encoder and decoder into pyramid architectures to progressively model spatial geometries. Point-MAE is one of the core components of this project, and it will be discussed in detail in Chapter 3.
This subsection reviewed significant work in the four main SSL families. Section 2.3 will continue with a review of work in the field of Graph Neural Networks.

2.3 Graph Neural Networks

As outlined in Section 1.3, this project intends to leverage Graph Neural Networks to enhance the anatomical structure identification system. Neural Networks have traditionally been used to operate on fixed-size and/or regular-structured inputs such as images, text, or video. However, many real-world entities are defined in terms of their connections to other things, and can therefore naturally be expressed as graphs. Examples of entities or information that can be expressed as graphs are molecules, social networks, or in the case of this thesis, anatomical structures. These objects come in the form of graph-structured data, and traditional neural networks cannot directly process such information effectively. Graph Neural Networks are a family of neural networks that can operate naturally on graph-structured data and actually make use of the information in the nodes and edges to guide the learning process. The earliest GNN was presented in [54], and since then the field has seen many breakthroughs that increased the capabilities of the GNN [55, 56, 57]. Nowadays, GNNs have successfully been applied to practical applications such as antibiotic discovery, fake news detection, and traffic prediction [58, 59, 60].

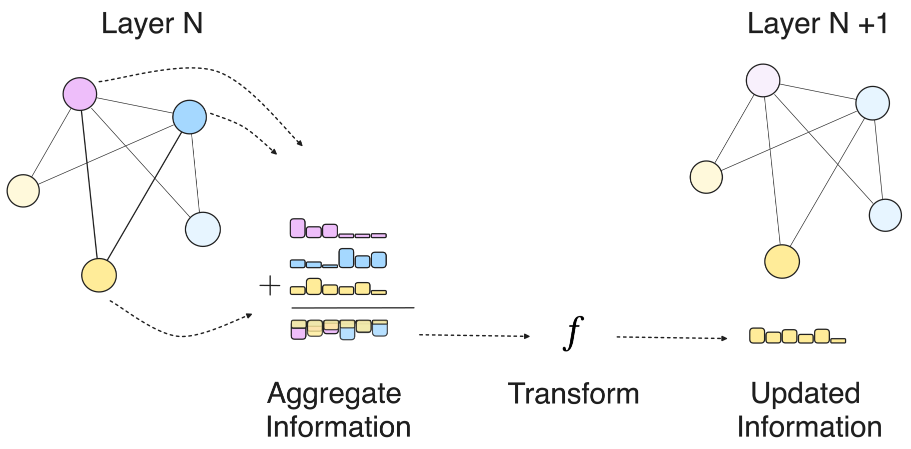


Figure 4: Node embedding updates via message passing.

An important feature of the GNN mechanism is that the mechanism should preserve graph symmetries, or in other words, it should be permutation invariant. One of the core operations of the GNN is message passing. Message passing is an algorithm that allows neighboring nodes or edges to exchange information and influence each other's node embeddings. This can essentially be seen as a way to aggregate information within the graph. Figure 4 shows how node embeddings are affected by their neighbors via message passing. Message passing for GNNs was formally defined in [61], but even before this work, very similar concepts had been explored. [55] tried to extend the idea of a convolution to the domain of GNNs with the graph convolution. Let $h_v^{(0)} = x_v$ be node v's initial embedding or all $v \in V$. Then for step k = 1, 2, ... up to K, the graph convolution updates the node embeddings via:

$$h_v^{(k)} = f^{(k)}\left(\mathbf{W}^{(k)} \cdot \frac{\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + \mathbf{B}^{(k)} \cdot h_v^{(k-1)}\right) \quad \text{for all } v \in V. \tag{6}$$

The components of this update function are defined as:

- $h_v^{(k)}$: The embedding of node v at layer k.
- $f^{(k)}$: An activation function at layer k.
- $\mathbf{W}^{(k)}$: The weight matrix for layer k.
- $\mathcal{N}(v)$: The set of neighbors of node v in the graph.
- $|\mathcal{N}(v)|$: The number of neighbors of node v.
- $\mathbf{B}^{(k)}$: A weight matrix for the self-connection at layer k.
- V: The set of all nodes in the graph.

Here the activation function $f^{(k)}$, weight matrix $\mathbf{W}^{(k)}$ and $\mathbf{B}^{(k)}$ are shared across all nodes. This graph convolution can be interpreted as a form of localized message passing. The fact that these weight matrices are shared allows the model to scale well because the number of parameters in the model is not tied to the size of the graph. Another framework presented in [57] called GraphSAGE proposed a framework for learning node embeddings via sampling and aggregation of a node's local neighborhood. GraphSAGE updates the node embeddings via:

$$h_v^{(k)} = f^{(k)}\left(\mathbf{W}^{(k)} \cdot \left[\underset{u \in \mathcal{N}(v)}{\mathrm{AGG}}\left(\left\{h_u^{(k-1)}\right\}\right), h_v^{(k-1)}\right]\right) \quad \text{for all } v \in V. \tag{7}$$

Where again, the activation function $f^{(k)}$, weight matrix $\mathbf{W}^{(k)}$ and *AGG* are shared across all nodes. GraphSAGE considers three choices for *AGG*, which are Mean, Dimension-wise Maximum, and LSTM. Due to the effectiveness of GNNs, many of the architectures and methods from traditional neural networks have been extended to graph-based tasks. This is of course also true for the Transformer, which is presented in [62]. This mechanism is used in the GNN part of this project and will be explained in detail in Chapter 3. Given that this project revolves around 3-dimensional data, it is also worth noting that GNNs have also successfully been applied to point cloud data. [63] proposes Point-GNN, a graph neural network to detect objects from a LiDAR point cloud. Point-GNN consists of three main components: graph construction from a point cloud, a graph neural network for object detection, and bounding box merging and scoring.
This section provided a comprehensive review of key concepts and methodologies relevant to this thesis. We have explored machine learning techniques in medical imaging, including anatomical landmark detection and image segmentation. Following this, we discussed the field of SSL, discussing its four main families. Finally, we reviewed pivotal work in the field of GNNs. Building upon this foundation of knowledge, we will now proceed to Chapter 3, where we will detail the specific methods employed in this project to address the problem of anatomical structure identification.

3 Methods

This Chapter will cover the methodology of the anatomical structure identification pipeline, including the datasets used for training, validation, and testing of our method. We begin by presenting a detailed discussion of these datasets in Section 3.1. After this, we continue with an in-depth exploration of our proposed approach. A high-level overview of the main steps in our methodology can be found in Figure 5. From this figure, three distinct steps can be identified. Step one is centered around SSL, and is discussed in detail in Section 3.2. In step two, features from the pre-trained encoder discussed in step one are integrated into a predefined graph, which is elaborated on in Section 3.3. In the final step, a GNN is used for structure identification, as detailed in Section 3.4.

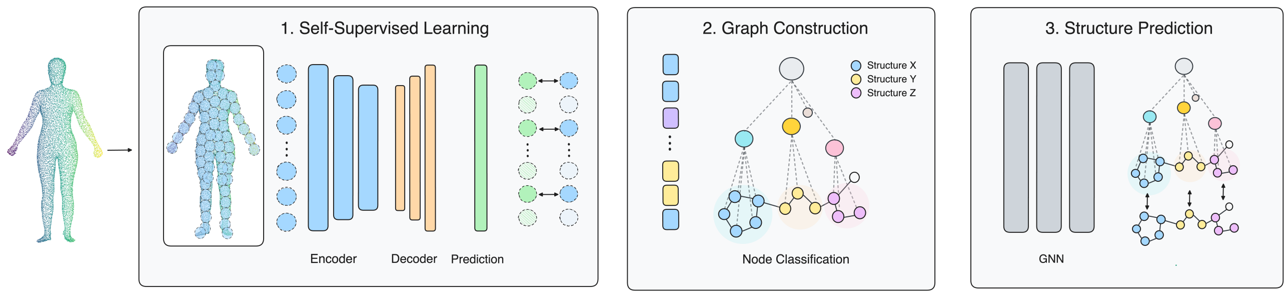


Figure 5: Main components of the methodology.

3.1 Datasets

This section introduces four datasets used in this thesis: TableNet (subset of PartNet), CAESAR, Table-GRAPE, and Anatomy-GRAPE. TableNet and CAESAR are used during the self-supervised stage, while Table-GRAPE and Anatomy-GRAPE are employed during the supervised stage. TableNet and Table-GRAPE are designed for an exploratory toy problem, helping us to validate our approach in a simpler context. Conversely, CAESAR and Anatomy-GRAPE target the more complex challenge of anatomical structure identification.

PartNet Ideally, we would use an open-source dataset comprising thousands of anatomically labeled 3D human models. While there exist decent sized datasets of human models, none contain the labels we require. To efficiently evaluate our method's effectiveness before undertaking the manual annotation of such a dataset, we propose a preliminary assessment referred to as the toy problem. The objective here is simplified: develop a method capable of predicting part labels for a systematically structured 3D object represented as a point cloud. This adjustment allows us to leverage existing datasets of part-labeled 3D objects, while still being extendable to our original problem of human anatomical structure identification.

PartNet is a large-scale dataset of 3D objects annotated with fine-grained, instance-level, and hierarchical 3D part information [64]. With a collection of 26,671 3D models spanning 24 everyday object categories such as tables, lamps, and vases, the PartNet dataset serves as a valuable resource for research in areas like 3D part recognition, fine-grained object segmentation, and hierarchical object segmentation.

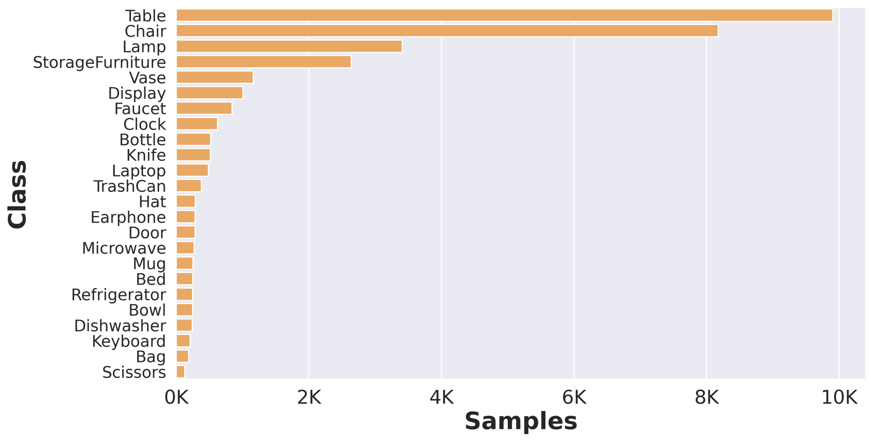


Figure 7: PartNet object class distribution.

Figure 6 illustrates representative samples from four of the 24 object categories in the PartNet dataset, where each part is distinguished by a unique color. The complexity of objects can vary, with some comprising as few as two or three parts, whereas others consist of more than 200. The distribution across these 24 object categories is depicted in Figure 7, revealing a pronounced skewed distribution. Notably, categories such as table and lamp consist of close to 10,000 samples, while most other categories contain fewer than 1,500 samples. Beyond mere part labeling, every object in the dataset also adheres to an expert-defined part hierarchy, an example of which is displayed in Figure 8. This hierarchy defines the interconnection between parts and identifies combinations of parts that constitute distinct sub-structures.

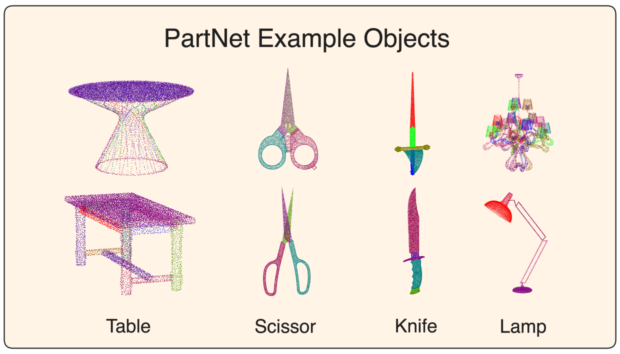


Figure 6: PartNet example objects.

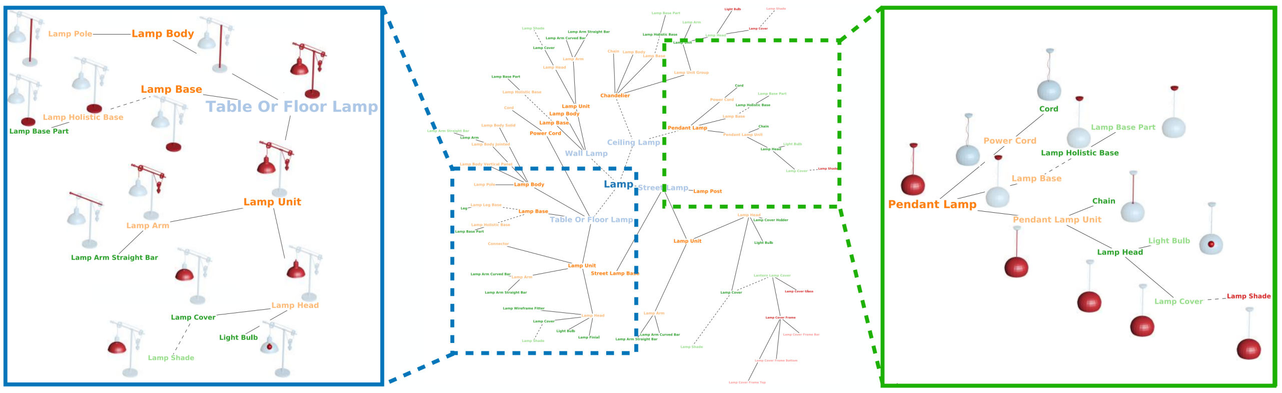


Figure 8: Expert-defined hierarchical template for the lamp (middle) and the instantiations for a table lamp (left) and a ceiling lamp (right). The And-nodes are drawn in solid lines and Or-nodes in dash lines [64].

PartNet offers a complete dataset that is well-suited for evaluating our toy problem. It provides a broad collection of 3D object point clouds, ideal for the self-supervised learning phase of our pipeline. Furthermore, it includes detailed part labels necessary for training our part classifier. Finally, PartNet provides part hierarchies, detailing the interconnection within objects that can be used as a starting point when creating a graph dataset for the Graph Neural Network segment of our pipeline. However, since PartNet encompasses multiple object categories, we narrow our focus to align with the ultimate goal of anatomical structure identification, which involves a single object category—the human body. The next paragraph will introduce TableNet, a subset of PartNet that only involves Table objects.

TableNet Ignoring male and female anatomical differences, as well as medical defects, it can be expected that the anatomy across human individuals broadly follows the same model. Therefore, we view the human anatomy as a singular object category, that follows the same anatomical structure from person to person. To align the toy problem closely with the ultimate goal of anatomical structure identification, we are therefore interested in predicting parts for one object category, rather than predicting parts across all 24 object categories provided by PartNet. This is why we create a sub-dataset from PartNet, which requires two main steps. First, we select a preferred object category and filter out the rest. Then, we apply filtering and cleaning to the labels, parts, and samples within this sub-dataset. To ensure reproducibility, we will go into detail on how this sub-dataset has been created. The first step involved selecting an appropriate object category from PartNet. As illustrated in Figure 7, among the available object categories only *Table* and *Lamp* offered a sufficient volume of samples. The choice ultimately favored *Table* due to its more consistent structural and shape characteristics across samples, in contrast to the diverse and sometimes odd designs observed in the *Lamp* samples. In the second step, we clean the remaining samples. Each sample comes in the form of a point cloud, where each point is tagged with a label indicating to which part it belongs. This means that given a point cloud consisting of 10,000 points, there will be an equivalent number of labels. Looking at Figure 9, which shows the distribution of point labels across all table samples, it becomes clear that this distribution is highly skewed. Further analysis of the expert-defined hierarchy in Figure 31 reveals three table categories: ”Regular Table”, ”Picnic Table” and ”Game Table”, with the latter two accounting for 913 samples and featuring many infrequent parts like ”Airhockey Puck” or ”Foosball Rod”. Due to their atypical shapes and components, all ”Picnic Table” and ”Game Table” samples were excluded from the dataset. For the remaining samples, we only retained the 22 most common part labels, assigning all other points in the ”miscellaneous” label, resulting in a total of 23 labels for the table object category. After completing these modifications

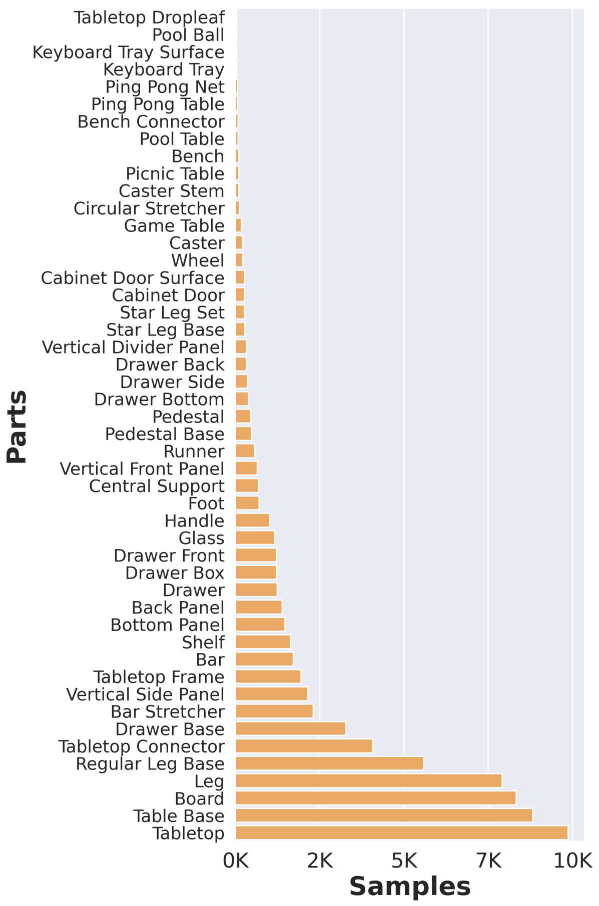


Figure 9: Distribution of point labels across TableNet.

of PartNet, we established the new dataset termed TableNet. This dataset consists of 7,670 point clouds depicting various tables, where each individual point within a sample is categorically labeled according to 23 unique part labels.

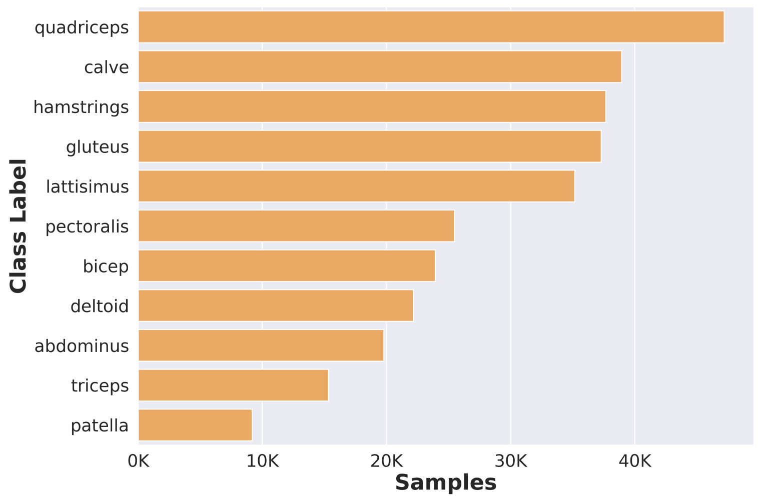


Figure 10: Distribution of point labels across annotated CAESAR.

CAESAR To train and evaluate our main objective of anatomical structure identification, we require a dataset with two key features: 1) a diverse set of realistic 3D human models comprising both male and female samples with sufficient physiological variation, and 2) corresponding anatomical labels that specify which locations on the 3D surface correspond to specific anatomical structures. As stated earlier in this section, we were unable to find an open-source dataset that met both these conditions. To overcome this, we use an unlabeled dataset of 3D human bodies called CAESAR [65], and manually annotate a small portion to obtain the required anatomical labels. CAESAR contains 3D whole-body scans for both men and women of various weights, between the ages of 18 and 65. We used a publicly available version of the CEASAR dataset provided by Yang et al. [66]. This dataset contains 3000 3D human meshes, evenly split between male and female models. Each mesh is registered to a common topology of 12,500 vertices and 25,000 facets. Since we are interested in point cloud models, these meshes are converted to a point cloud format, with 12500 points per point cloud. One point is sampled per vertex, meaning there is no exact common space or resolution between points. Examples of both the raw mesh and subsequent point cloud format are provided in Figure 11. To acquire anatomical labels, we first identified key anatomical structures that are well-suited to verify our method. A constraint was that the task remained feasible for a single person to annotate within a limited time frame. After consulting with two trained medical experts, one physician and one medical professional trained in technical medicine, we obtained a set of 10 large muscular structures and 1 small bone structure. The small bone structure was selected to test the model's capability of identifying smaller anatomical structures. Ideally, we would test more than one, but time and annotation constraints prevented this. The set of selected structures is: Hamstring, Quadricep, Calve, Gluteus, Deltoid, Bicep, Latissimus, Pectoralis, Abdomin, and Patella. Annotation of these structures was performed in Blender [67]. Each structure was paired with a unique color code. Annotating a mesh was found to be more straightforward than annotating a point cloud, therefore, we annotated the mesh and then converted it to a point cloud. The annotation process followed a step-by-step method. For each sample to be annotated:

1. Load the raw mesh into blender.
2. Annotate all 11 structures in the mesh by in-painting the location in the vertex paint mode.
3. Save the mesh in a PLY format to retain color information.
4. Convert the mesh to a point cloud using the Python trimesh library. We sample 1 point per triangle. Points retain color information.
5. Generate a CSV file containing the integer labels of all points for the sample. The label is assigned based on the color of the point.

a total of 80 samples were annotated, of which 40 were male and 40 female. After annotation, 5 random samples were selected and inspected by the previously mentioned medical professionals to check for medical and anatomical accuracy. The annotations were deemed to be sufficiently accurate for the purposes of this thesis.

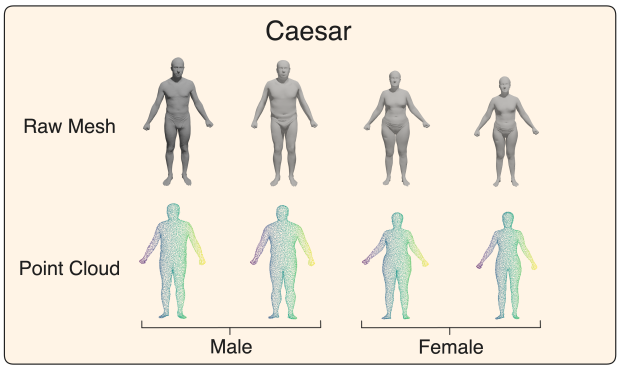


Figure 11: CAESAR examples, raw mesh, and point cloud format.

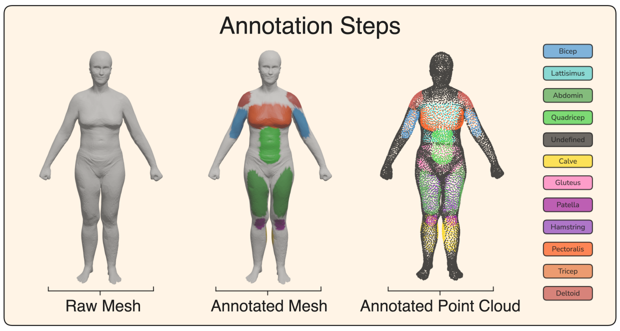


Figure 12: Annotation steps for the CAESAR data.

GRAPE **G**raph **R**epresentation of **A**dvanced **P**art **E**ncodings or GRAPE is a novel dataset created for training the GNN that comprises GNN portion of the anatomical structure identification pipeline. GRAPE is comprised of graph-based representations of 3D objects, where nodes are enriched with latent representations of their corresponding object parts. To create this dataset, we need three components: a default graph structure of the 3D object, the latent representations of the parts, and the part labels for each latent. We will create two versions of GRAPE, one for our toy problem called Table-GRAPE, and one for our main task called Anatomy-GRAPE.

Table-GRAPE builds on TableNet. Its default graph structure is a modified version of the expert-defined table hierarchy 31 that accounts for the modifications detailed in the paragraph on TableNet 3.1. This modified graph is depicted in Figure 13. A pre-trained encoder, detailed in Section 3.2, generated latent representations for table parts within the TableNet dataset. These parts are represented by the leaf nodes of the graph. The embeddings for the non-leaf nodes are created by averaging the embeddings of their direct children. This process is repeated iteratively, calculating the mean embeddings at each level, continuing until the top node of the graph is reached. The part labels from TableNet are propagated to the newly generated part latents. As each latent represents a cluster of points from the original point cloud, a majority voting scheme is used to determine the part latent label. This process results in a dataset containing graph-based representations of tables and their parts, where each node is associated with a rich, high-dimensional feature vector that encapsulates the geometric and positional properties of the table part. The edges represent the spatial and hierarchical relationships between different parts.

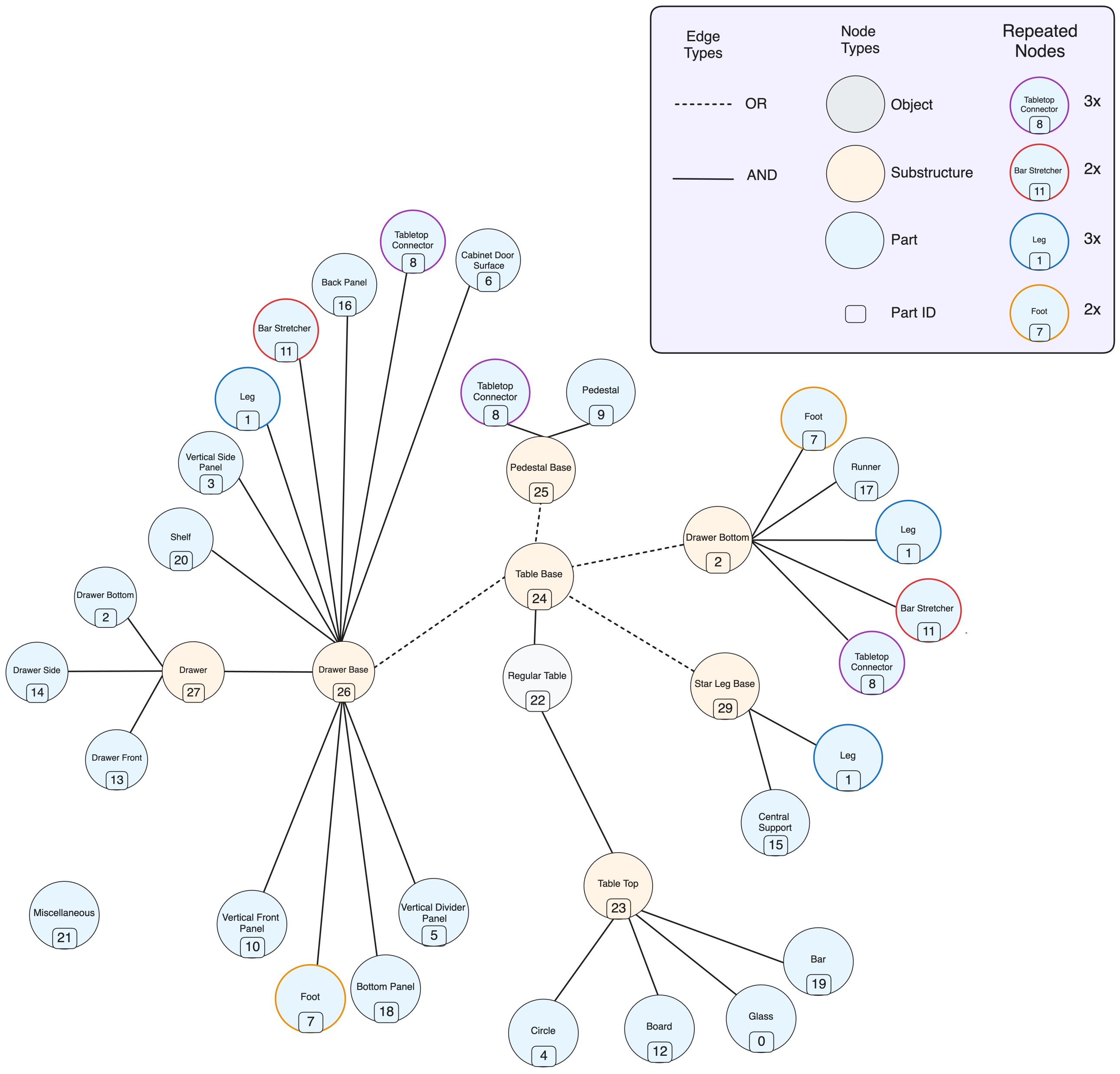


Figure 13: Default table graph used to construct Table-GRAPE.

Anatomy-GRAPE builds on the annotated portion of CAESAR. This means we have 80 samples that can be converted into a GRAPE format and used for training, validation, and testing of the GNN. A relatively small dataset, but as the GNN does not require much data as discussed in 6 sufficient to validate our methodology. The default graph structure for Anatomy-GRAPE is displayed in Figure 14. A notable distinction between the default graph of Anatomy-GRAPE and Table-GRAPE is the absence of OR edges in Anatomy-GRAPE. The construction process for both graphs follows the same methodology detailed in the preceding paragraph on Table-GRAPE. This method is explained in more detail in Section 3.3.

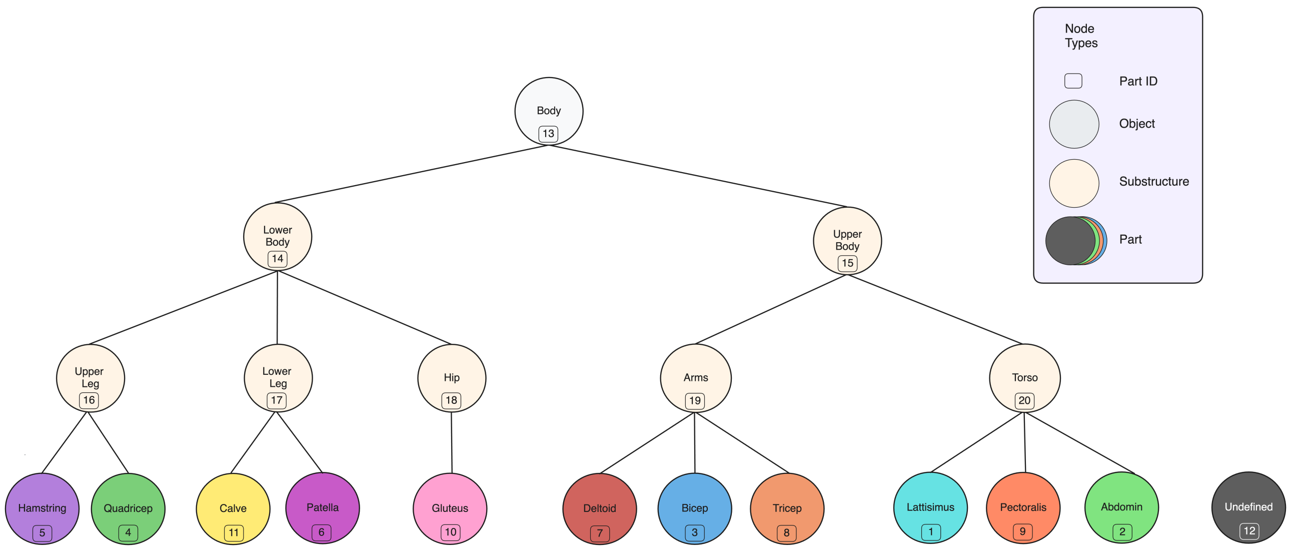


Figure 14: Default Graph used to construct Anatomy-GRAPE.

3.2 Masked Autoencoders for Point Cloud Self-supervised Learning

This section delves into the adaptation of Masked Autoencoders (MAEs) for point cloud SSL, focusing on the framework proposed by Pang et al. [6], called Point-MAE. Point clouds present a unique set of challenges due to their structural complexity. Traditional MAEs, as pioneered in the context of image SSL [33], reconstruct highly masked input images. Point-MAE extends this idea to point clouds, with the goal of reconstructing a point cloud from a highly masked version. However, 3D point clouds differ significantly from 2D images. In the context of MAEs for image reconstructions, certain aspects, such as designing a masking strategy, are fairly straightforward. Adapting such a strategy for point clouds presents distinct challenges due to the inherent complexity of 3D spatial relationships. Other challenges complicate the task further. One such issue is the leakage of location information, where the positional data of masked parts is inadvertently revealed to the model, reducing the complexity of the task. Additionally, the problem of uneven information density, where some areas of the point cloud are less densely populated with points than others, further complicates the reconstruction task. These challenges require a modification of the standard MAE approach used in images. In the following sections, we will go over our implementation of Point-MAE and address these challenges

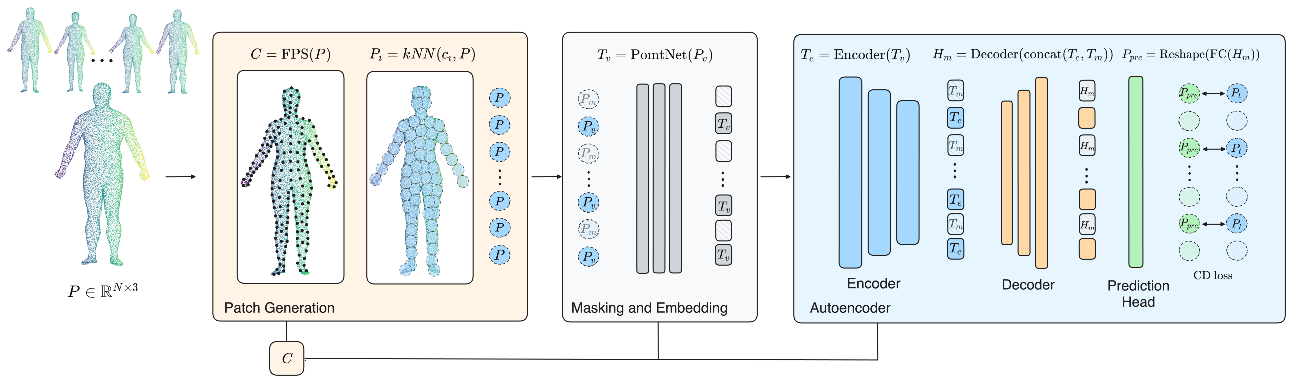

$P \in \mathbb{R}^{N\times 3}$
$C = \mathrm{FPS}(P)$
$P_i = kNN(c_i, P)$
Patch Generation
$T_v = \mathrm{PointNet}(P_v)$
Masking and Embedding
$T_e = \mathrm{Encoder}(T_v)$
$H_m = \mathrm{Decoder}(\mathrm{concat}(T_e, T_m))$
$P_{pre} = \mathrm{Reshape}(\mathrm{FC}(H_m))$
Encoder
Decoder
Prediction Head
CD loss
Autoencoder
C

Figure 15: Point-MAE Architecture.

Point-MAE Figure 15 illustrates the high level architecture of Point-MAE. There are two main steps: first the input point cloud is processed via a masking and embedding step, and next the masked point patches are passed through a standard Transformer-based Autoencoder, tasked with reconstructing the masked sections. The remainder of this section will be dedicated to a detailed explanation of each step within the Point-MAE architecture.

Point Patches Generation Patch generation is a highly crucial part of this architecture, as these patches will serve as input tokens for our model. A naive approach would be to treat every single point as an individual token. However, due to the quadratic complexity of the self-attention mechanism in the Transformer, such an approach would be inefficient with respect to the consumption of computational resources. In Vision Transformers (ViT) [68], the input image is divided into P regular patches, which are then treated as the input tokens. For data with a two-dimensional geometry, this is a very natural method of patch creation, however, a successful extension of this idea to 3D point clouds requires some extra steps. Taking inspiration from Point-BERT [69], Point-MAE uses two additional algorithms to construct point patches. Given a point cloud consisting of N points, we first use Farthest Point Sampling (FPS), to identify center points in our point cloud. Subsequently, these center points are used by the K-Nearest Neighbors (KNN) algorithm to construct clusters of points, which will form our point patches. A high-level pseudo algorithm of FPS and KNN can be found in Algorithm 1 and 2 respectively. Formally, given an input point cloud with p points $X^i \in \mathbb{R}^{p\times 3}$, FPS samples n points for centers C. Based on the center points C, KNN selects k nearest neighbors from the input for corresponding point patch P:

$$C = FPS(X^i), \quad C \in \mathbb{R}^{p\times 3} \tag{8}$$

$$P = KNN(X^i, C), \quad P \in \mathbb{R}^{p\times 3} \tag{9}$$

It is important to recognize that this method of patch generation has some drawbacks. Firstly, given a sufficiently dense point cloud, this method is very likely to introduce point-overlapping patches. Secondly, we need to manually set the number of patches ($numPatches$) we would like to sample, and how many points each patch should contain ($pointsPerPatch$), which has a very important consequence; if the ratio $numPatches \times pointsPerPatch : N$ is too low, some points may not belong to any patch, resulting in a loss of information. For example, a $1:1$ ratio will result in such a situation. The creators of Point-MAE empirically found a ratio of $numPatches \times pointsPerPatch = 2N$ to be effective. [2]

Algorithm 1 Farthest Point Sampling (FPS).

```
1:  Input: Point set $P$, number of points $npoints$
2:  Output: Subset of selected points $S$
3:  procedure FPS($P$, $npoints$)
4:      Initialize $S$ with a random point from $P$
5:      while $|S| < npoints$ do
6:          Find $p_{farthest} \in P \setminus S$ with the maximum distance to the nearest point in $S$
7:          Add $p_{farthest}$ to $S$
8:      end while
9:      return $S$
10: end procedure
```

[2]This finding followed from personal correspondence with the creators of Point-MAE.

Algorithm 2 K-Nearest Neighbors (KNN).

```
 1: Input: Query point $q$, set of points $P$, number of neighbors $k$
 2: Output: Set of $k$ nearest neighbors of $q$ from $P$, denoted as $N_k(q)$
 3: procedure KNN($q$, $P$, $k$)
 4:     Initialize an empty priority queue $Q$ with size $k$
 5:     for each point $p \in P$ do
 6:         Calculate distance $d$ between $q$ and $p$
 7:         Insert $p$ into $Q$ with priority $d$
 8:         if $Q$ size exceeds $k$ then
 9:             Remove the point with the highest distance from $Q$
10:         end if
11:     end for
12:     return points in $Q$ as $N_k(q)$
13: end procedure
```

While FPS + KNN is a well-established and popular method of patch generation [6], [69], [5], voxel-based methods present an alternative approach worth considering. This method breaks the input up into non-overlapping voxels [70], [71]. The benefit of this approach is that it is guaranteed that every point is part of a voxel, unlike the FPS + KNN approach, however, it also presents several unique challenges. Firstly, a large portion of the voxels in the field of view are generally empty. Secondly, the inherently irregular point density in most point clouds leads to voxels containing varying numbers of points, which requires an architecture that supports variable-sized input tokens. This, in turn, requires an architecture capable of handling input tokens of different sizes, such as the dynamic voxel feature encoder [72]. Considering the established efficacy of the FPS + KNN method, we chose this approach for our patch generation. However, it is essential to acknowledge the limitations of any selected algorithm. Specifically for FPS + KNN, the choice of the ratio between $numPatches \times pointsPerPatch$ and N is critical, as setting this too low can result in a situation where some points may not belong to any patches, thus losing information.

Masking Taking inspiration from BERT in language [44] and MAEs in images [33], we present a similar masking strategy. Taking into consideration that point patches may overlap, and certain points can be part of more than one patch we mask patches instead of individual points. This is done to keep information complete in each point patch. This means that the masking of a shared point does not cause an incomplete representation in any of the patches it belongs to. It has empirically been found that very similar to image MAEs, random masking with a high masking ratio of around 60-80 percent works well [6]. The set of masked point patches is used as a ground truth when calculating the reconstruction loss. Given a masking ratio m, we define the set of masked point patches as $P_{gt} \in \mathbb{R}^{mn \times k \times 3}$

Embedding For the embedding process of each masked point patch, Point-MAE employs a share-weighted learnable mask token. This singular token representation is universally applied as a placeholder for all masked point patches in the training batch. This design choice is grounded in both computational efficiency and memory optimization. By using a single token representation that is updated during training, the share-weighted masked point patch efficiently learns a high-quality, generalized representation of the missing information in a point cloud. To understand why it is possible to share a single token over all masked patches, it is crucial to consider the processing it undergoes

in the transformer blocks. The transformer architecture includes both a self-attention mechanism and positional encoding, which capture the contextual information surrounding each masked patch, allowing the model to integrate the characteristics of the local environment surrounding that point.
As the order of points within a point cloud does not carry any inherent meaning, the embedding should follow the principle of permutation invariance, meaning that the network should produce the same output regardless of how the input points within a patch are ordered. The ViT [68] applies a simple linear projection of the flattened patches, however, in the case of point cloud data, this method will fail. This is because a linear projection will involve a matrix multiplication, which inherently is not permutation-invariant, meaning that changing the order of vectors (points in this case) in the input will generally change the result of the multiplication. Hence, using a linear projection could result in the model relying on the specific order of points, which would contradict the desired property. Point-MAE therefore implements a lightweight PointNet [73]:

$$T_v = PointNet(P_v), \quad T_v \in \mathbb{R}^{(1-m)n \times C} \tag{10}$$

Here, the visible point patches $P_v \in \mathbb{R}^{(1-m)n \times k \times 3}$ (where 3 represents the x, y, z coordinates) are embedded into visible tokens T_v. This embedding strategy mainly consists of Multilayer Perceptrons (MLPs) and max pooling layers. Considering the importance of the spatial position of each point patch, we employ coordinates that have been normalized with respect to the patch center, ensuring scale and position consistency, in addition to a global understanding of the structure. Given the transformer-based nature of the encoder-decoder, we apply a position embedding based on previous work [69] [5]. A patch center is mapped to an embedding dimension via a learnable MLP, note however, that we use separate MLPs for the encoder and decoder respectively, which we will touch on in the next paragraph.

Encoder-Decoder Point-MAE's autoencoder backbone is based on a standard Transformer architecture, as it entirely consists of standard Transformer blocks [74]. Drawing inspiration from the MAE [33], it adopts an asymmetric encoder-decoder design. The motivation for using an asymmetric design is multifaceted. Firstly, a simpler decoder is beneficial with respect to computational efficiency, and since the decoder operates on a lower-dimensional latent space, it can afford to be less complex while still achieving good performance. Secondly, while a more complex decoder can potentially capture finer details, it also comes with a serious risk of overfitting. Adopting a lightweight decoder is therefore more beneficial, as it reduces the risk of overfitting while also promoting the learning of generalized features that are more useful for downstream tasks. Lastly, given that Point-MAE's primary objective is to learn high-quality latent representations suitable for downstream tasks, it is the encoder that requires a higher degree of complexity in order to capture the intricate geometry of the input data. This strategic imbalance allows Point-MAE to learn representations that are broadly applicable.
The encoder specifically only processes visible tokens, while the mask tokens T_m are strategically shifted to the decoder. This shift serves a dual purpose, influencing both the downstream performance and computational load. Firstly, shifting mask tokens to the decoder has been demonstrated to improve performance on downstream tasks [6]. The reason for this improved performance comes from the intricacies of the location information that is provided to all patches in the form of position embeddings. Presenting both token types to the encoder would lead to early leakage of location information, simplifying the reconstruction task. By presenting only visible tokens to the encoder, we avoid such leakage and also increase the difficulty of the reconstruction task, resulting in better latent features. The second purpose of shifting the mask tokens is a decrease in computational load [6],[33]. As we mask around 60 to 80 percent of the point cloud, shifting the mask tokens significantly reduces the

number of input tokens to the encoder. Due to the quadratic complexity of the Transformer, this leads to notable savings in computational resources.
To summarise, our encoder consists of standard transformer blocks, and only encodes visible tokens T_v, shifting the mask tokens as T_m to the decoder for later processing. The encoder produces encoded tokens T_e. Position embeddings are added to every Transformer block, providing location information based on the patch center. The decoder shares a similar architecture as the encoder, with the main difference being that it contains fewer Transformer blocks. It takes both the encoded tokens T_e and mask tokens T_m as input. Again, positional embeddings are added to every Transformer block, providing location information to all tokens. The decoder outputs the decoded mask tokens H_m, which are fed through a simple prediction had, that will be discussed in the following paragraph. The encoded-decoder structure can formally be formulated as:

$$T_e = \text{Encoder}(T_v), \quad T_e \in \mathbb{R}^{(1-m)n \times C}; \tag{11}$$

$$H_m = \text{Decoder}(\text{concat}(T_e, T_m)), \quad H_m \in \mathbb{R}^{mn \times C}. \tag{12}$$

Prediction Head The last layer of our backbone is a simple prediction head consisting of a single 1D convolution. The prediction head projects the output of the decoder H_m to a vector, which is subsequently reshaped to match the dimensionality of the original mask patches. This gives us the predicted masked point patches P_{pre}:

$$P_{\text{pre}} = \text{Reshape}(\text{FC}(H_m)), \quad P_{\text{pre}} \in \mathbb{R}^{mn \times k \times 3}. \tag{13}$$

Reconstruction Target The target of our reconstruction task is to recover the coordinates of every point within a given point patch. An effective metric is required to compare the similarity between the set of predicted points and the set of target points. This is done by creating a loss function based on Chamfer Distance [53]. Chamfer Distance essentially looks at two sets of points. For each point within each set, the algorithm finds the nearest neighbor in the other set and sums the squared distances up. Given the predicted point patches P_{pre} and ground truth P_{gt}, the reconstruction loss $\mathcal{L}$ is computed as:

$$\mathcal{L} = \frac{1}{|P_{\text{pre}}|} \sum_{a \in P_{\text{pre}}} \min_{b \in P_{\text{gt}}} \|a - b\|_2^2 + \frac{1}{|P_{\text{gt}}|} \sum_{b \in P_{\text{gt}}} \min_{a \in P_{\text{pre}}} \|a - b\|_2^2 \tag{14}$$

To summarize, Point-MAE consists of three main steps. In step one, centers within the point cloud are identified via the FPS algorithm, and subsequent patches are created via the KNN algorithm. Step two focuses on masking and embedding these patches. First, a large portion of the patches are masked and then embedded via a lightweight embedding module to transform the patches into a format suitable for the third and final step. The final step consists of an asymmetric autoencoder, where the encoder takes as input the mask tokens, and visible embedded patches are passed to the decoder. A simple prediction head aims to reconstruct the points in the masked patches. The outcome is that after sufficient training, the model has ideally learned a very good representation of the data. This allows us to use the pre-trained Point-MAE in the subsequent step.
The subsequent section will focus on the second step of the anatomical structure identification pipeline, as displayed in Figure 5. In this step, the pre-trained Point-MAE model is used to create features for the graph nodes.

3.3 Graph Construction

The motivation for incorporating graph-based prediction into the anatomical structure identification pipeline is twofold. Firstly, it aims to provide enhanced structural understanding through the connection between nodes. The human anatomy is inherently a connected 3D structure, which can be naturally represented as a graph. Secondly, it aims to enhance robustness and generalization. Human physiques are broadly consistent but also exhibit notable variations from person to person. The graph provides consistency across individuals. This consistency makes the system more robust to physiological variations and challenging conditions such as occlusion.
This section details the graph construction process and explains how point clouds are transformed into a graph format. This graph representation of the point cloud aims to encapsulate both rich spatial information through the features generated by the Point-MAE encoder, as well as hierarchical and relational information through the graph structure. This graph format has been introduced in Section 3.1, and is referred to as GRAPE. Figure 16 provides a high-level overview of the main steps involved in the graph construction process. This section will start with a high-level explanation of the graph construction step, followed by a more formal definition.
GRAPE represents each point cloud as a hierarchical graph. The base structure of this graph must be predefined before processing any point cloud, which allows for consistent modeling across samples of a singular object type. The predefined graph ensures consistency across the generated samples, as each generated graph is based on the same default graph. Not all point clouds contain every possible part, and the relative size of a part might differ per point cloud. This results in the actual variations between the generated graph. For example, a table point cloud may lack a drawer. In such cases, the resulting graph will omit the corresponding nodes for the missing parts. The initial node features of the graph are generated by a pre-trained Point-MAE encoder. This encoder is designed to create latent representations that are consistent across corresponding structures in different instances. For example, if the encoder processes a patch representing an elbow from various human figures, it should produce similar latent features for each elbow patch, regardless of the specific individual.
Figure 16 illustrates a four-step process: patch generation, embedding, encoding, and graph generation. The first three steps represent a pre-trained Point-MAE configured for inference. The version of Point-MAE used in the graph generation process has two main adjustments compared to the earlier version of Point-MAE shown in 15. First, the masking and embedding step, now omits masking, as we want to encode the complete sample. Second, the autoencoder now omits the decoder, as we use the output generated by the encoder as node features. Once the encoder has generated the features T_e, the hierarchical undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, T, Y)$ is constructed, where:

- $\mathcal{V}$ is the set of nodes.
- $\mathcal{E}$ is the set of undirected edges
- T is the node feature matrix, where each node $v \in \mathcal{V}$ has an associated feature vector t_v originating from the Point-MAE encoder.
- Y is the label matrix.

Each feature vector t_v represents the latent representation of a single-point patch. For each feature vector created by the encoder, a new leaf node is generated. This maintains a consistent feature vector sizes across nodes, which is required by the GNN. For non-leaf nodes, the feature vector is computed as the mean of its children's feature vectors. Formally, for a non-leaf node v, its feature vector $\overline{t_e}$ is computed as $\overline{t_v} = \frac{1}{|C(v)|} \sum_{u \in C(v)} t_u$ where $C(v)$ is the set of child nodes for v. The mean operation

was selected as an initial aggregation method, but other methods such as max pooling or weighted averages could potentially also be viable.
This subsection detailed how point clouds are encoded and transformed into a graph format called GRAPE. The idea of this graph representation is to encapsulate both rich spatial information through the features generated by the Point-MAE encoder, as well as hierarchical and relational information of the object components through the graph structure.

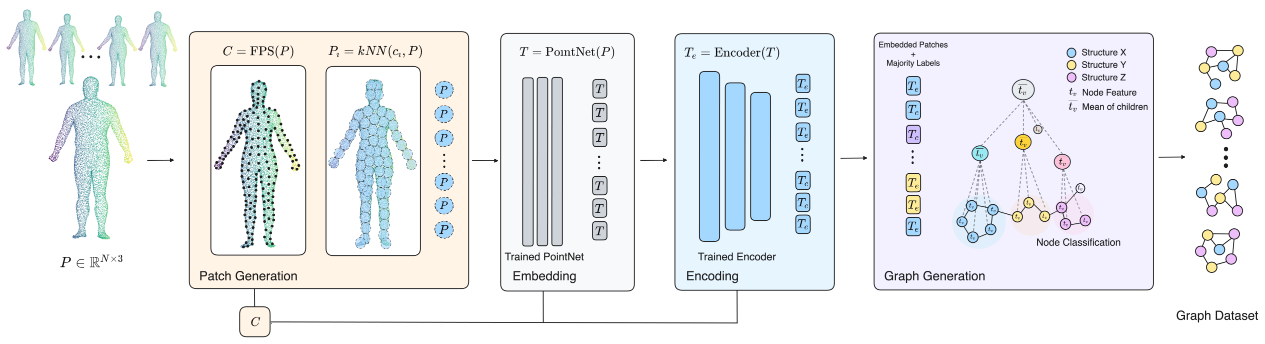
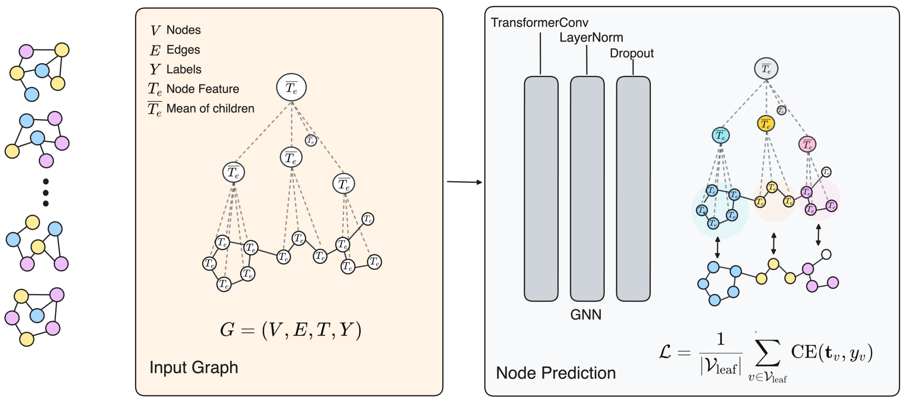


Figure 16: Graph Dataset Generation.

3.4 Graph-Based Anatomical Structure Identification



V Nodes
E Edges
Y Labels
T_e Node Feature
$\overline{T_e}$ Mean of children
$G = (V, E, T, Y)$
Input Graph
TransformerConv
LayerNorm
Dropout
GNN
Node Prediction
$\mathcal{L} = \frac{1}{|\mathcal{V}_{\text{leaf}}|} \sum_{v\in\mathcal{V}_{\text{leaf}}} \text{CE}(\mathbf{t}_v, y_v)$

Figure 17: Graph Neural Network.

Following graph construction, we employ a GNN to leverage both the node features generated by Point-MAE and the inter-node relationships. GNNs are uniquely suited to operate on graph-structured data. They are able to process both the features, as well as the relational and hierarchical information within the graph structure. In this section, we will present GRAPE-GNN, our custom GNN used for both table part identification and anatomical structure identification.

GRAPE-GNN GRAPE-GNN is a simple and effective GNN architecture. Its simplicity is characterized by its shallow design and limited set of operations. Despite its simplicity, GRAPE-GNN demonstrates promising performance, as presented in Chapter 5. At the heart of GRAPE-GNN is the graph transformer operator, which integrates the transformer mechanism into the GNN. Layer normalization and dropout help to stabilize training and prevent overfitting. GRAPE-GNN is trained to perform leaf node prediction, as the leaf nodes within the hierarchical graphs represent the actual parts.

Input GRAPE-GNN takes as input a complete hierarchical undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, T, Y)$. Here $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of undirected edges. T is the node feature matrix where each node $v \in \mathcal{V}$ has an associated feature vector t_v that originates from the Point-MAE encoder. Lastly, Y is the label matrix. More specifically, the node feature matrix is of the format $\mathbf{T} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where d is the dimension of each node embedding. The edge list is in the format of $\mathcal{E} \in \mathbb{R}^{2 \times |\mathcal{E}|}$, where each column represents an undirected edge between two nodes (v_i, v_j). The labels matrix is of the format $Y \in \mathbb{R}^{|\mathcal{V}| \times c}$, where c is the number of classes, and y_i represents the label for node v_i.

Graph Transformer Graph Attention Networks (GAT) were one of the earliest attempts to incorporate an attention mechanism into GNNs [56]. It was quickly discovered that GATs compute a very limited kind of attention, where the ranking of the attention scores is unconditioned on the query node. GATv2 resolved this by introducing dynamic attention, where the attention scores are conditioned on the query node, making it strictly more expressive than GAT [75]. More recently, there is the Graph Transformer, which uses multi-head dot product attention, where attention coefficients are computed via softmax applied to the dot product of transformed node features [62]. The graph transformer can potentially be more expressive than both GAT and GATv2 due to its use of multi-head dot product attention. Preliminary experiments indicated that the graph transformer was the superior option, leading to its selection as the primary operator in GRAPE-GNN. However, these initial tests were exploratory in nature and not conducted with the methodological rigor required for formal inclusion in this thesis. The graph transformer operator works in the following way. For each node $v \in \mathcal{V}$, there is an initial node embedding $t_v^{(0)}$. The embeddings are then updated according to the following update rule. For k = 1, 2, $\dots, k$:

$$\mathbf{t}_v^{(k)} = \mathbf{W}_1 \mathbf{t}_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} \alpha_{vu} \, \mathbf{W}_2 \mathbf{t}_u^{(k-1)} \tag{15}$$

Here $t_v^{(k)}$ is the embedding of node v at step k, $\mathcal{N}(v)$ denotes the set of neighboring nodes of v, $\mathbf{W}_1$ and $\mathbf{W}_2$ are the learnable weight matrices, and α_{vu} are the attention coefficients. The attention coefficients are computed via the earlier described multi-head dot product attention, according to:

$$\alpha_{vu} = \text{softmax}\left(\frac{(\mathbf{W}_3 \mathbf{t}_v)^\top (\mathbf{W}_4 \mathbf{t}_u)}{\sqrt{d}} \right) \tag{16}$$

Here, $\mathbf{W}_3$ and $\mathbf{W}_4$ are the learnable weight matrices and d is the dimension of the embedding.

Layer Normalization and Dropout GRAPE-GNN applies Layer normalization and dropout after each hidden transformer layer. Layer normalization ensures that the activations of each layer maintain a mean of zero and a standard deviation of one. This helps stabilize and accelerate the training process. Layer normalization is defined as:

$$\mathbf{t}_v' = \frac{\mathbf{t}_v - \mathbb{E}[\mathbf{t}_v]}{\sqrt{\text{Var}[\mathbf{t}_v] + \varepsilon}} \odot \gamma + \beta \tag{17}$$

Here $\mathbf{t}_v$ is the input feature vector for node v, $\mathbb{E}[t_v]$ is the mean of the input features for node v. $Var[t_v]$ is the variance of the input features for node v, and ε is a small constant for numerical stability. γ is a learnable scaling parameter, and β is a learnable shifting parameter. Dropout is added to prevent overfitting. During training, we randomly zeroed some of the elements of the input tensor with probability `p`. The zeroed elements are chosen independently for each forward call and are sampled from a Bernoulli distribution.

Node Classification The final layer of GRAPE-GNN focuses on classifying the leaf nodes of the graph, as these represent the actual anatomical parts. Intermediate nodes in the graph are present to provide structural and global information. A mask that filters out all non-leaf nodes is applied to each graph before making predictions, allowing the model to focus on relevant nodes for classification. The loss is computed via standard cross-entropy loss, taking only leaf nodes into consideration:

$$\mathcal{L} = \frac{1}{|\mathcal{V}_{leaf}|} \sum_{v \in \mathcal{V}_{leaf}} \text{CrossEntropy}(\mathbf{h}_v, y_v) \tag{18}$$

Where V_{leaf} is the set of leaf nodes, $\mathbf{h}_v$ is again the node embedding for node v and y_v is the true label of node v.
This concludes the methodology. This chapter started with an explanation of the PartNet and CAESAR datasets for SSL pre-training of Point-MAE. It then explained the graph construction and GRAPE-GNN for anatomical structure identification. The next Section will detail the experimental setup, the main experiments conducted to validate our method, and answer the research questions posed in Section 1.

4 Experimental Setup

This Chapter covers a description of the experiments designed to evaluate our methodology and answer the research questions. It starts with a general overview of the technological framework, followed by a description of the data setup and an hyperparameter overview. After this, the section continues with a description of the five main experiments, detailing their objectives, motivations, and performance metrics used to evaluate the results.

4.1 Technological Framework

The models and experiments detailed in this thesis have been implemented in the `Python` programming language, with small functionalities written in C++ for speed improvements. `Pytorch` [76] has been used as the main deep learning framework for implementing the the models. `Pytorch Lightning` [77] has been used to reduce boilerplate Pytorch code and simplify distributed training. `Weights and Biases` [78] has been used for experiment tracking. All training and inference has been done on two NVIDIA H100 pcie GPUs, which have been provided by Nedap.

4.2 Dataset Setup

As stated in Section 3.1, four datasets are used. The datasets used for SSL pre-training are TableNet and CAESAR. Information on the data splits applied to these datasets can be seen in Table 1. The datasets used for training of the GNN are Table-GRAPE and Anatomy-GRAPE. Information on the data split applied to these datasets can be seen in Table 2. For all experiments, we consistently employed an 80/15/5 split for training, validation, and testing, respectively.

Dataset	Total Samples	Train Samples	Validation Samples	Test Samples
TableNet	9906	7925	1486	495
CAESAR	3000	2400	450	150

Table 1: Dataset splits for TableNet and CAESAR.

Dataset	Total Samples	Train Samples	Validation Samples	Test Samples
Table-GRAPE	5159	3714	929	516
Anatomy-GRAPE	80	57	15	8

Table 2: Dataset splits for Table-GRAPE and Anatomy-GRAPE.

4.2.1 Hyperparameter overview

Table 3 presents the hyperparameters used for the training of Point-MAE and GRAPE-GNN. Point-MAE utilized identical hyperparameters for both TableNet and CAESAR datasets. These parameters were derived from the original Point-MAE implementation [6], which demonstrated good performance in their study. For GRAPE-GNN, minor adjustments to the hyperparameters were necessary when transitioning from Table-GRAPE to Anatomy-GRAPE training. These modifications, primarily in batch size and number of classes, were implemented to accommodate the smaller scale of the Anatomy-GRAPE dataset.

Point-MAE Hyperparameters		GRAPE-GNN Hyperparameters		
Hyperparameter	Value	Hyperparameter	Table-GRAPE	Anatomy-GRAPE
Optimizer Type	AdamW	Optimizer Type	Adam	Adam
Learning Rate	0.001	Learning Rate	0.001	0.001
Weight Decay	0.05	GNN Input Channels	16	16
Scheduler Type	Cosine Annealing WR	GNN Hidden Channels	64	64
Group Size	32	Number of Classes	22	12
Number of Groups	256	Heads	6	6
Loss Function	l_2 Chamfer Distance	Heads in Final Conv	4	4
Mask Ratio (%)	60	Dropout (%)	50	50
Mask Type	random	Depth	2	2
Transformer Dimension	384	Batch Size	128	8
Encoder Dimensions	384			
Depth	22			
Drop Path Rate (%)	10			
Decoder Depth	4			
Decoder Heads	6			
Number of Points (npoints)	4096			
Batch Size	128			
Early Stopping	False	Early Stopping	Patience = 8	Patience = 8

Table 3: Hyperparameter Overview for Training Point-MAE and GRAPE-GNN.

4.3 Main Experiments

We designed five main experiments to validate the proposed method and address the research questions outlined in Chapter 1. Experiment one and three focused on Point-MAE and the quality of its learned latent representations, while experiments two, four and five examined GRAPE-GNN. All experiments employed a training, validation, and testing split. SSL experiments utilized a cosine annealing with warm restarts and deliberately omitted early stopping. This is because warm restarts introduce fluctuations in the loss, where early stopping might prematurely halt training before the model fully exploits the lower learning rates to refine the latent space. In contrast to Point-MAE, all GRAPE-GNN experiments implemented early stopping on the validation loss with a patience of 8 epochs. The following sections provide a brief overview and motivation for each experiment.

Experiment 1: Point-MAE The first experiment was designed to validate our Point-MAE implementation. We trained Point-MAE on both TableNet and CAESAR datasets, analyzed training metrics and visualized predictions for each. To further assess the quality of learned representations, we generated a t-SNE embedding plot of the Point-MAE embeddings for the annotated CAESAR dataset, with points colored according to their corresponding anatomical structure. Due to time constraints the t-SNE plot was only created for CAESAR. Additionally, due to the extremely long training times of our Point-MAE model, which is around 3 days on a single H100 pcie GPU, it was unfeasible to a run a k-fold cross validation experiment. Hyperparameters remained consistent across both dataset experiments, as detailed in Table 3. This experiment was designed to address Research Question 2.

Experiment 2: Structure Prediction The second experiment evaluated GRAPE-GNN's structure prediction capabilities. We conducted 5-fold cross-validation on both Table-GRAPE and Anatomy-GRAPE datasets, generating train, validation, and test metrics, along with comprehensive classification reports (precision, recall, and F1 scores for each class). This experiment was designed to address Research Question 1. Additionally, two sub-experiments were also performed:

- Experiment 2a) Training Set Size Impact: We trained GRAPE-GNN on incrementally smaller subsets (100-5%) of the Table-GRAPE dataset to assess performance relative to training data volume. This aimed to give insight in the minimum CAESAR data annotation required for acceptable performance.
- Experiment 2b) Structure Size vs. Performance: Using the Anatomy-GRAPE dataset, we analyzed the relationship between anatomical structure size and GRAPE-GNN performance.

Experiment 3: Node Embedding Noise The third experiment assessed the effectiveness and quality of Point-MAE embeddings as node features. We conducted 5-fold cross-validation training of GRAPE-GNN on both Table-GRAPE and Anatomy-GRAPE datasets, systematically introducing noise to the node embeddings. Specifically, 20, 40, 60 or 80 percent of each embedding was replaced with random noise sampled from a standard normal distribution. This experiment was designed to address Research Question 2.

Experiment 4: GNN vs. MLP The fourth experiment assessed the effectiveness of graph-based structure prediction compared to direct prediction on the Point-MAE embeddings. To ensure a fair comparison, a MLP was designed with a similar number of trainable parameters to GRAPE-GNN. This MLP served as a baseline model, that had no access to the graph structure, unable to leverage available connections between nodes. It is worth noting that a truly fair comparison, where both models are of the exact same capabilities, is not feasible due to the inherent differences between GNNs and MLPs. A 5-fold cross-validation experiment was conducted and compared against the GRAPE-GNN results from Experiment 2. Experiment 4 was designed to address Research Question 3.

Experiment 5: Point Cloud Occlusion The fifth and final experiment aimed to asses the robustness of GRAPE-GNN to point cloud occlusion. In this experiment, one large connected part, comprising 42% percent was removed before feeding the point cloud to the anatomical structure identification pipeline. This aimed to simulate occlusion. A 5-fold cross-validation experiment was conducted and the results were compared against the results from Experiment 2. This experiment was designed to answer Research Question 1, specifically with respect to robustness.
This concludes the experimental setup. The following section will present the results for the discussed experiments.

5 Results

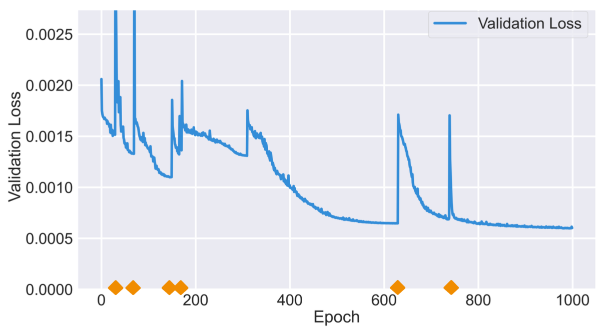
This chapter presents the results of the experiments described in Chapter 4. Each subsection corresponds to a specific experiment and is divided into two parts:

1. Results for TableNet (SSL experiments) or Table-GRAPE (GNN experiments).
2. Results for CEASAR (SSL experiments) or Anatomy-GRAPE (GNN experiments).

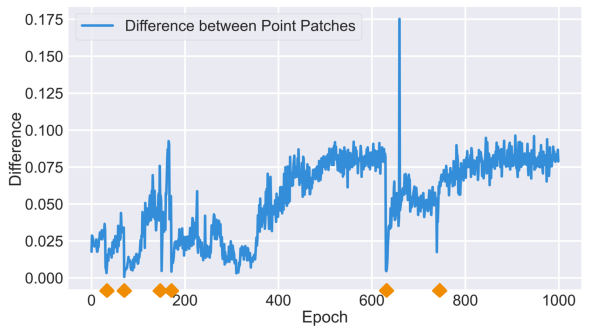
5.1 Point-MAE SSL

This section presents the results obtained in experiment 1, which focused on training Point-MAE on both the TableNet and CAESAR datasets. First, we present the results for Point-MAE trained on TableNet, after that we present the results for Point-MAE trained on CAESAR.

TableNet Figure 18 presents two key metrics used to evaluate Point-MAE. Figure 18a shows the validation loss over training epochs, and Figure 18b shows the modified coefficient of variation (M-CV) for the set of predicted point patches, measuring how much the distances within these patches differ from each other relative to their mean. The motivation for using this metric is that Point-MAE tends can get stuck in local minima that result in highly similar point patches. However, for accurate point cloud reconstruction variation across these patches is required. An increasing M-CV indicates the model is improving this capability. Both figures contain orange indications on the x-axis. These aim to indicate points of interest, where the loss spikes up, and the M-CV spikes down. Figure 18a shows a validation loss curve that generally decreases and plateaus near the end. Intermittent positive spikes are followed by sharp declines back to pre-spike levels, and a gradual decrease and eventual stabilization. Figure 18b shows an M-CV curve that generally increases and also plateaus near the end of training. Occasional drops in M-CV can be observed.



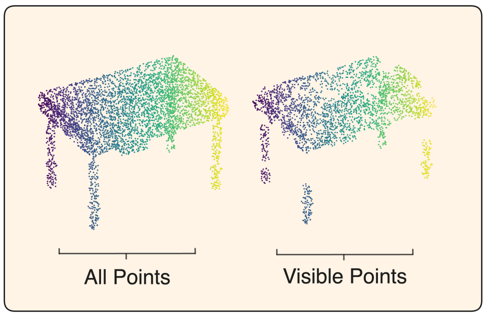
(a) Validation loss for Point-MAE training on TableNet, with critical points of interest highlighted in orange along the x-axis.



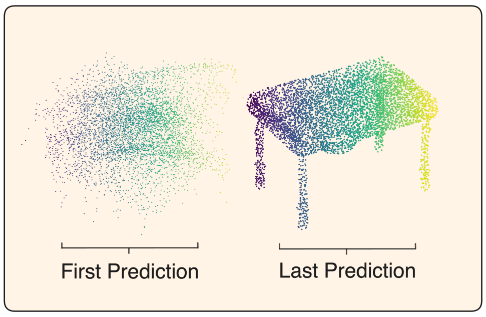
(b) Inter patch M-CV for Point-MAE training on TableNet, with critical points of interest highlighted in orange along the x-axis.

Figure 18: Point-MAE results on TableNet.

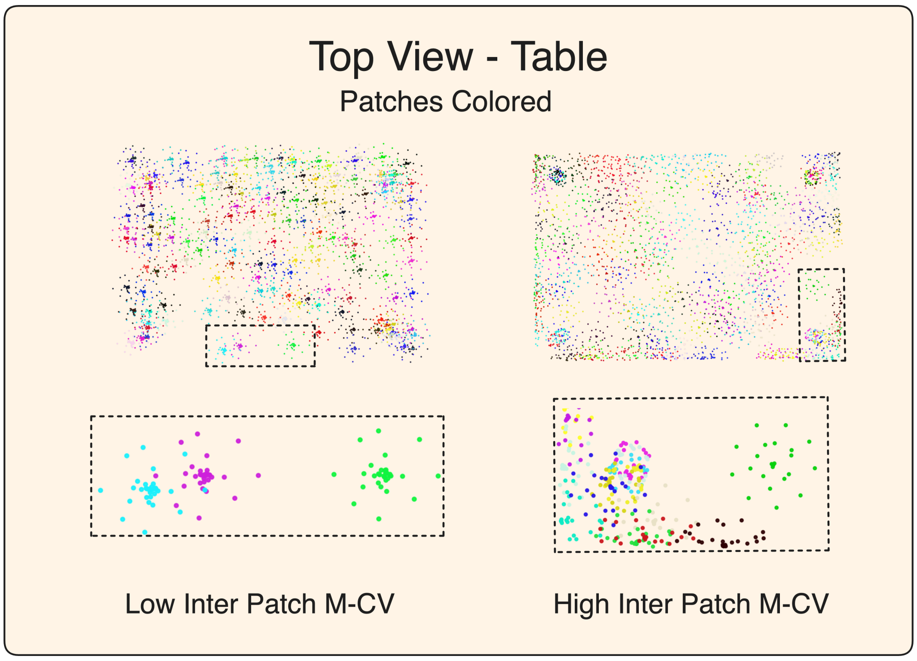
Figure 19a illustrates a sample input point cloud for Point-MAE on TableNet data, depicting both the complete and masked versions. Figure 19b demonstrates Point-MAE's predictive capability, contrasting its initial and final predictions. The initial prediction shows an unstructured point distribution, while the final prediction shows a distinct table shape. The point colors in Figures 19a and 19b are merely for easier depth interpretation and have no further meaning. Figure 19c, shows a top-down view of the Table at two distinct moments, corresponding to a low and high M-CV. In this figure, each predicted point patch is colored, such that the different patches can be distinguished. Upon closer inspection of the enlarged sections, it can be seen that low M-CV periods are characterized by highly similar point distributions across patches, whereas high M-CV periods show substantial inter-patch variation.



(a) Point-MAE input example: TableNet.



(b) Point-MAE output example: TableNet.

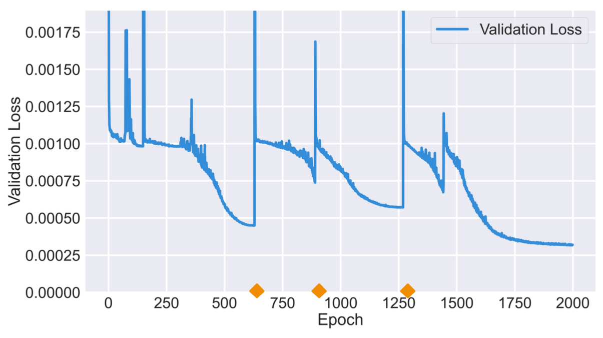


(c) Predicted patch variation visualized (predicted patches colored).

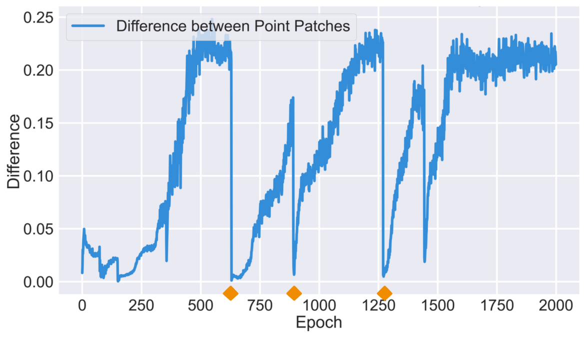
Figure 19: Point-MAE input/output visualized: TableNet.

CAESAR Figure 20 shows the validation loss and M-CV for Point-MAE training on the CAESAR dataset. The validation curve in Figure 20a resembles the loss curve for TableNet, displaying a general decrease with intermittent spikes and eventual stabilization. The M-CV curve in Figure 20b can be seen to rise to the same level three times, where the first two are followed by a steep decline and then a rapid recovery to a stable level. Orange indication on the x-axis can be seen in this figure as well, indicating points of interest where spikes occur.
Figure 21a illustrates a sample input point cloud for Point-MAE on CAESAR data, depicting the completer and masked version. Figure 21b demonstrates Point-MAE's predictive capabilities for the

CAESAR data, contrasting the first and last predictions. Just like for TableNet, the initial prediction shows an unstructured cloud of points, while the final prediction shows a shape distinctly resembling the complete input.

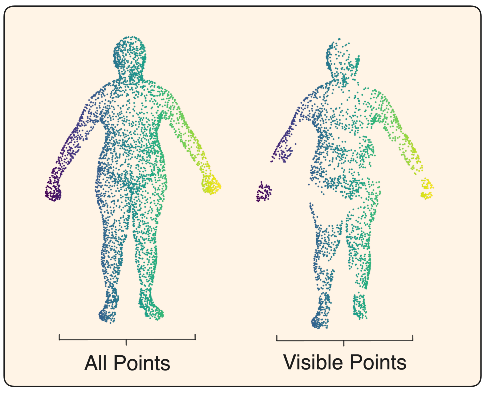


(a) Validation loss for Point-MAE training on CAESAR, with critical points of interest highlighted in orange along the x-axis.

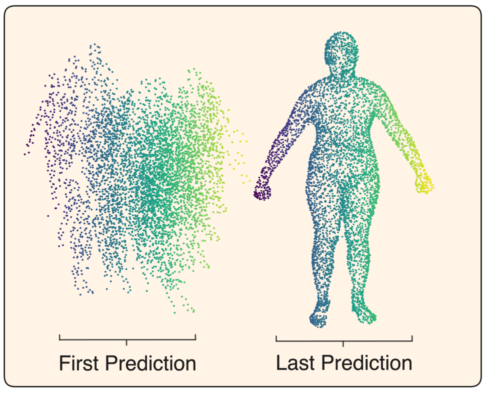


(b) Inter patch M-CV for Point-MAE training on CAESAR, with critical points of interest highlighted in orange along the x-axis.

Figure 20: Point-MAE results on CAESAR dataset.



(a) Point-MAE input example: CAESAR.



(b) Point-MAE output example: CAESAR.

Figure 21: Point-MAE input/output visualized: CAESAR.

Figure 22 shows an embedding plot of the Point-MAE embedding for the annotated CAESAR data. This plot is the result of a Principal Component Analysis (PCA), to reduce the embeddings to a reasonable size, followed by t-distributed Stochastic Neighbor Embedding (t-SNE). Each embedding is colored according to its label. Clear clusters can be seen for some classes, such as Calves, Hamstrings, and Quadriceps, while other clusters such as biceps and triceps slightly overlap.

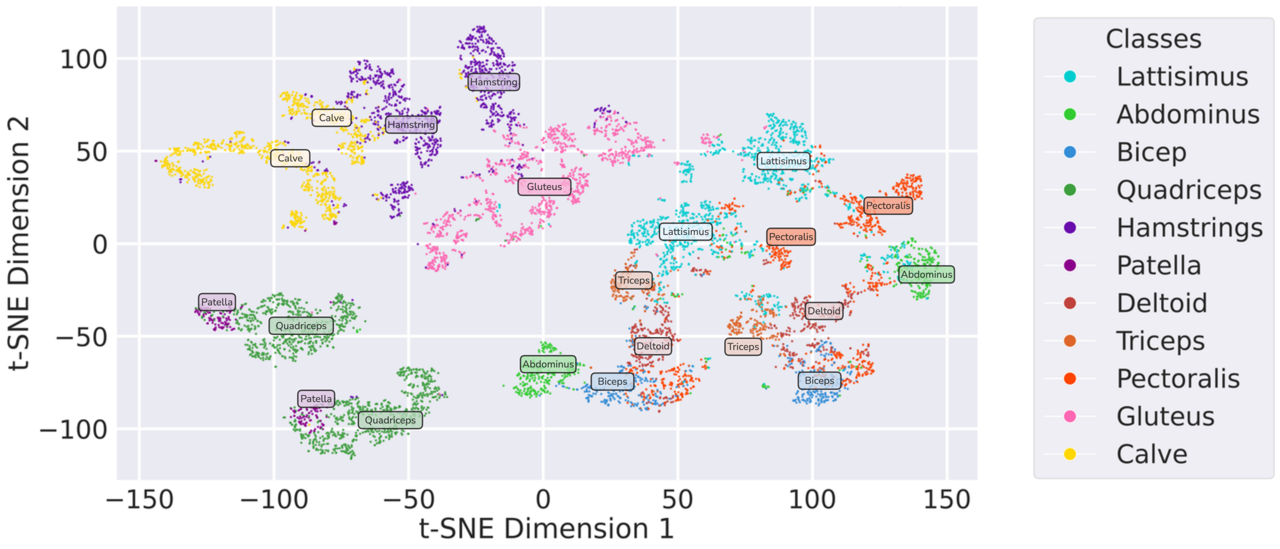


Figure 22: t-SNE visualization of Point-MAE patch embeddings: CAESAR.

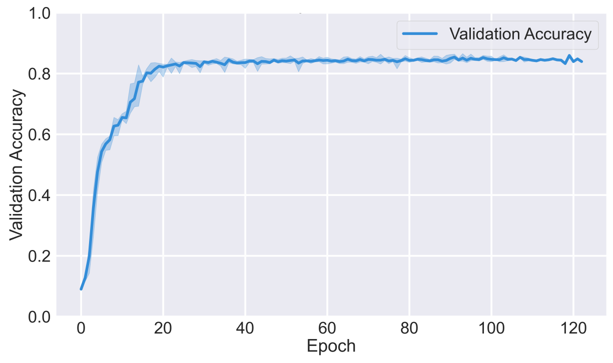
5.2 Structure Prediction

This section presents the results obtained in experiment 2, which focused on validating GRAPE-GNN's structure prediction capabilities on both the Table-GRAPE and Anatomy-GRAPE datasets. First, we present the results for GRAPE-GNN trained on Table-GRAPE, after which we continue with the results for Anatomy-GRAPE.

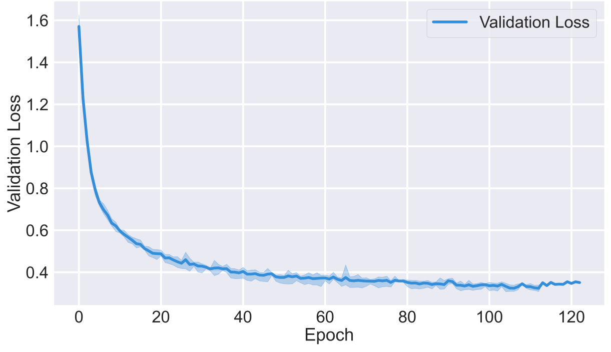
Balanced Accuracy		
Train	Validation	Test
0.883	0.859	0.885

Table 4: Best performance across train, validation, and test data of GRAPE-GNN on Table-GRAPE.

Table-GRAPE Figure 23 depicts the balanced validation accuracy and validation loss for GRAPE-GNN trained on Table-GRAPE. Balanced accuracy, defined as the mean recall across all classes, is used to mitigate the potential bias introduced by class imbalance when inspecting the results. The spread in both plots indicates the minimum and maximum range of the metric across the 5-fold cross-validation experiment. From Figure 23a it can be seen that the line follows a sharp ascend towards ± 0.8 followed by a slower ascend and plateau to 0.859. This trend is stable across all five folds, which can be seen from the narrow spread. From Figure 23b it can be seen that the loss also follows a stable trend, similar across folds. A sharp decline is followed by a more gradual movement towards a plateau level. Table 4 shows the best accuracies achieved during training, validation, and testing.
For a more detailed overview of GRAPE-GNN's predictive performance, Table 5 can be consulted for the precision, recall, and f-1 score for each class in Table-GRAPE. Additionally, the table also includes a column for the support of each class, which is represented in percentages. The table is ordered from high to low support. Overall good performance is observed. Notable results include the Glass class, which demonstrates poor performance. Central Support, Runner, and Miscellaneous demonstrate extremely high precision, recall, and f-1.



(a) Balanced validation accuracy of GRAPE-GNN on Table-GRAPE. Mean of 5-fold CV with shaded area representing min-max range.



(b) Validation loss of GRAPE-GNN on Table-GRAPE. Mean of 5-fold CV with shaded area representing min-max range.

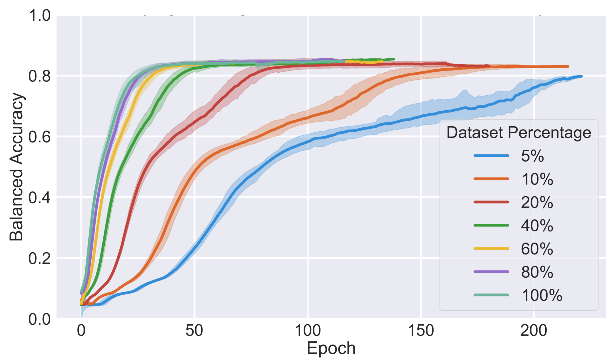
Figure 23: GRAPE-GNN performance on Table-GRAPE.

Label Name	Precision	Recall	F1-Score	Support (%)
Board	0.86	0.97	0.91	46.1
Leg	0.95	0.98	0.97	16.96
Tabletop Connector	0.98	0.99	0.98	7.41
Glass	0.6	0.18	0.25	5.94
Bar	0.78	0.43	0.55	4.19
Bar Stretcher	0.89	0.89	0.89	3.05
Vertical Side Panel	0.82	0.73	0.77	2.15
Pedestal	0.95	0.92	0.93	1.97
Central Support	1.0	1.0	1.0	1.77
Bottom Panel	0.84	0.85	0.85	1.43
Runner	1.0	0.99	1.0	1.27
Back Panel	0.78	0.79	0.78	1.11
Shelf	0.85	0.86	0.86	1.11
Foot	0.93	0.82	0.87	0.91
Drawer Front	0.94	0.91	0.92	0.78
Circle	0.89	0.63	0.73	0.75
Miscellaneous	1.0	1.0	1.0	0.62
Vertical Front Panel	0.81	0.72	0.76	0.61
Drawer Bottom	0.9	0.96	0.93	0.55
Cabinet Door Surface	0.85	0.85	0.85	0.47
Drawer Side	0.93	0.9	0.91	0.46
Vertical Divider Panel	0.97	0.93	0.95	0.39

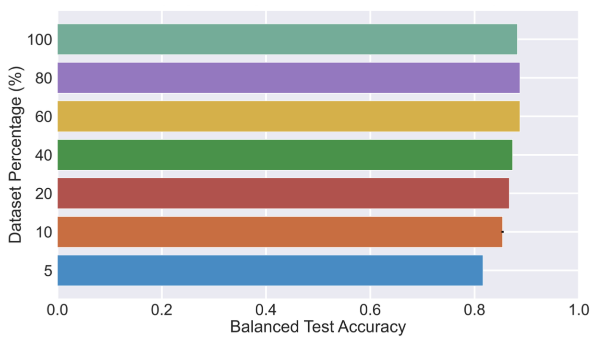
Table 5: GRAPE-GNN per-class performance on Table-GRAPE.

Figure 24 illustrates the results for Experiment 2a, showing the (balanced) validation accuracy and loss when training GRAPE-GNN on 5, 10, 20, 60, 80, and 100 percent of Table-GRAPE. From

Figure 24a it can be seen that as GRAPE-GNN is trained on smaller datasets the steepness of the curve towards the plateau level seems to decrease. However, when interpreting this it is important to recognize that a smaller dataset means fewer gradient steps per epoch. It can also be seen that the spread increases as GRAPE-GNN is trained on less data. Nonetheless, the model is still able to achieve similar performance at the end of training, with all accuracy curves converging near of around 85%. Figure 24b displays the test accuracies for experiment 2a. A subtle but clear trend can be observed when training on 60% or less, showing a decrease in test accuracy as the training data is reduced. The same test set was used across all experiments.



(a) Balanced validation accuracy of GRAPE-GNN on Table-GRAPE at decreasing training set sizes. Mean of 5-fold CV with shaded area representing min-max range.



(b) Balanced test accuracy of GRAPE-GNN on Table-GRAPE at decreasing training set sizes. Mean of 5-fold CV.

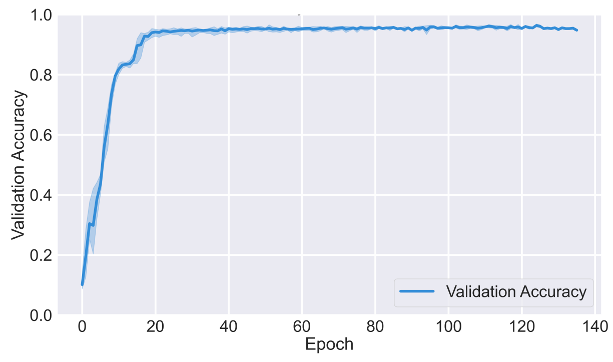
Figure 24: Results of training GRAPE-GNN on decreasing training set sizes of Table-GRAPE.

Anatomy-GRAPE Figure 25 presents the balanced validation accuracy, validation loss, and the results for experiment 2b for GRAPE-GNN trained on Anatomy-GRAPE. Figure 25a shows the balanced validation accuracy, which is characterized by a sharp initial increase followed by a long-tailed progression, ending in a final accuracy of 0.987. The curve displays a narrow spread, being more pronounced at the beginning and minimal towards the end. Figure 25b illustrates a clean loss curve with a sharp initial decrease, stabilizing around epoch 50, and gradually plateauing with minimal spread. Figure 25c displays a scatter plot of the per-class test accuracies. Each point represents an individual class, with the x-axis denoting the F1 score and the y-axis indicating anatomical structure size (proportional to support). A blue regression line with a shaded confidence interval is included. The line reveals the positive correlation between balanced test accuracy and structure size. Notable is that the patella, despite being the smallest structure, achieves remarkably high classification accuracy

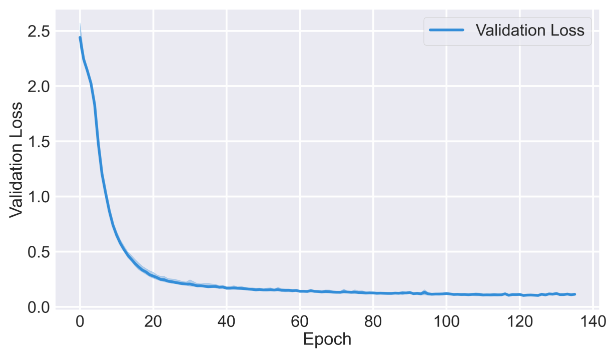
Table 6 details the highest balanced accuracies GRAPE-GNN achieved for the training, validation, and test sets. The model achieved a test accuracy of 0.967, close behind the training and validation accuracies of 0.987 and 0.969 respectively. Table 7 provides a more detailed report on the per-class accuracies. Overall high precision and recall are achieved. No class exhibits extremely low performance, however, the Gluteus class shows remarkably high performance, achieving a precision and recall of 1.0. The potential reason for this high performance is discussed in Section 6.2.

Balanced Accuracy		
Train	Validation	Test
0.987	0.969	0.967

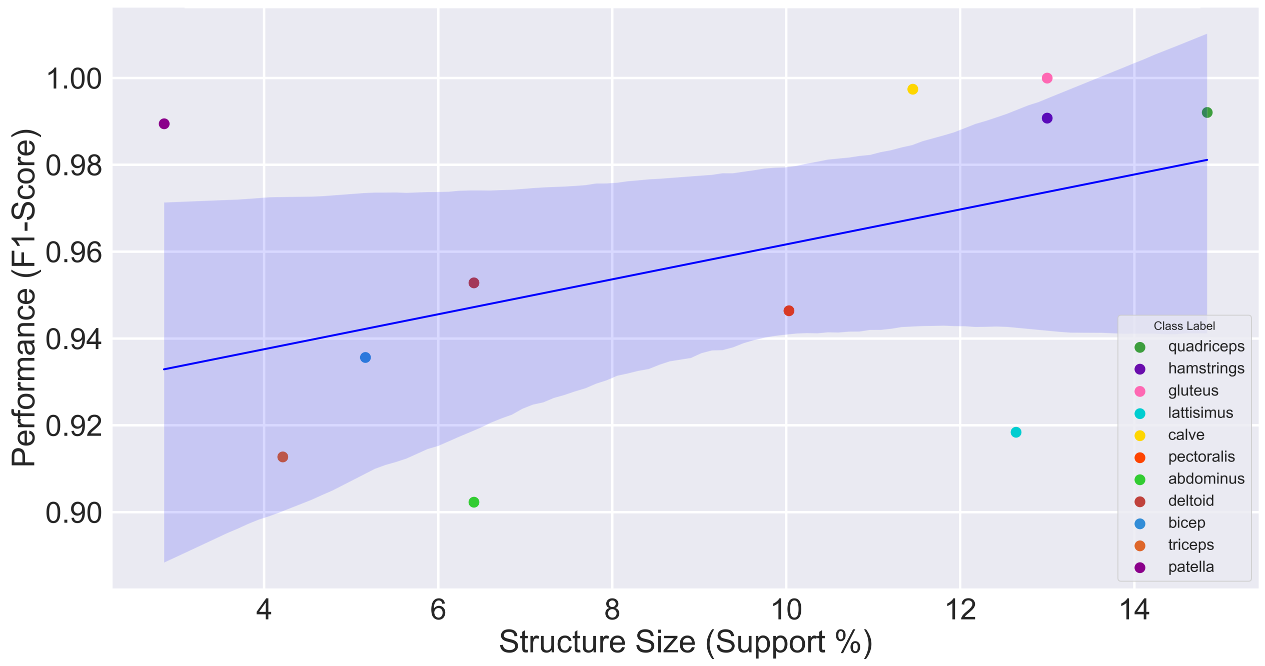
Table 6: Best performance across train, test, and validation data of GRAPE-GNN on Anatomy-GRAPE.



(a) Balanced validation accuracy of GRAPE-GNN on Anatomy-GRAPE. Mean of 5-fold CV with shaded area representing min-max range.



(b) Validation loss of GRAPE-GNN on Anatomy-GRAPE. Mean of 5-fold CV with shaded area representing min-max range.



(c) Relationship between structure size and GRAPE-GNN performance: Anatomy-GRAPE.

Figure 25: GRAPE-GNN performance on Anatomy-GRAPE.

Label Name	Precision	Recall	F1-Score	Support (%)
Quadriceps	0.984	1.000	0.992	14.840
Hamstrings	1.000	0.982	0.991	13.000
Gluteus	1.000	1.000	1.000	13.000
Lattisimus	0.912	0.925	0.918	12.640
Calve	0.995	1.000	0.997	11.450
Pectoralis	0.952	0.941	0.946	10.030
Abdominus	0.907	0.898	0.902	6.410
Deltoid	0.971	0.935	0.953	6.410
Bicep	0.952	0.920	0.936	5.160
Triceps	0.872	0.958	0.913	4.210
Patella	1.000	0.979	0.990	2.850

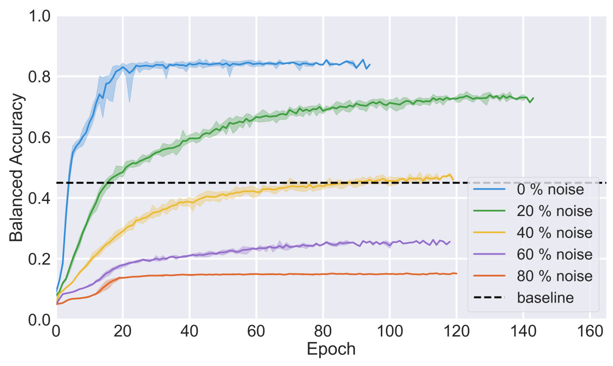
Table 7: GRAPE-GNN per-class performance on Anatomy-GRAPE.

5.3 Node Embedding Noise

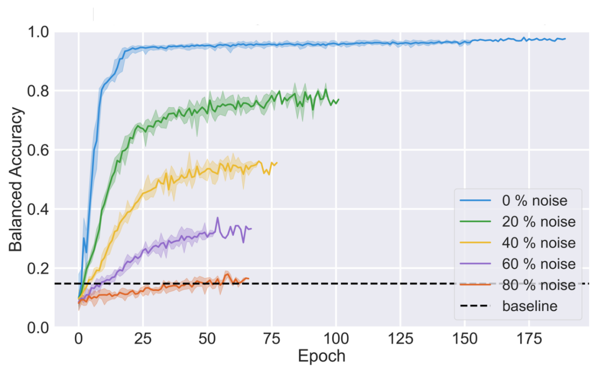
This section presents the results obtained in experiment 3, which focused on assessing the effectiveness and quality of Point-MAE embeddings as node features. Increasing amounts of noise were added to the node embeddings (20, 40 60, 80) and performances were compared. We first present the results for Table-GRAPE, followed by the results for Anatomy-GRAPE.

Table-GRAPE Figure 26 presents plots for the balanced validation accuracy, loss, and balanced test accuracy for GRAPE-GNN trained on Table-GRAPE with noise levels ranging from 0 to 80 percent. A dashed line represents the baseline performance, which corresponds to the majority class. Figure 26a shows the performance of GRAPE-GNN decreasing as noise increases. The spread of each curve remains consistent across different noise levels. At 40% noise, the performance drops to the baseline level. Figure 26c shows a clear and consistent increase in loss as the noise increases. Lastly, Figure 26e shows the same behavior in the test accuracy, a consistent drop in performance as the noise level increases, dropping from around 0.885 with 0% noise, to around 0.15 with 80% noise.

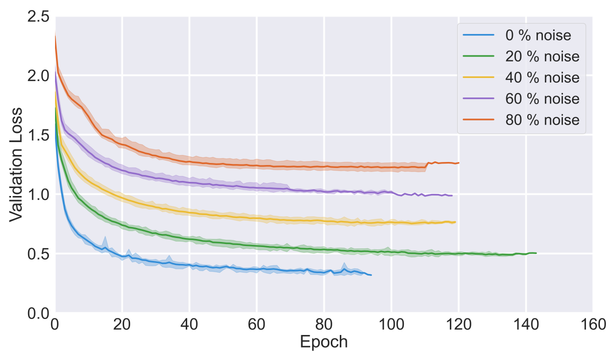
Anatomy-GRAPE Figure 26 presents plots for the balanced validation accuracy, loss, and balanced test accuracy for GRAPE-GNN trained on Anatomy-GRAPE with noise levels ranging from 0 to 80 percent. The curves in these Figures show similar behavior as could be seen in Table-GRAPE. Figure 26b shows a consistent drop in performance as the noise increases, as well as a less steady curve for all experiments with noise. These also show more spread, which was less noticeable for Table-GRAPE. Additionally, training seems to halt sooner as the noise level increases, a feature that was also less noticeable for Table-GRAPE. Figure 26d shows the loss curves with decreasing steepness and increasing loss as noise increases. Lastly, Figure 26d shows the same test accuracy behavior as previously seen in the Table-GRAPE results. A consistent drop in accuracy from around 0.967 when no noise is present, to around 0.15 at 80% noise.



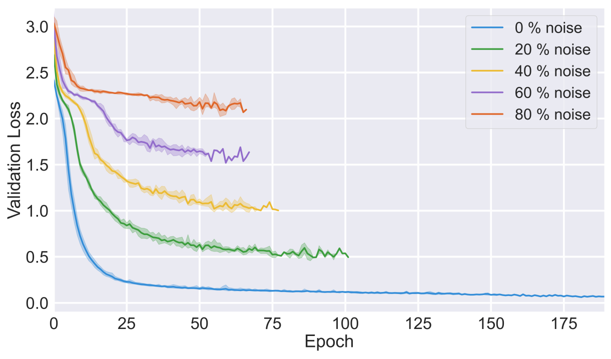
(a) Balanced validation accuracy of GRAPE-GNN on Table-GRAPE at increasing noise levels. Mean of 5-fold CV with shaded area representing min-max range.



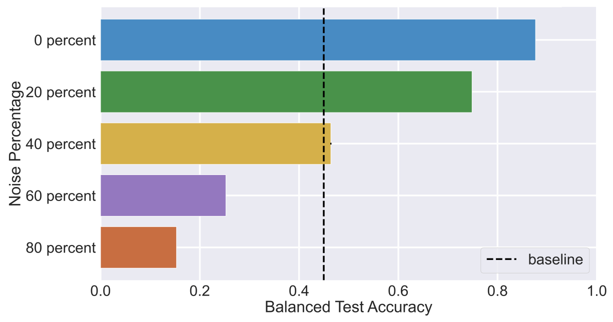
(b) Balanced validation accuracy of GRAPE-GNN on Anatomy-GRAPE at increasing noise levels. Mean of 5-fold CV with shaded area representing min-max range.



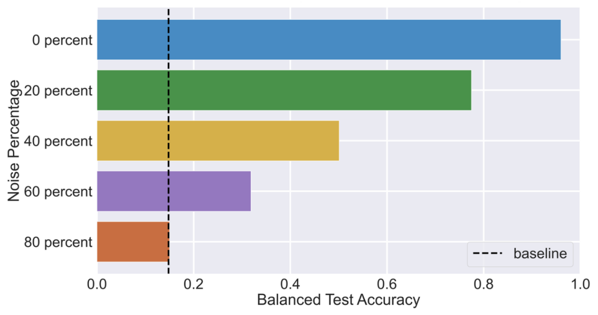
(c) Validation loss of GRAPE-GNN on Table-GRAPE at increasing noise levels. Mean of 5-fold CV with shaded area representing min-max range.



(d) Validation loss of GRAPE-GNN on Anatomy-GRAPE at increasing noise levels. Mean of 5-fold CV with shaded area representing min-max range.



(e) Balanced test accuracy of GRAPE-GNN on Table-GRAPE at increasing noise levels. Mean of 5-fold CV.



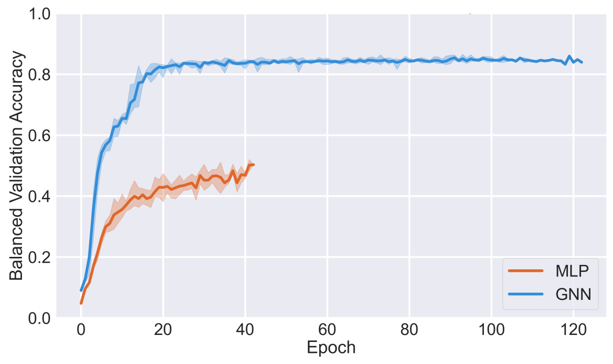
(f) Balanced test accuracy of GRAPE-GNN on Anatomy-GRAPE at increasing noise levels. Mean of 5-fold CV.

Figure 26: Effect of node embedding noise on Table-GRAPE and Anatomy-GRAPE. Top row: Balanced validation accuracy. Middle row: Validation loss. Bottom row: Balanced test accuracy.

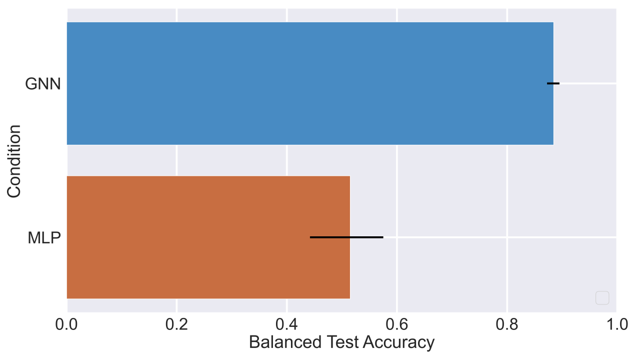
5.4 GNN vs. MLP

This section presents the results obtained in experiment 4, which focused on accessing the effectiveness of graph-based structure prediction compared to direct prediction on the Point-MAE embeddings. The results from Experiment 2 were compared against the MLP that directly predicts on Point-MAE embeddings. First, we present the results GRAPE-GNN and the MLP obtained on Table-GRAPE, after which we continue with the results for Anatomy-GRAPE.

Table-GRAPE Figure 27 presents the balanced validation and test accuracies for GRAPE-GNN and the MLP when trained on Table-GRAPE. Figure 27a displays the validation accuracy curves for both models. GRAPE-GNN shows a steep initial increase, while the MLP curve flattens around epoch 15. The MLP also shows a wider spread across the five folds. Early stopping occurs at approximately epoch 42 for MLP, while GRAPE-GNN continues until epoch 121. Figure 27b presents the test performance of both models. It shows a notable difference in accuracy, with the black bars indicating the range between the minimum and maximum values. This range is also notably larger for the MLP.



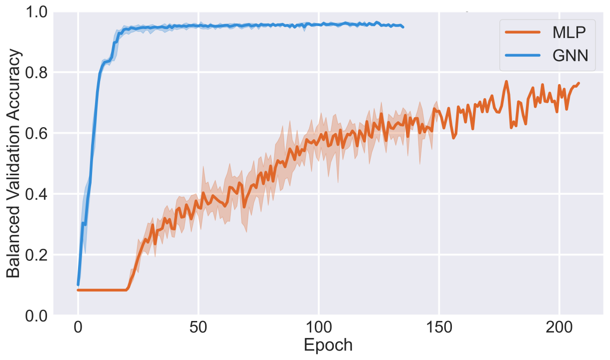
(a) Balanced validation accuracy comparison of GRAPE-GNN using the full Table-GRAPE vs. MLP predicting directly on Point-MAE-generated embeddings. Mean of 5-fold CV with shaded area representing min-max range.



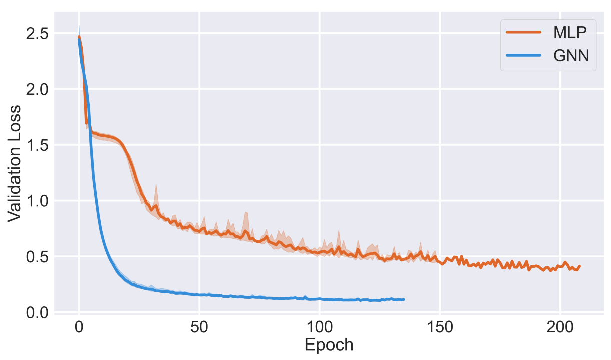
(b) Balanced test accuracy comparison of GRAPE-GNN using the full Table-GRAPE vs. MLP predicting directly on Point-MAE-generated embeddings. Mean of 5-fold CV with black bars representing min-max range.

Figure 27: GNN vs. MLP performance on Table-GRAPE.

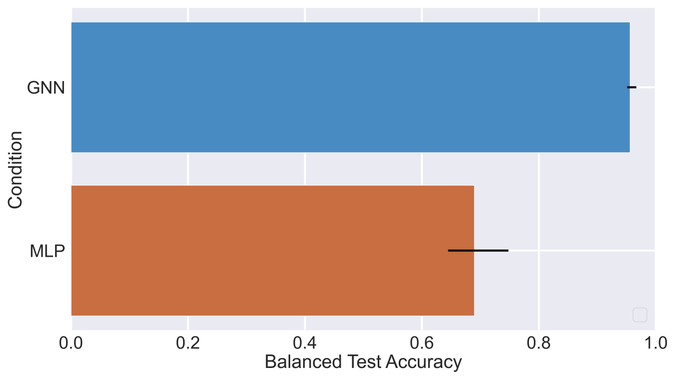
Anatomy-GRAPE Figure 28 presents the balanced validation and test accuracies GRAPE-GNN and the MLP achieved when trained on Anatomy-GRAPE, as well as the validation loss. From Figure 28a it can be seen that MLP follows a gradual but steadily increasing curve, far more stochastic and less steep compared to the GNN. GRAPE-GNN achieves a stable performance far sooner, after approximately 20 epochs. Looking at the MLP's accuracy curve, it seems like there is still an upward trend at the end of training. However, training of the MLP was halted by early stopping as the validation loss did not improve for 8 epochs, which can be seen from Figure 28b. Figure 28c shows that the balanced test accuracy of the MLP is also substantially lower than GRAPE-GNN, with again, more spread.



(a) Balanced validation accuracy comparison of GRAPE-GNN using the full Anatomy-GRAPE vs. MLP predicting directly on Point-MAE-generated embeddings. Mean of 5-fold CV with shaded area representing min-max range.



(b) Loss comparison of GRAPE-GNN using the full Anatomy-GRAPE vs. MLP predicting directly on Point-MAE-generated embeddings. Mean of 5-fold CV with shaded area representing min-max range.



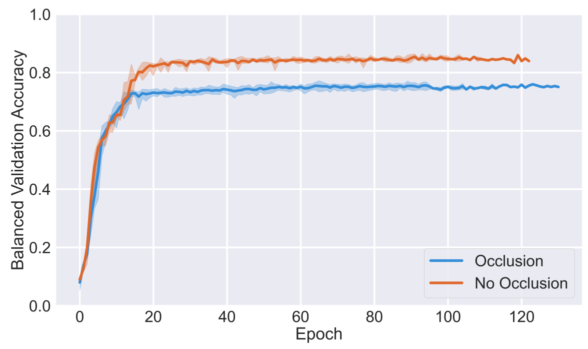
(c) Balanced test accuracy comparison of GRAPE-GNN using the full Anatomy-GRAPE vs. MLP predicting directly on Point-MAE-generated embeddings. Mean of 5-fold CV with black bars representing min-max range.

Figure 28: GNN vs. MLP Performance on Anatomy-GRAPE.

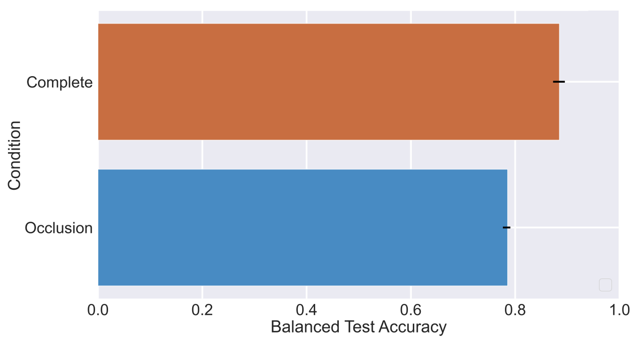
5.5 Point Cloud Occlusion

This section presents the results of the final experiment, which focused on assessing the robustness of GRAPE-GNN to point cloud occlusion. The results from experiment 2 were compared against the results for partially occluded point clouds. First, we present the results GRAPE-GNN achieved on Table-GRAPE, after which we continue with the results for Anatomy-GRAPE.

Table-GRAPE Figure 29 presents the balanced validation and test accuracies GRAPE-GNN achieved on the complete and occluded version of Table-GRAPE. Figure 29a shows two lines that follow a similar trend, the orange depicts GRAPE-GNN trained on no occlusion, while the blue depicts the occlusion case. Both cases show a similar increase at the start and a similar spread throughout. After epoch 20, a gap of ±10 percent accuracy persists. Figure 29b shows a similar gap with the occlusion case scoring 0.78 balanced validation accuracy and the no occlusion scoring 0.88.



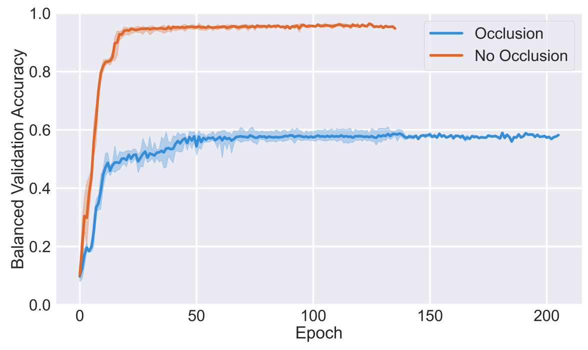
(a) Balanced validation accuracy comparison of GRAPE-GNN using the full (no occlusion) Table-GRAPE vs. Table-GRAPE with 42% occluded samples. Mean of 5-fold CV with shaded area representing min-max range.



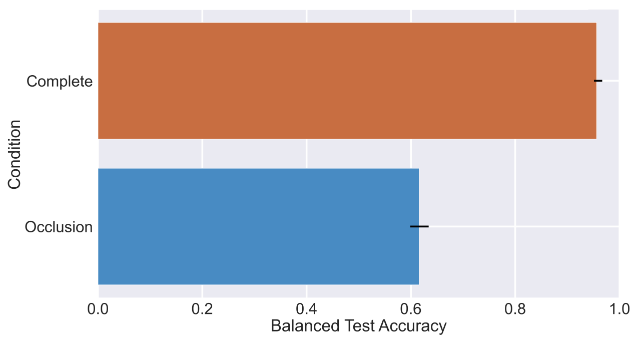
(b) Balanced test accuracy comparison of GRAPE-GNN using the full (no occlusion) Table-GRAPE vs. Table-GRAPE with 42% occluded samples. Mean of 5-fold CV with black bars representing min-max range.

Figure 29: Performance of GRAPE-GNN on complete (no occlusion) Table-GRAPE vs. 42% occluded Table-GRAPE.

Anatomy-GRAPE Figure 30 shows the balanced validation and test accuracies GRAPE-GNN achieved on the complete and occluded version of Anatomy-GRAPE. From Figure 30a the same blue and orange lines can be seen representing the occlusion and no occlusion (complete) case. A distinct gap in performance can be observed. This gap far exceeds the one seen for Table-GRAPE, with the difference being ±36%. Additionally, the occlusion case also exhibits a larger spread. The notable performance gap can also be observed in Figure 30b where the test accuracy drops from 0.95 to 0.61.



(a) Balanced validation accuracy comparison of GRAPE-GNN using the full (no occlusion) Anatomy-GRAPE vs. Anatomy-GRAPE with 42% occluded samples. Mean of 5-fold CV with shaded area representing min-max range.



(b) Balanced test accuracy comparison of GRAPE-GNN using the full (no occlusion) Anatomy-GRAPE vs. Anatomy-GRAPE with 42% occluded samples. Mean of 5-fold CV with black bars representing min-max range.

Figure 30: Performance of GRAPE-GNN on complete (no occlusion) Anatomy-GRAPE vs. 42% occluded Anatomy-GRAPE.

6 Discussion

This chapter provides a detailed analysis of the results presented in Chapter 5. The discussion follows the structure of the previous chapters, addressing masked autoencoders for point cloud self-supervised learning and progressing to graph-based anatomical structure identification. The conclusion of this thesis follows, bringing these previously discussed elements together, and answering the three research questions. This chapter rounds off with a discussion on potential future work.

6.1 Masked Autoencoders for Point Cloud Self-supervised Learning

Point-MAE was used to learn high-quality latent representation of point cloud point patches. The model was trained, validated, and tested on the TableNet and CAESAR datasets. The results presented in Section 5.1 and 5.3 reveal three key findings, which will be discussed in detail below.

Key Findings The first key finding is that Point-MAE extends effectively to both the PartNet and CAESAR datasets, as demonstrated by the loss curves in Figures 18a and 20a, as well as the clean reconstructions visible in Figures 19b and 21b. Figures 18a and 20a display notable spikes, which are highlighted by orange markers. These spikes are an effect of the Cosine Annealing Learning Rate with warm restarts. The learning rate decreases and periodically spikes back up to its original level. The increased learning rate is often immediately followed by an increased loss, which accounts for the spikes seen in the loss curves. This technique helps the model break free from local minima. A good example of this can be seen in Figure 20a, where the loss seems to plateau at the first and second markers. However, due to the increased learning rate the model is able to achieve a lower loss after the third marker.

The second key finding is the importance of inter-patch point variation. The model is quickly able to learn the global shape of the objects, but to create high-quality reconstructions the model needs to create variable-shaped patches. Figure 19c highlights this. The figure shows two top-down views of the table, the left shows the table at a stage that is characterized by low patch variation, while the right shows it at a high variation stage. The individual patches, consisting of 32 points each, have been colored for easier visual discrimination. The left view shows clustered patches, which upon inspection of the enlarged area all bear close resemblance to each other with respect to their point distribution. Points in the right view clearly are more evenly distributed, and the enlarged area reveals patches of different shapes. This feature is also captured in the M-CV plots in Figures 18b and 20b. The M-CV curve in Figures 18b shows a slower and chaotic ascend compared to Figure 20b. This can be explained by the great variability between objects in the TableNet dataset (see Figure 6), compared to the more consistent human physiques in the CAESAR dataset (see Figure 11. Point-MAE is able to find a high M-CV state sooner for the CAESAR dataset, as can be seen by the three quick and steep ascends visible in Figure 20b.

The third and final key finding is that the patch embeddings generated by the trained Point-MAE appear to capture and represent geometric features and spatial relationships of the point cloud to a meaningful extent. The results in Figures 26e and 26f suggest that these embeddings hold useful information. Both for Table-GRAPE and Anatomy-GRAPE, a clear and consistent drop in performance was observed as noise was added to the embeddings. This suggests that the model effectively utilizes the information within the embeddings, rather than relying solely on the graph structure. Furthermore, Figure 22 reveals the t-SNE visualization of the Point-MAE embeddings for the annotated CAESAR data. Each point represents a single patch embedding and is colored according to its class label. Distinct clusters are visible, with certain anatomical structures, such as the patella and quadriceps, forming separate clusters for the left and right sides. The formation of clear clusters suggest that the embeddings capture meaningful geometrical and spatial distinctions. However, some clusters overlap, which can be partially explained by the method used to assign labels to patches. As majority voting (across the individual points) to determine the label of a point patch, some patches may actually belong to two classes. This can happen when anatomical structures are adjacent, such as the deltoid or bicep. Another challenge relating to the patch embeddings is determining whether the model relies more on geometry, positional encodings, or both. This makes it difficult to identify which features are most critical for performance.

Implications Designing effective node embeddings for a graph can be challenging in a complex domain. While point-clouds have been modeled as graphs before, these methods often focus on treating each individual point as a node [63, 79]. However, individual points lack geometry information, therefore, these approaches encode this geometry information in the graph structure. In our approach the graph structure is reserved for the representation of connections and hierarchies between anatomical structures, rendering such methods less applicable. The idea of using a point cloud SSL model like Point-MAE to generate patch embeddings as node embeddings is novel. We were unable to find existing literature that employs a similar method. The findings from Experiments 1 and 3 suggest that this approach is well-suited for capturing geometrically and spatially informative features of point clouds.

Limitations It is also essential to address the limitations of the Point-MAE embedding method. The first limitation revolves around the balance between patch size and geometry information. Larger patches can encompass more geometric detail, but this comes at the cost of reduced granularity. Smaller structures risk being overshadowed by larger ones. For example, if the patch size significantly exceeds the size of a smaller structure, the larger neighboring structures will dominate the patch. As a result, the majority voting across the points will favor these large structures when assigning patch labels, rendering the small structure invisible and preventing it from ever being represented in the graph.
Another challenge in the anatomical structure identification method is accounting for the depth of structures, as many lie beneath others, deeper within the body. As a result, these structures will share the same embedding. To address this, one could assign multiple labels to each location, incorporate some kind of depth feature to create distinct embeddings or design multiple models tailored to progressively deeper structures.
A third limitation arises when the structure is larger than the patch size, resulting in multiple patch embeddings representing a single structure. It might be more effective to generate a single, unified embedding per structure. However, this approach might require significant technical effort, as depending on the size of the structure, varying numbers of patches would need to be transformed into a consistent dimensionality to meet the GNN's requirement for uniform node embeddings.
A final limitation relates to poses, and the variability in positions humans can assume. Point-MAE, and GRAPE-GNN subsequently, should retain performance as a person takes on different poses or orientations. These pose variations were not addressed in this thesis. This was not addressed due to the difficulty of obtaining high-quality point cloud datasets of humans in different poses, in combination with time constraints.
These were the main findings, implications and limitation of the self-supervised section of our method. The next paragraph continues with a discussion on the graph-based structure identification section of our method.

6.2 Graph-Based Anatomical Structure Identification

The motivation for incorporating graph-based prediction into anatomical structure identification is to provide enhanced structural understanding through the connections between nodes, and to enhance robustness and generalization by leveraging the consistent graph-like representation of the human anatomy. This idea resulted in our novel method for anatomical structure identification. We were unable to find existing literature that employs a similar method. The results in Sections 5.2, 5.4 and 5.5 reveal several key findings about our graph based structure identification method, which will be discussed in detail below.

Key Findings The first key finding is that GRAPE-GNN shows promising structure prediction capabilities, which is revealed by the results of experiment 2, detailed in Section 5.2. Figures 23a and 25a reveal consistent performance across the 5 folds, as depicted by the minimal spread. Table-GRAPE achieves good overall balanced accuracy, while Anatomy-GRAPE shows remarkably high performance. This difference can be partially explained by the relative simplicity of Anatomy-GRAPE, involving fewer classes and a more balanced distribution of samples across classes. Additionally, the sample uniformity of Anatomy-GRAPE contrasts with the diversity found in Table-GRAPE, contributing to the performance difference. Tables 5 and 7 show certain classes that achieve perfect F1-scores of 1.0. These cases likely reflect the GNN's ability to exploit specific properties of the graph structure. For example, the *Gluteus* class as shown in Figure 14 is the only one connected to the Hip parent node. This topological feature likely is exploited by the GNN to achieve high accuracy. In the Table-GRAPE dataset, the high performance for *Central Support* and *Runner* classes, also suggests exploitation of graph information. Figure 13 shows that Runner and Central Support always co-occur with the Leg class (a majority class) as child nodes of Drawer Bottom or Star Leg Base. Via message passing, the model potentially learns to leverage these specific hierarchical substructures and co-occurrences to reinforce the identity of these classes.

The second key finding is that GRAPE-GNN is able to achieve good performance with very limited training data. From Figure 24 it can be seen that using only 10% of the original training dataset (±300 samples), GRAPE-GNN performance is still nearly identical to that achieved on the full training dataset. This can be explained by two factors. Firstly, the initial node embeddings are generated by Point-MAE, which has already been trained on the full TableNet training dataset in the SSL phase. Therefore, the initial node embeddings are already of high-quality, reducing the need to learn these representations from scratch. Secondly, the default graph structure produces a high degree of consistency across samples. Each sample is derived from the same default graph, resulting in an inherent topological graph uniformity in the GRAPE datasets, which the model can exploit.

The third key finding suggests that the graph structure provides significant advantages, as suggested by the results of Experiment 4. Figures 27 and 28 reveal a notable difference in performance between GRAPE-GNN and the MLP that directly predicts on the Point-MAE embeddings. It is important to mention again, that due to the inherent differences between MLPs and GNNs, a truly fair comparison with respect to potential model capability is unfeasible. Nonetheless, the results provide us with insights into the benefit of graph-based structure identification. The first distinction is the stable and rapid increase in GRAPE-GNN's accuracy compared to the more stochastic and gradual improvement of the MLP. Although Figure 28a at first glance might suggest that the MLP was approaching similar final performance, training was halted by early stopping, as the validation loss failed to decrease for 8 epochs. The loss curve in Figure 28b also indicates a plateau near the end, suggesting diminishing returns from continued training.

The fourth key finding is that GRAPE-GNN's robustness to point cloud occlusion differs significantly

between Table-GRAPE and Anatomy-GRAPE. Figures 29 and 30 show the results of experiment 5. From Figure 29 it can be seen that GRAPE-GNN's performance drops by roughly 10 percent when trained on occluded point clouds. A notable, but modest difference. However, in the case of Anatomy-GRAPE, the difference in accuracy was roughly 37%. This is a significant drop, indicating that GRAPE-GNN trained on Anatomy-GRAPE is less robust to point cloud occlusion compared to the same model trained on Table-GRAPE. An explanation for this difference could be the fact that both CAESAR and Anatomy-GRAPE were far smaller than their table counterparts TableNet and Table-GRAPE. This might influence both Point-MAE and GRAPE-GNN. It could also be due to the potentially destructive effect occlusion has on the graph. If certain structures are occluded, it might result in disruptions to the graph topology. It could be that simply due to the design of the default graphs Anatomy-GRAPE inherently suffers more from this than Table-GRAPE. A potential remedy to the reduced performance is to include a masking strategy in the SSL training phase that resembles this kind of masking. Instead of only masking random patches, sometimes large connected blocks of patches could be occluded. Additionally, a more stable default graph structure might be explored to guarantee a more stable topology.
The final key finding is the relation between structure size and performance. During the annotation of CAESAR, we intentionally added a relatively small structure, the patella in this case. This was done as a test case to determine if the model would have particular difficulty identifying smaller structures. Figure 25c reveals a positive relationship between structure size and F1 score. However, it's noteworthy that while the patella is the smallest structure, it scores remarkably high, falling outside the confidence interval in the scatter plot. There are two important considerations one needs to account for when interpreting this plot. The first is that each patch is represented as a separate node in the graph. As each patch is of the same size, each node represents roughly the same portion of the point cloud. This means that from the graph perspective, each node (corresponding to part of a structure) represents the same size, with nodes corresponding to bigger structures simply occurring more often. Secondly, as each sample in Anatomy-GRAPE contains all eleven anatomical structures, the size of a structure directly correlates with its representation in the dataset. This means larger structures inherently provide more examples for the model to learn from, which logically explains the observed positive relationship between structure size and F1 score. Despite this, the patella's high performance suggests that the model can effectively learn to identify even smaller structures with limited examples. These were the five key findings relating to graph-based structure identification. We now continue with the implications of these findings.

Implications GRAPE represents point clouds as graphs, where nodes are latent representations of (anatomical) structures and the edges represent hierarchical and relational connections between these structures. The results show that within the bounds of our experiments, GRAPE in combination with our custom GRAPE-GNN, shows promising results for structure identification. We found that the graph structure creates enhanced structural understanding, resulting in stable and consistent predictive performance. Additionally, the use of a default graph allows for efficient learning and exploitation of topological features by the model. The ability to achieve high performance with limited annotated data, as well as its ability to identify both large and small structures suggests a useful method. Additionally, the method was shown to be relatively robust to occlusion in the case of Table-GRAPE, and clear simple strategies are available to potentially improve this.

Limitations While the method shows promising results, there are still certain limitations and questions that need to be addressed. The first is something not yet touched upon in detail, which is the architecture of GRAPE-GNN. The design of GRAPE-GNN was determined through an iterative

process, drawing inspiration from PyTorch Geometric examples and informed by exploratory experiments with various hyperparameters. Although these exploratory tests provided valuable insights, they were not conducted with the rigor necessary for inclusion in this study. The current model may therefore be far from the optimal one, and more research into the architecture is required.
A second obvious limitation is the performance on GRAPE-GNN on the occluded version of Anatomy-GRAPE. It is highly likely that if this model was used in a real application, the input samples would most likely often be partially occluded. It is therefore essential that the model can deal with this. Given the relative robustness GRAPE-GNN showed to occlusion on Table-GRAPE, it was unexpected to see the large drop in performance on Anatomy-GRAPE. Possible explanations, and solutions for this have already been discussed in the previous paragraph. Further exploration is needed to identify methods for increased robustness.
A third limitation is that the current method assumes near perfect point clouds. This means that this method will require a separate data cleaning pipeline to go from raw point cloud scans, that include noise, to clean and pre-processed data the models can accept. No experiments have been done to investigate the effect of such noise on the model.
A fourth limitation is the lack of well-configured and standardized baselines for this problem. Ideally, we would like to compare our model in a setting to other methods, in order to draw scientifically sound conclusions about its relative performance and effectiveness. Without such standardized comparisons, it's challenging to definitively state how GRAPE-GNN stacks up against other potential approaches for anatomical structure identification.
A final limitation is the requirement of high-quality anatomical annotations. The usefulness of the model is entirely dependent on the quality of these annotations. Even if the model achieves high performance, its practical value is limited if the annotations are not precise enough. Obtaining these requires medical specialists, as well as specialized software, making the barrier to entry high.

6.3 Conclusion

This study focused on developing a method for identification of anatomical structures in three-dimensional point cloud representations of the human body. The primary goal was to enable the accurate and robust prediction of underlying anatomical structures based on exterior surface data alone. To achieve this, our study explored Point-MAE for point cloud SSL and GRAPE-GNN for graph-based anatomical structure identification. In this section we answer the three research questions posed in Chapter 1, summarize our main contributions, and draw a final conclusion regarding our proposed method.

Research Questions This study aimed to address three primary research questions. In this paragraph, we will restate these questions and answer them one by one, based on the experimental results presented in Chapter 5 and our interpretation of these results detailed in Chapter 6:

Q1. Can machine learning be used for accurate and robust anatomical structure identification in humans based on exterior surface data alone?

The results presented in Section 5.2 demonstrate that within the bounds of our experiments, machine learning can effectively identify anatomical structures from exterior surface data alone. GRAPE-GNN achieved good performance on both the Table-GRAPE, but particularly high performance on the Anatomy-GRAPE datasets. Overall high precision and recall was observed. However, robustness to occlusion varied. GRAPE-GNN showed decent robustness on the Table-GRAPE data, but a significant drop in performance on the Anatomy-GRAPE data. Further investigation is needed to investigate and improve GRAPE-GNN's robustness to occlusion, as well as investigate its robustness to other factors such as point cloud noise, rotation, or pose variation. Additionally, while good performance was observed in our experiment, it is essential to acknowledge the complexity of the anatomical structure identification problem. Much work remains before the method can be applied in real-world applications.

Q2. Can self-supervised learning models accurately capture and represent fine-grained geometric features and spatial relationships of the human body based on surface level point cloud data?

Point-MAE has demonstrated the ability to capture meaningful geometric and spatial information within patch embeddings. Experiment 3 revealed that these embeddings, serving as node representations, have a large impact on the model's performance. Furthermore, the t-SNE visualisation revealed distinct clusters for different anatomical structures. Results also indicated good performance on the smaller, more fine-grained patella. However, to conclusively evaluate the models effectiveness in capturing smaller structures, more extensive investigation is required. A critical trade-off exists between patch size, granularity, and geometric information. Smaller patches are necessary to capture fine details but generally contain less geometric information. This is an inherent limitation of the patch based approach. Future research should be done to explore alternative point cloud embedding methods, as well as develop more rigorous metrics for quantifying geometric information within these embeddings.

Q3. Can integrating graph-based representations enhance the accuracy of the anatomical structure identification?

The results from our experiments, particularly those detailed in Sections 5.2 and 5.4 , provide strong evidence that integrating graph-based representations can enhance the accuracy of anatomical structure identification. GRAPE-GNN consistently outperformed the MLP using the same Point-MAE embeddings, demonstrating both higher accuracy and faster convergence. The graph structure allowed for the exploitation of topological features, leading to perfect F1-scores for some classes. Our graph based approach also resulted in a large degree of consistency across the dataset, as all samples stem from the same default graph. This potentially reduced the need for a large graph dataset, as revealed by experiment 2a. However, it needs to be stressed that while superior performance of the graph-based approach was observed in our experiment, more rigorous validation is needed to make conclusive statements. Furthermore, only one graph design was explored in this study. A comprehensive evaluation of various graph architectures and topologies is necessary to fully understand the potential of graph-based representations in anatomical structure identification.

Contributions The study provides three main contributions. The first is the GRAPE framework, which combines SSL for point cloud data with a hierarchical graph based representation of the anatomical structures. The second contribution is GRAPE-GNN, our custom graph neural network for anatomical structure identification. This model, in combination with the GRAPE datasets enabled us to leverage both spatial, as well as hierarchical and relation information, for anatomical structure identification. The second contribution is the anatomically labelled subset of the CAESAR dataset, in addition to the graph datasets Table-GRAPE and Anatomy-GRAPE.

In summary, this study has demonstrated the potential of combining SSL via Point-MAE and graph-based representations for anatomical structure identification in point cloud data. While our results are promising, they also highlight several areas for improvement and further research, which we will discuss in the final section on future work.

6.4 Future Work

This final section will briefly explore potential future works. During implementation and validation of our method, we have encountered various areas of improvement. However, due to time limitation, we could not address these in this study.

Edge Features Certain GNN architectures support edge features, an extension that our method did not explore. Incorporating edge features could potentially improve the model's ability to distinguish anatomical structures, as well as capture more nuanced relationships between them. Such features could represent information about the connection between the structures, or relative spatial orientations. Future works could explore methods of generating such features, which could entail designing them by hand using anatomical knowledge, or deriving them from point clouds using a learning mechanism.

Embedding Models This study exclusively used Point-MAE to generate patch embeddings, which proved effective for the purposes of this study. However, alternative embedding models could potentially improve the embedding quality, or be more suited to the physiological nature of human point clouds. Future work should investigate other SSL approaches such as PointContrast [80] or Point-M2AE [5]. These methods might provide alternative perspectives on feature extraction and representation, and might better to represent anatomical structures in embedding space. Alternatively, supervised models such as DGCNN [81] or PointNet++ [82] could be adapted for point cloud embedding, leveraging transfer learning from models pre-trained on large scale point cloud datasets. Furthermore, exploring multi-modal embedding techniques that incorporate RGB or surface normal information could increase the robustness of the anatomical structure identification system.

Graph Designs This study used one design principle behind the graphs. The idea was to represent the anatomical parts as part of a larger hierarchical graph that aims to model how the smaller anatomical structures constitutes to the larger human body. The hierarchical design was inspired by the graphs from the PartNet dataset, and this representation also seemed logical for the human case. It could prove worthwhile to explore other designs, that might capture other anatomical relationships and spatial dependencies more informative to the GNN. Our graph also only incorporated undirected edges, and only one edge type. This was done to simplify the initial implementation and reduce computational complexity given the time constraints. However, to effectively capture the complex and nuanced interrelations of the human anatomy and the point cloud data, exploration of more sophisticated graph structures is necessary. Future work could investigate the use of directed edges, multiple edge types, or even hypergraph representations to capture higher order relationships.

Robustness The discussion already touched on the lack of robustness to point cloud occlusion GRAPE-GNN showed when applied to Anatomy-GRAPE. This limitation provides a clear direction for future work. While Point-MAE currently uses random masking during training, implementing a strategy that simulates large-scale point cloud occlusion could enhance GRAPE-GNN's robustness to occlusion.Furthermore, the method should also be subjected to other challenging conditions such as point cloud noise, rotation and pose variation. Developing methods to address these challenges would greatly improve our methods usefulness to the real world, where input data is almost never perfect.

GNN Architectures As previously stated, GRAPE-GNN was designed through an iterative process, drawing inspiration from various sources and informed by exploratory experiments with various hyperparameters. Although these exploratory tests provided valuable insights during development, they were not conducted with the rigor necessary for inclusion in our study. An exploration of various GNN architectures, with a comprehensive approach to hyperparameter optimization could provide deeper insights and potentially lead to more effective models. Particularly in the context of more sophisticated graph designs that incorporate edge embeddings, multiple edge types, and hypergraphs, it is essential to evaluate GNN operators that support these advanced features.

Multi-layer Anatomical Models The human anatomy is an intricate composition of multiple layers, consisting of muscles, bones, tendons, organs and much more. This study did not delve into the identification of two or more structure that are located beneath or within each other. Future research that investigates this layering could improve the methods applicability. A possible direction is to model each layer or tissue type as in a separate graph, and perform prediction on each graph independently to identify all structures at a specific location. Alternatively, a graph node could include multiple labels to represent overlapping structures. However, this approach might lead to more ambiguous nodes, as they would represent multiple structures simultaneously. Exploring these models in future research could not only improve applicability, and representational capacity of the anatomical structure identification method.

Bibliography

[1] K. Börner, S. A. Teichmann, E. M. Quardokus, J. C. Gee, K. Browne, D. Osumi-Sutherland, B. W. Herr, A. Bueckle, H. Paul, M. Haniffa, L. Jardine, A. Bernard, S.-L. Ding, J. A. Miller, S. Lin, M. K. Halushka, A. Boppana, T. A. Longacre, J. Hickey, Y. Lin, M. T. Valerius, Y. He, G. Pryhuber, X. Sun, M. Jorgensen, A. J. Radtke, C. Wasserfall, F. Ginty, J. Ho, J. Sunshine, R. T. Beuschel, M. Brusko, S. Lee, R. Malhotra, S. Jain, and G. Weber, "Anatomical structures, cell types and biomarkers of the human reference atlas," *Nature Cell Biology*, vol. 23, no. 11, pp. 1117–1128, 2021.

[2] A. Džakula and D. Relić, "Health workforce shortage - doing the right things or doing things right?," *Croat Med J*, vol. 63, pp. 107–109, Apr 2022.

[3] M. Observatory, "Migration and the health and care workforce," 2023. Accessed: 2024-05-30.

[4] S. N. (CBS), "Helft zorgwerknemers vindt werkdruk te hoog," 2022. Accessed: 2024-05-30.

[5] R. Zhang, Z. Guo, P. Gao, R. Fang, B. Zhao, D. Wang, Y. Qiao, and H. Li, "Point-m2ae: Multi-scale masked autoencoders for hierarchical point cloud pre-training," *arXiv preprint arXiv:2205.14401*, 2022.

[6] Y. Pang, W. Wang, F. E. Tay, W. Liu, Y. Tian, and L. Yuan, "Masked autoencoders for point cloud self-supervised learning," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pp. 604–621, Springer, 2022.

[7] X. Chen, X. Wang, K. Zhang, K.-M. Fung, T. C. Thai, K. Moore, R. S. Mannel, H. Liu, B. Zheng, and Y. Qiu, "Recent advances and clinical applications of deep learning in medical image analysis," *Medical Image Analysis*, vol. 79, p. 102444, 2022.

[8] L. Alzubaidi, M. Al-Amidie, A. Al-Asadi, A. Humaidi, O. Al-Shamma, M. Fadhel, J. Zhang, J. Santamaría, and Y. Duan, "Novel transfer learning approach for medical imaging with limited labeled data," *Cancers (Basel)*, vol. 13, p. 1590, Mar 2021.

[9] P. Goyal, Q. Duval, I. Seessel, M. Caron, I. Misra, L. Sagun, A. Joulin, and P. Bojanowski, "Vision models are more robust and fair when pretrained on uncurated images without supervision," 2022.

[10] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," 2019.

[11] SNOMED International, "SNOMED CT," 2024. Accessed: 2024-06-04.

[12] "Complete anatomy." https://www.elsevier.com/products/complete-anatomy. Accessed: 2024-08-05.

[13] X. Tang, "The role of artificial intelligence in medical imaging research," *BJR Open*, vol. 2, p. 20190031, Nov 2019.

[14] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, p. 60–88, Dec. 2017.

[15] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.

[16] S. Goyal and T. Kataria, “Image guidance in radiation therapy: techniques and applications,” *Radiol Res Pract*, vol. 2014, p. 705604, 2014. Epub 2014 Dec 17. PMID: 25587445; PMCID: PMC4281403.

[17] E. J. da Silva, A. Leal, J. Milano, L. J. da Silva, R. Clemente, and R. Ramina, “Image-guided surgical planning using anatomical landmarks in the retrosigmoid approach,” *Acta Neurochir (Wien)*, vol. 152, pp. 905–910, May 2010. Epub 2009 Nov 10. PMID: 19902141.

[18] M. Valstar, B. Martinez, X. Binefa, and M. Pantic, “Facial point detection using boosted regression and graph models,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010*, pp. 2729–2736, IEEE, June 2010. 10.1109/CVPR.2010.5539996 ; 23rd IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010 ; Conference date: 13-06-2010 Through 18-06-2010.

[19] X. Cao, Y. Wei, F. Wen, and J. Sun, “Face alignment by explicit shape regression,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2887–2894, 2012.

[20] T. Cootes, G. Edwards, and C. Taylor, “Active appearance models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[21] Y. Sun, X. Wang, and X. Tang, “Deep convolutional network cascade for facial point detection,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3476–3483, 2013.

[22] Z. Zhang, P. Luo, C. Loy, and X. Tang, “Facial landmark detection by deep multi-task learning,” in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8694 of *Lecture Notes in Computer Science*, pp. 94–108, Springer, Cham, 2014.

[23] F. C. Ghesu, B. Georgescu, T. Mansi, D. Neumann, J. Hornegger, and D. Comaniciu, “An artificial agent for anatomical landmark detection in medical images,” in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2016* (S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, eds.), (Cham), pp. 229–237, Springer International Publishing, 2016.

[24] A. Alansary, O. Oktay, Y. Li, L. L. Folgoc, B. Hou, G. Vaillant, K. Kamnitsas, A. Vlontzos, B. Glocker, B. Kainz, and D. Rueckert, “Evaluating reinforcement learning agents for anatomical landmark detection,” *Medical Image Analysis*, vol. 53, pp. 156–164, 2019.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” 2013.

[26] J. M. H. Noothout, B. D. De Vos, J. M. Wolterink, E. M. Postma, P. A. M. Smeets, R. A. P. Takx, T. Leiner, M. A. Viergever, and I. Isgum, “Deep learning-based regression and classification for automatic landmark localization in medical images,” *IEEE Transactions on Medical Imaging*, vol. 39, p. 4011–4022, Dec. 2020.

[27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.

[28] L. Liu, J. M. Wolterink, C. Brune, and R. N. J. Veldhuis, “Anatomy-aided deep learning for medical image segmentation: a review,” *Physics in Medicine & Biology*, vol. 66, no. 11, pp. 10.1088/1361–6560/abfbf4, 2021.

[29] M. S. Nosrati and G. Hamarneh, “Incorporating prior knowledge in medical image segmentation: a survey,” 2016.

[30] A. Tragakis, C. Kaul, R. Murray-Smith, and D. Husmeier, “The fully convolutional transformer for medical image segmentation,” 2023.

[31] J. Ma, Y. He, F. Li, L. Han, C. You, and B. Wang, “Segment anything in medical images,” *Nature Communications*, vol. 15, p. 654, Jan 2024.

[32] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, “A simple framework for contrastive learning of visual representations,” *CoRR*, vol. abs/2002.05709, 2020.

[33] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” *arXiv:2111.06377*, 2021.

[34] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, “Bootstrap your own latent: A new approach to self-supervised learning,” 2020.

[35] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” 2020.

[36] I. Misra and L. van der Maaten, “Self-supervised learning of pretext-invariant representations,” 2019.

[37] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” 2020.

[38] R. Krishnan, P. Rajpurkar, and E. J. Topol, “Self-supervised learning in medicine and healthcare,” *Nature Biomedical Engineering*, vol. 6, no. 12, pp. 1346–1352, 2022.

[39] O. Ciga, T. Xu, and A. L. Martel, “Self supervised contrastive learning for digital histopathology,” *Machine Learning with Applications*, vol. 7, p. 100198, 2022.

[40] A. van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” 2019.

[41] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” 2021.

[42] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, “Simmim: A simple framework for masked image modeling,” 2022.

[43] H. Hotelling, “Relations between two sets of variates,” in *Breakthroughs in Statistics*, pp. 162–190, Springer, 1992.

[44] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[45] R. Balestriero, M. Ibrahim, V. Sobal, A. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, A. Schwarzschild, A. G. Wilson, J. Geiping, Q. Garrido, P. Fernandez, A. Bar, H. Pirsiavash, Y. LeCun, and M. Goldblum, “A cookbook of self-supervised learning,” 2023.

[46] J. Bromley, I. Guyon, Y. Lecun, E. Sackinger, and R. Shah, “Signature verification using a siamese time delay neural network,” in *Advances in neural information processing systems (NIPS 1993)* (J. Cowan and G. Tesauro, eds.), vol. 6, Morgan Kaufmann, 1993.

[47] S. Chopra, R. Hadsell, and Y. Lecun, “Learning a similarity metric discriminatively, with application to face verification,” 07 2005.

[48] A. Bardes, J. Ponce, and Y. LeCun, “Vicreg: Variance-invariance-covariance regularization for self-supervised learning,” 2022.

[49] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, “Barlow twins: Self-supervised learning via redundancy reduction,” 2021.

[50] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” 2018.

[51] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” pp. 1096–1103, 01 2008.

[52] M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” *CoRR*, vol. abs/1603.09246, 2016.

[53] H. Fan, H. Su, and L. Guibas, “A point set generation network for 3d object reconstruction from a single image,” 2016.

[54] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[55] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2017.

[56] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” 2018.

[57] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” 2018.

[58] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, V. M. Tran, A. Chiappino-Pepe, A. H. Badran, I. W. Andrews, E. J. Chory, G. M. Church, E. D. Brown, T. S. Jaakkola, R. Barzilay, and J. J. Collins, “A deep learning approach to antibiotic discovery,” *Cell*, vol. 180, no. 4, pp. 688–702.e13, 2020.

[59] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, “Fake news detection on social media using geometric deep learning,” 2019.

[60] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire, P. W. Battaglia, V. Gupta, A. Li, Z. Xu, A. Sanchez-Gonzalez, Y. Li, and P. Velickovic, “Eta prediction with graph neural networks in google maps,” in *Proceedings of the 30th ACM International Conference on Information &amp; Knowledge Management*, CIKM ’21, ACM, Oct. 2021.

[61] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” 2017.

[62] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, “Masked label prediction: Unified message passing model for semi-supervised classification,” 2021.

[63] W. Shi, Ragunathan, and Rajkumar, “Point-gnn: Graph neural network for 3d object detection in a point cloud,” 2020.

[64] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, “PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[65] K. Robinette, S. Blackwell, H. Daanen, M. Boehmer, S. Fleming, T. Brill, D. Hoeferlin, and D. Burnsides, “Anthropometry resource (caesar) final report, volume i: Summary,” 01 2002.

[66] Y. Yang, Y. Yu, Y. Zhou, S. Du, J. Davis, and R. Yang, “Semantic parametric reshaping of human body models,” in *Proceedings of the 3DV Workshop on Dynamic Shape Measurement and Analysis*, (Zurich, Switzerland), IEEE, 2014.

[67] Blender Online Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Amsterdam, The Netherlands, 2023. Version 3.0. URL `https://www.blender.org`.

[68] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” *ICLR*, 2021.

[69] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, “Point-bert: Pre-training 3d point cloud transformers with masked point modeling,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[70] C. Min, L. Xiao, D. Zhao, Y. Nie, and B. Dai, “Occupancy-mae: Self-supervised pre-training large-scale lidar point clouds with masked occupancy autoencoders,” *IEEE Transactions on Intelligent Vehicles*, 2023.

[71] G. Hess, J. Jaxing, E. Svensson, D. Hagerman, C. Petersson, and L. Svensson, “Masked autoencoder for self-supervised pre-training on lidar point clouds,” in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, IEEE, Jan. 2023.

[72] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, “End-to-end multi-view fusion for 3d object detection in lidar point clouds,” 2019.

[73] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” *arXiv preprint arXiv:1612.00593*, 2016.

[74] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2023.

[75] S. Brody, U. Alon, and E. Yahav, “How attentive are graph attention networks?,” 2022.

[76] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, ACM, Apr. 2024.

[77] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," Mar. 2019.

[78] L. Biewald, "Experiment tracking with weights and biases," 2020. Software available from wandb.com.

[79] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud semantic segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10288–10297, 2019.

[80] S. Xie, J. Gu, D. Guo, C. R. Qi, L. J. Guibas, and O. Litany, "Pointcontrast: Unsupervised pre-training for 3d point cloud understanding," 2020.

[81] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," 2019.

[82] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017.

Appendix

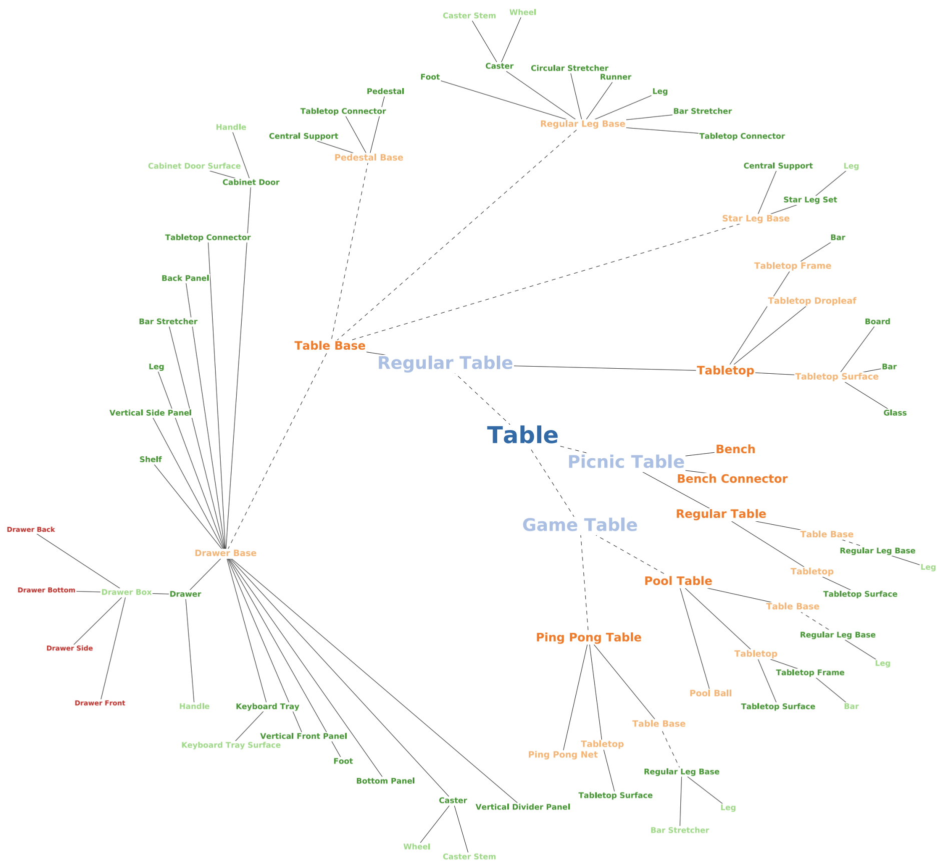


Figure 31: Expert-defined part hierarchy for the table category [64].