



UNIVERSIDAD DE CHILE  
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS  
ESCUELA DE POSTGRADO Y EDUCACIÓN CONTINUA  
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

## **SELF-SUPERVISED LEARNING ON 3D REPRESENTATIONS**

TESIS PARA OPTAR AL GRADO DE  
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE  
INGENIERO CIVIL EN COMPUTACIÓN

**LUCAS OYARZÚN MÉNDEZ**

PROFESOR GUÍA:  
IVÁN SIPIRAN MENDOZA

PROFESOR CO-GUÍA:  
JOSÉ SAAVEDRA RONDO

MIEMBROS DE LA COMISIÓN:  
JUAN BARRIOS NUÑEZ  
FELIPE BRAVO MÁRQUEZ  
RICARDO ÑANCULEF ALEGRÍA

Este trabajo ha sido parcialmente financiado por:  
ANID FONDECYT GRANT 11220211

SANTIAGO DE CHILE

2024

RESUMEN DE LA TESIS PARA OPTAR  
AL TÍTULO DE MAGÍSTER EN CIENCIAS,  
MENCION COMPUTACIÓN  
AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN  
POR: LUCAS OYARZÚN MÉNDEZ  
FECHA: 2024  
PROF. GUÍA: IVÁN SIPIRAN

## APRENDIZAJE AUTOSUPERVISADO SOBRE REPRESENTACIONES 3D

La creciente accesibilidad de los datos de nubes de puntos 3D ha acelerado el desarrollo de métodos de aprendizaje profundo para su análisis y procesamiento. Sin embargo, la eficacia de las redes neuronales en este campo se ve a menudo obstaculizada por la necesidad de amplios conjuntos de datos, cuya creación es costosa y laboriosa. Esta tesis explora el marco de las redes siamesas como estrategias de preentrenamiento para redes neuronales en nubes de puntos 3D. Aunque estos métodos han demostrado resultados en el preentrenamiento de redes neuronales 2D, su estudio en el contexto de datos 3D es relativamente reciente.

Nuestra investigación propone la aplicación de técnicas de entrenamiento basadas en redes siamesas, como BYOL o SIMSIAM, a los codificadores de las principales redes neuronales diseñadas para el procesamiento de nubes de puntos 3D. Estos regímenes de preentrenamiento permiten a los codificadores generar representaciones de datos sin depender de etiquetas, lo que puede mejorar el rendimiento de la red en tareas posteriores como la clasificación, la segmentación y el reconocimiento de objetos en entornos urbanos, mejorando así la fiabilidad de los vehículos autónomos en escenarios complejos.

También se experimentó sobre un enfoque alternativo: sustituir el conjunto de datos de preentrenamiento convencional del estado del arte por SimpleShape, un conjunto de datos creado artificialmente. Este enfoque se inspira en el concepto de aprendizaje supervisado basado en fórmulas. Nuestro estudio pretende investigar la eficacia de este enfoque sobre nubes de puntos 3D. Los resultados indican que incluso una versión simplificada de este conjunto de datos puede producir resultados competitivos frente a las pruebas de referencia actuales, a pesar de no estar diseñado específicamente para el preentrenamiento de nubes de puntos 3D.

La eficacia de estas representaciones adquiridas se evaluó utilizando métricas de evaluación estándar en el campo. La precisión de la clasificación obtenida en el conjunto de datos ScanObjectNN en su variante OBJ-BG fue del 94,15%. En la prueba comparativa de aprendizaje de few-shot en Model-Net40 5w10s, la precisión fue del 97,1%, y en ModelNet40 5w20s, del 98,8%. Con estos resultados, los marcos de trabajo propuestos superan los resultados del estado del arte en estas pruebas comparativas. El código desarrollado e implementado a efectos de esta investigación se encuentra disponible en

[https://github.com/LucasOyarzun/Point\\_Simsiam](https://github.com/LucasOyarzun/Point_Simsiam).



ABSTRACT OF THE THESIS TO APPLY FOR  
THE DEGREE OF MASTER IN SCIENCES,  
MENTION IN COMPUTER SCIENCE  
THE DEGREE OF CIVIL COMPUTER ENGINEER  
BY: LUCAS OYARZÚN MÉNDEZ  
YEAR: 2024  
THESIS ADVISOR: IVÁN SIPIRAN

## SELF-SUPERVISED LEARNING ON 3D REPRESENTATIONS

The increasing accessibility of 3D point cloud data has accelerated the development of deep learning methods for its analysis and processing. However, neural networks' effectiveness in this field is often hindered by the need for extensive labeled datasets, which is both costly and labor-intensive. This thesis explores Siamese network frameworks as pre-training strategies for neural networks on 3D point clouds. While these methods have demonstrated exceptional results in 2D neural network pre-training, their study in the context of 3D point clouds is relatively recent.

Our research proposes the application of training techniques based on Siamese networks, such as BYOL or SIMSIAM, to the encoders of leading neural networks designed for 3D point cloud processing. These pre-training regimes enable the encoders to generate data representations adeptly without relying on labels, potentially bolstering network performance in downstream tasks like classification, segmentation, and object recognition in urban environments, thereby enhancing the reliability of autonomous vehicles in complex scenarios.

An alternative approach was experimented with: substituting the conventional pre-training dataset in current state-of-the-art models with SimpleShape, an artificially created dataset. This approach is inspired by the concept of formula-driven supervised learning. Our study aims to investigate the effectiveness of this approach. The results indicate that even a simplified version of this dataset can produce competitive results against current benchmarks, despite not being specifically tailored for 3D point cloud pre-training.

The effectiveness of these acquired representations was evaluated using well-established evaluation metrics. The classification accuracy achieved on the ScanObjectNN dataset under its OBJ-BG variant was 94.15%. On the few-shot learning benchmark on ModelNet40 5w10s, the accuracy was 97.1%, and on ModelNet40 5w20s, it was 98.8%. With these results, the proposed frameworks surpass the state-of-the-art results in these benchmarks. The code developed and implemented for the purpose of this research is available at [https://github.com/LucasOyarzun/Point\\_Simsiam](https://github.com/LucasOyarzun/Point_Simsiam).

*To Lucas,  
who, at the age of seven, made the right choice.*

# Acknowledgments

To my sister, Linmara, my guiding star, who paved the way for me to reach today. Thank you for walking beside me and teaching me the art of living for happiness.

To my father, Nibaldo, who has always been by my side, believing in me and inspiring me to give my all for my dreams.

To my mother, Lorena, who has been my lifelong compass. From her, I learned that with enough effort, I could achieve anything. Thank you for everything.

To my grandmother, Inés, who instilled in me love and discipline. For being my sanctuary and supporting me from afar.

To Janice, my beloved companion, who learned to love and understand me, caring for me and accompanying me every step of the way.

To Professor Nelson, who believed in me from the very beginning. Thank you for being there throughout my journey and showing me that I still had untapped potential.

To Professor Iván, who inspired, guided and taught me throughout all this process.

To Neptuno, brother who always welcomed me home with genuine joy.

# Table of content

<b>1. Introduction</b>	<b>1</b>
1.1. Point clouds learning . . . . .	1
1.2. Self-supervised learning . . . . .	2
1.3. Synthetic datasets . . . . .	4
1.4. Document structure . . . . .	5
<b>2. Related work</b>	<b>6</b>
2.1. Point cloud learning . . . . .	6
2.1.1. PointNet . . . . .	9
2.1.2. DGCNN . . . . .	10
2.1.3. PointTransformer . . . . .	12
2.1.4. Point-MLP . . . . .	14
2.2. Self-supervised learning . . . . .	15
2.2.1. Autoencoders . . . . .	15
2.2.1.1. Masked autoencoders . . . . .	16
2.2.2. Siamese networks . . . . .	16
2.2.2.1. BYOL . . . . .	17
2.2.2.2. SIMSIAM . . . . .	18
2.3. SSL on point clouds . . . . .	19
2.3.1. Point-MAE . . . . .	20
2.3.2. Contrastive learning . . . . .	21
2.3.3. Formula-driven supervised learning . . . . .	22
2.3.3.1. SimpleShape . . . . .	23
2.4. Research gaps in SSL on point clouds . . . . .	23
<b>3. Methodology</b>	<b>24</b>
3.1. Problem statement . . . . .	24
3.2. Research questions . . . . .	26
3.3. Hypothesis . . . . .	26
3.4. Goals . . . . .	26
3.5. Scope and assumptions . . . . .	27

<b>4. Proposal</b>	<b>29</b>
4.1. Siamese networks proposed models . . . . .	29
4.1.1. Data augmentation . . . . .	29
4.1.2. Encoders . . . . .	33
4.1.3. Prediction head networks . . . . .	34
4.1.4. General proposed model . . . . .	35
4.2. SimpleShape for formula-driven supervised learning . . . . .	36
4.2.1. Dataset creation . . . . .	36
<b>5. Datasets</b>	<b>40</b>
5.1. ShapeNet . . . . .	40
5.2. ShapeNetPart . . . . .	41
5.3. ModelNet . . . . .	42
5.4. ScanObjectNN . . . . .	43
5.5. SimpleShape . . . . .	44
<b>6. Evaluation metrics</b>	<b>45</b>
6.1. Linear probing . . . . .	45
6.2. Classification accuracy . . . . .	46
6.3. Few-shot classification accuracy . . . . .	47
6.4. Segmentation mIoU . . . . .	47
6.5. Dimensionality reduction evaluation . . . . .	48
6.5.1. t-SNE . . . . .	49
6.5.2. UMAP . . . . .	49
6.6. Retrieval techniques . . . . .	50
6.6.1. Nearest neighbor (NN) . . . . .	51
6.6.2. Mean average precision (mAP) . . . . .	51
<b>7. Experiments to perform</b>	<b>52</b>
7.1. Data preparation . . . . .	52
7.2. Models . . . . .	53
7.3. Standard benchmarks . . . . .	54
7.3.1. Pre-training . . . . .	55
7.3.2. Linear probing . . . . .	55
7.3.3. Classification . . . . .	55
7.3.4. Segmentation . . . . .	55
7.3.5. Dimensionality Reduction Evaluation . . . . .	56
7.3.6. Retrieval Techniques . . . . .	56
<b>8. Results and analysis</b>	<b>57</b>
8.1. Siamese networks . . . . .	57
8.1.1. Linear probing . . . . .	58

8.1.2.	Classification on ModelNet40 . . . . .	59
8.1.3.	Classification on ScanObjectNN . . . . .	62
8.1.4.	Few-shot classification . . . . .	65
8.1.5.	Point cloud segmentation . . . . .	67
8.1.6.	Dimensionality reduction evaluation . . . . .	69
8.1.7.	Retrieval techniques . . . . .	75
8.2.	Formula-driven supervised learning . . . . .	78
8.2.1.	Linear probing . . . . .	78
8.2.2.	Downstream tasks . . . . .	78
8.2.2.1.	Classification on ModelNet40 . . . . .	78
8.2.2.2.	Classification on ScanObjectNN . . . . .	80
8.2.2.3.	Few-shot classification . . . . .	80
8.2.2.4.	Point Cloud segmentation . . . . .	81
8.2.2.5.	Downstream tasks results analysis . . . . .	81
8.2.3.	Dimensionality reduction evaluation . . . . .	82
8.2.4.	Retrieval techniques . . . . .	88
<b>9.</b>	<b>Conclusions</b>	<b>91</b>
9.1.	Contributions . . . . .	92
9.2.	Achievement of goals . . . . .	93
9.3.	Future work . . . . .	93
	<b>Bibliography</b>	<b>95</b>

# Table index

4.1.	Families of plane curves used as directrix of the cylinders and cones in SimpleShape datasets. . . . .	38
4.2.	Examples of shapes generated by SimpleShape [29]. . . . .	39
7.1.	Inference time and pre-training time of proposed methods with different encoders for ModelNet40 train set (subsampled to 1024 points) and the number of trainable parameters for each model’s encoder. . . . .	54
8.1.	Linear evaluation on ModelNet40 [11] using <i>SVM</i> and <i>KNN</i> ( $k = 20$ ). Methods are categorized according to their self-supervised learning (SSL) methodology. The best of the proposed models and the best state-of-the-art model are highlighted in bold. Models are organized by methodology: first, denoising autoencoder models; then, contrastive learning-based models; and finally, the proposed models. . . . .	59
8.2.	Comparison of pre-trained models vs from scratch performance: Shape classification on ModelNet40. ‘Acc (%)’ denotes overall accuracy and ‘Voting Acc (%)’ denotes voting accuracy. The results are listed starting with autoencoder-based models, followed by models that primarily use contrastive learning, and finally the results of the proposed models. . . . .	61
8.3.	Shape classification on ModelNet40. ‘Voting Acc (%)’ indicates voting accuracy. Each version of PointSIMSIAM and PointBYOL is distinguished, along with the ‘from scratch’ results of their respective encoders. . . . .	62
8.4.	Accuracy (%) results of fine-tuning on ScanObjectNN classification. Models are categorized according to their pre-training methodology: first, models based on the autoencoder methodology; then, state-of-the-art models following contrastive learning; and finally, the models proposed in this research. . . . .	64
8.5.	Comparison of pre-trained models vs from scratch performance: Accuracy (%) results of fine-tuning ScanObjectNN classification, comparing each Siamese network model with its backbone results when trained from scratch. . . . .	65
8.6.	Few-shot classification results on ModelNet40. Mean accuracy (%) and standard deviation (%) from 10 independent experiments are reported. . . . .	66

8.7.	Comparison of pre-trained models vs from scratch performance: Few-shot classification results on ModelNet40, comparing each Siamese network model with its backbone results when trained from scratch. Mean accuracy (%) and standard deviation (%) from 10 independent experiments are shown. . . . .	67
8.8.	Part Segmentation on ShapeNetPart [61]. ‘ $mIoU_I$ ’ denotes the mean IoU of the model over all instances in the dataset. The results are categorized based on autoencoder models, contrastive models, and proposed models. . . . .	68
8.9.	Comparison of pre-trained models vs from scratch performance: Part segmentation on ShapeNetPart [61], comparing each Siamese network model with its backbone results when trained from scratch. ‘ $mIoU_I$ ’ denotes the model’s mean IoU across all instances in the dataset. The results are categorized based on autoencoder models, contrastive models, and proposed models. . . . .	69
8.10.	<b>Retrieval results.</b> ‘NN (%)’ denotes the percentage of instances where the model accurately identified the nearest neighbor. ‘mAP (%)’ signifies the model’s mean average precision in its retrieval tasks across various levels of recall. . . .	76
8.11.	Linear evaluation on ModelNet40 [11] by SVM and kNN. . . . .	79
8.12.	Shape classification on ModelNet40. ‘Acc (%)’ denotes overall accuracy and ‘Voting Acc (%)’ indicates voting accuracy. . . . .	79
8.13.	Few-shot classification results on ModelNet40. Mean accuracy (%) and standard deviation (%) from 10 independent experiments are reported. . . . .	80
8.14.	Accuracy (%) results from fine-tuning on ScanObjectNN classification. . . . .	80
8.15.	Part segmentation on ShapeNetPart [61], comparing each Siamese network model with its backbone results when trained from scratch. ‘ $mIoU_I$ ’ denotes the model’s mean IoU across all instances in the dataset. The results are categorized based on autoencoder models, contrastive models, and proposed models. . . .	81
8.16.	<b>SimpleShape Retrieval results.</b> ‘NN (%)’ denotes the percentage of instances where the model correctly identified the nearest neighbor. ‘mAP (%)’ denotes the average precision of the model in its retrieval tasks at different recall levels. . . . .	88



# Figure Index

1.1.	<b>Transfer learning process.</b> The pre-training model performs a pre-training task in which it learns $\theta^*$ parameters that it will transfer as initial parameters to the subsequent model to initiate its training and achieve the target prediction task. . . . .	3
2.1.	<b>PointNet architecture.</b> The network takes as input a set of $N$ points from a given point cloud. The network is designed to perform both sample and feature transformations, which are then aggregated using a max-pooling strategy. This process facilitates the network’s ability to produce a classification score for $c$ different classes. Building on the basic classification network, the segmentation network is extended to integrate both global and local features through concatenation. In this architecture, ‘mlp’ denotes multi-layer perceptrons, with the subsequent layer sizes given in parentheses. To ensure stability and improve convergence during training, batch normalization is applied across all layers in conjunction with the ReLU activation function. . . . .	9
2.2.	<b>Left: Edge feature computation.</b> This illustration shows the computation of an edge feature $e_{ij}$ from a pair of points $x_i$ and $x_j$ . The function $h_{\Theta}()$ denotes the edge features computed over a fully connected layer. <b>Right: The EdgeConv operation.</b> This operation aggregates the edge features for each edge starting at point $x_i$ . Image extracted from DGCNN [48] description. . . . .	10
2.3.	<b>DGCNN architecture.</b> The network consists of a spatial transformation module that aligns the input to a canonical space. Features are then extracted and aggregated at the end of the network for segmentation (bottom) and classification (top). The symbol $\oplus$ denotes the concatenation operation. For segmentation tasks, the architecture further incorporates data from the categorical vector of the dataset and concatenates the global feature with local features from previous layers. . . . .	11
2.4.	<b>PointTransformer architecture.</b> The network consists of several PointTransformer layers connected by ‘transition down’ blocks for encoding and ‘transition up’ blocks for decoding. It consists entirely of transformer layers, pointwise transformations, and pooling operations. . . . .	12

2.5.	<b>PointTransformer Layer.</b> This is a self-attention layer applied to an input of $N_1$ points, each with $f$ features. It performs linear projections and MLP layers on each input point with respect to its $k$ nearest neighbors. . . . .	12
2.6.	<b>Point-MLP (top) and Point-MLP elite (bottom) architectures.</b> The structure is divided into several point MLP stages, each consisting entirely of a geometric affine module, multilayer perceptrons, and max-pooling operations. .	14
2.7.	<b>BYOL architecture.</b> It trains to increase the similarity between $q_\theta(z_\theta)$ and the result of applying a stop-gradient operation [27] at $z'_\theta$ . Where $\theta$ are the trained weights and $\xi$ e is an exponential moving average of $\theta$ . At the end of training, all but $f_\theta$ are discarded and $y_\theta$ is used as the image representation. .	17
2.8.	<b>SIMSIAM architecture.</b> It consists of two instances of an encoder network $f$ , a projector network $g$ , a predictor network $q$ , and a gradient stopping process.	18
2.9.	<b>Point-MAE architecture.</b> Consists of three main stages: masking, embedding, and autoencoding. First, the point cloud is partitioned into patches that are randomly masked and then embedded. This is followed by a pre-training phase of the autoencoder, where the encoder is fed only the visible tokens, while the decoder receives both the mask tokens and the embeddings of the visible tokens. This design allows the model to learn to reconstruct the full input from a partially observed dataset, thereby capturing the underlying structure of the point cloud data. . . . .	20
4.1.	Linear transformation and masking pipeline illustration. . . . .	31
4.2.	Linear transformation and masking on ModelNet [11] chair sample. Masking ratio is setted to 60% of the total points in the model. . . . .	32
4.3.	<b>PointSIMSIAM architecture.</b> It is the same architecture than SIMSIAM [28] for 2D images, but it applies 3D linear transformation and masking to the input. It can be built to pre-train any 3D encoder. . . . .	35
4.4.	A visual illustration of point clouds in SimpleShape datasets. . . . .	37
8.1.	<b>Siamese networks t-SNE results part 1.</b> t-SNE is used to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-70, 70]. . . . .	71
8.2.	<b>Siamese networks t-SNE results part 2.</b> t-SNE is used to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-70, 70]. . . . .	72
8.3.	<b>Siamese networks UMAP results part 1.</b> UMAP is applied to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-20, 20]. . . . .	73
8.4.	<b>Siamese networks UMAP results part 2.</b> UMAP is applied to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-20, 20]. . . . .	74

8.5.	<b>Visual Representation of retrieval.</b> Objects randomly selected from ModelNet are displayed alongside their 5 nearest elements in the feature space, according to output from PointSIMSIAM with Point-MLP encoder. . . . .	77
8.6.	<b>SimpleShape t-SNE results part 1.</b> t-SNE is used to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-70, 70]. . . . .	84
8.7.	<b>SimpleShape t-SNE results part 2.</b> t-SNE is used to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-70, 70]. . . . .	85
8.8.	<b>SimpleShape UMAP results part 1.</b> UMAP is applied to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-20, 20]. . . . .	86
8.9.	<b>SimpleShape UMAP results part 2.</b> UMAP is applied to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-20, 20]. . . . .	87
8.10.	<b>Visual Representation of Retrieval for SimpleShape.</b> Objects randomly selected from ModelNet are displayed next to their 5 nearest elements in feature space, according to the output of Point-MAE pre-trained with SimpleShape15K. . . . .	90

# Chapter 1

## Introduction

Neural networks, often referred to as artificial neural networks (ANNs) [1], are computational models inspired by the intricate workings of the human brain in its decision-making processes. These networks are composed of multiple interconnected units called neurons that work together to acquire knowledge and make predictions. Neural networks can learn in a variety of ways, including supervised, self-supervised, or unsupervised learning. The extensive research and study of these networks forms the basis of the burgeoning field of deep learning (DL) [2].

Deep learning models are known for their need for large amounts of data compared to other machine learning methods, high computational complexity, lack of interpretability of their predictions, and excellent performance on complex machine learning problems that require learning patterns and representations of the objects under study [3]. These models have made significant advances in tasks such as classification, segmentation, object detection, object reconstruction, text translation, and text generation, especially in 2D image space, natural language processing, and 3D space.

In recent years, significant advances in technology have led to the development of modern techniques and devices capable of capturing the three-dimensional shapes of various objects [4, 5]. These three-dimensional representations serve as valuable inputs for computational algorithms to analyze and understand. Our research is specifically focused on the application of neural networks for pattern and entity recognition in 3D point clouds, due to their growing importance in fields such as robotics, virtual reality, and the automotive industry [6–9].

### 1.1. Point clouds learning

Point clouds are a common three-dimensional representation used in computer vision and robotics. They are composed of points in 3D space, each representing a location in the captured scene. These points can be generated using various sensing modalities, including LiDAR, depth sensors, or structured light scanners. In practical applications, point clouds

are a crucial input for algorithms that detect, classify, segment, and reconstruct objects.

The unstructured nature of point clouds presents significant computational challenges. Effective processing requires specialized algorithms that can handle issues such as noise, missing data, and variations in point density. Despite the challenges, point clouds are a valuable resource that provide detailed insight into the geometry and spatial relationships of objects in their environment. This rich source of information makes them indispensable in a wide range of applications, and they are a central tool for understanding and analyzing complex three-dimensional scenes.

## 1.2. Self-supervised learning

The availability of 3D data for neural network training is increasing, with well-known datasets such as ShapeNet [10], ModelNet [11], and ScanNet [12] gaining recognition. Deep learning research demands significant amounts of data, and the existence of rich and diverse datasets is crucial in enabling the training of neural networks with exceptional capabilities to tackle complex tasks.

However, obtaining new data samples presents significant challenges. Manual labeling of 3D data is a resource-intensive and time-consuming process, which limits dataset expansion. Furthermore, point clouds are unstructured and have variable point density, unlike the structured pixel grids of 2D images with a fixed distribution at each point. The inherent characteristic of point clouds makes it challenging for neural networks to extract and learn features from three-dimensional representations effectively.

To overcome the challenge of manually annotating datasets, self-supervised learning (SSL) techniques can differentiate records based on the points themselves, eliminating the need for manual labeling. This differentiation among samples enables models to learn features independently and transfer their knowledge to other networks responsible for subsequent classification or segmentation tasks.

Supervised machine learning is a systematic process that trains a model to represent a function  $y' = f(x)$  using a dataset  $D_s = \{x_i^{(s)}, y_i^{(s)}\}_{i=1}^N$ , enabling the model to predict  $y'$  values for new, previously unknown samples of  $x$ . In the context of deep learning, the predictive model consists of a feature extractor function  $h_\theta$ , known as the backbone, and a task-specific head function  $g_\Phi$ , known as the head network, which together compose  $f(x) = g_\Phi(h_\theta(x))$ . The model then adjusts  $h_\theta$  and  $g_\Phi$  to minimize a loss function  $L$ , such as the negative log-likelihood of Equation (1.1), as described by Ericsson L. et al. [13].

$$\arg \min_{\theta, \Phi} \sum L(g_\Phi(h_\theta(x_i^s), y_i^s)) \quad (1.1)$$

The self-supervised learning process revolves around the creation of pseudo-labels for a dataset  $D = \{x_i\}_{i=1}^M$ , with a size of  $M$ . It starts with a pre-training neural network, which is

primarily focused on an initial task known as a pretext task, denoted as  $P$ . This pretext task generates pairs of data points  $\{x_i, z_i\}_{i=1}^M = P(D)$ . Subsequently, a new network, represented as  $k_\gamma(h_\theta(\cdot))$ , is trained to predict the value  $z$  from its corresponding pair  $x$ . Once the training of this model is completed, a knowledge transfer step is performed to extract the parameters  $\theta^*$  of the Equation (1.1). These extracted parameters are then transferred to start the training of a new deep learning architecture, represented as  $g_\Phi(h_{\theta^*}(\cdot))$ , which is intended to perform various subsequent tasks, as mentioned above. The process of self-monitored pre-training, knowledge transfer, and the development of subsequent tasks is visually illustrated in Figure 1.1

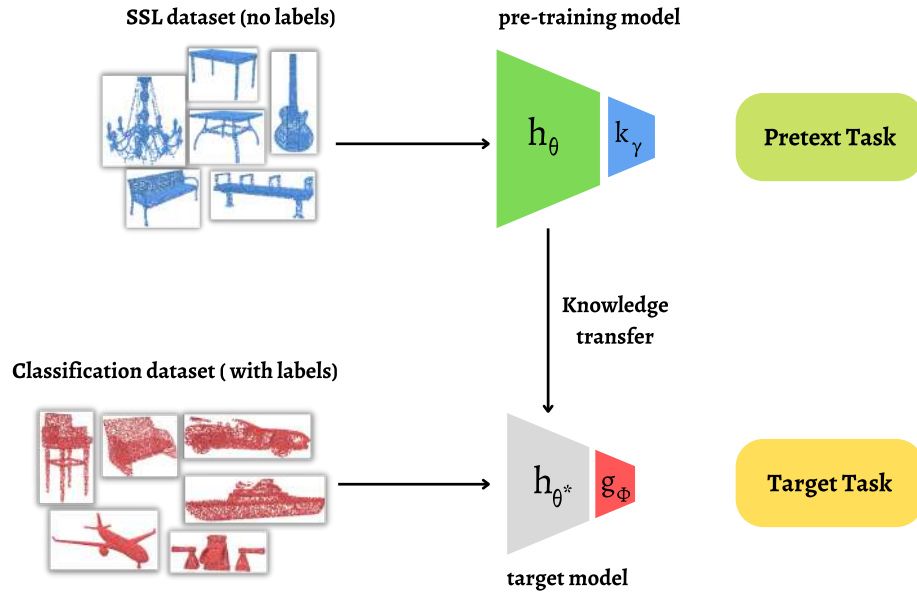


Figure 1.1: **Transfer learning process.** The pre-training model performs a pre-training task in which it learns  $\theta^*$  parameters that it will transfer as initial parameters to the subsequent model to initiate its training and achieve the target prediction task.

Given the tasks of machine learning in the domain of three-dimensional models, the challenges involved, and the growing demand for high-quality and large datasets, it is important to implement efficient and effective self-supervised machine learning methods.

Understanding 3D scenes is essential for various tasks, such as robotic grasping and self-driving navigation [14–16]. However, most approaches to this challenge are fully supervised, meaning they heavily rely on annotated 3D data. This process is often time-consuming and resource-intensive.

Recent advances in 3D self-supervised learning have focused on autoencoders [17–20]. In autoencoders, an unsupervised approach, the model learns to represent data features through two main networks: an encoder and a decoder. The encoder network transforms the complex

raw point cloud input into a compact coded representation, which the decoder then attempts to invert. During each training step, the decoder attempts to reconstruct the input sample while optimizing the architectural parameters. A notable variant of autoencoders are masked autoencoders [15, 21–26], which incorporate initial masking transformations that effectively hide certain data from the input. This technique encourages the auto-encoder to learn the features of the object from the masked version of the sample, and then recreate the original shape of the point cloud before the masking process.

Additional self-supervised deep learning techniques, such as BYOL [27] and SIMSIAM[28], prioritize contrastive learning and achieve remarkable results on image-related tasks using simpler neural network architectures. This is in contrast to many state-of-the-art proposals that rely on complex compositions of transformational models. However, not enough work has been done to replicate the advances of Siamese networks with three-dimensional data.

### 1.3. Synthetic datasets

In addition to developing novel self-supervised learning models and techniques, one way to enhance representation learning is to acquire new datasets for pre-training tasks. As previously mentioned, ShapeNet is the primary dataset for pre-training networks. However, recent developments, such as the introduction of the SimpleShape dataset [29], a synthetic dataset created from closed planar curves used as the directrices of cylinders and cones, provide a promising alternative for research and experimentation.

SimpleShape has the advantage of being a synthetic dataset that can be easily extended as needed, making it a convenient source of samples for pre-training neural networks. Since the focus of this work is to improve representation learning of 3D shapes, this artificial dataset can serve as input to a training pipeline.

In this context, a point MAE model [23] was trained using different versions of SimpleShape with different sample sizes ranging from 1000 to 100,000. If these experiments yield results similar to or better than those obtained from pre-training with ShapeNet, it would open the possibility of considering synthetic datasets as a valuable addition to the pre-training process.

The goal of this research is to create an experimental approach that uses self-supervised learning techniques to generate 3D representations from unlabeled datasets. These representations are then used to improve classification and segmentation tasks. Our proposed approach incorporates architectures inspired by SIMSIAM and BYOL, which are specifically designed to optimize efficiency and outperform current state-of-the-art methods by reducing data requirements, resource consumption, and training time.

In addition, we expect that achieving competitive results with SimpleShape pre-training will stimulate the adoption of synthetic datasets in self-supervised learning, thereby expanding the potential of pre-training with copyright-free samples.

## 1.4. Document structure

The following is the structure of this thesis:

1. A complete review of the state of the art related to self-supervised learning on 3D point clouds is presented in Chapter 2.
2. The methodology is described in Chapter 3, where the problem statement, research questions, hypothesis, and goals are found.
3. The models proposed in this thesis are presented in Chapter 4. The subsequent Chapters describes the datasets (5) and evaluation metrics (6) to be employed in the experiments in Chapter 7. Finally, the results and analysis of the experiments in Chapter 8, as well as the general conclusions in Chapter 9.



# Chapter 2

## Related work

This chapter provides an in-depth review of the state of the art in 3D point cloud deep learning. First, work on point cloud learning in general (2.1) is summarized, describing the main networks for this type of data and their respective encoders. Then, the literature related to self-supervised learning based on autoencoders and Siamese networks is described, ending with an analysis of the works that combine both ideas in the field of self-supervised learning on 3D point clouds and the research gap in this area of study.

### 2.1. Point cloud learning

Point cloud learning is the study of 3D point clouds, developing deep learning models that can understand their shapes and structures. Several methods have been developed to address the main challenges of working with point clouds, such as permutation invariance, noise, variable densities, and the limited number of samples available for training.

Point clouds typically consist of spatial data represented by a set of points with three variables. Additional data such as color, normals, and labels may also be included. The goal is to create general models for these types of clouds that can work with spatial data alone, using other features only when necessary.

One of the major challenges in developing methods for point clouds is the lack of data, unlike images, which have datasets such as ImageNet [30, 31] with 14M annotated images, CIFAR-10 and CIFAR-100 [32] with 60K annotated images, COCO [33] with 328K annotated images, and IMDB-Wiki [34] with more than 500K images. The most commonly used point cloud datasets are ModelNet40 [11] with 12K objects, ShapeNet [10] with 50K objects, ScanObjectNN [35] with 3K objects, ScanNet [12] with 15K frames, and KITTI [36], also with 15K frames.

The scarcity of large amounts of available labeled data adds to the inherent challenges of point clouds, which, unlike images, have areas of varying density and more sparse zones. This complicates learning, as methods must be able to relate points globally or locally, focusing

attention on the most descriptive areas of the object, which may be very small and dense, or very large with few points.

Guo, Yulan et al. [37] describe in detail the various current proposals for point cloud learning. For classification tasks, methods typically work by learning an embedding for each point in the cloud and then forming a global embedding for the entire shape by aggregating the previous embeddings. The resulting embedding is then used as input to fully connected layers that determine the class to which the method considers the object to belong. These methods can be divided into multi-view based, volumetric based, and point based methods.

Multi-view based methods, as the name suggests, generate different 2D views by projecting the object onto planes around it. From these views, they extract an embedding from each view and aggregate them to perform the final classification. MVCNN [38] was the first model of this technique, which simply combines the embeddings by max-pooling. In addition, several works use this technique, including models that exploit the integration of convolutional features [39] or aggregate data using graph convolutions [40].

Volumetric-based methods enclose the point cloud in 3D meshes and then apply 3D or 2D convolutional nets to each plane. VoxNet [41] followed this idea with a volumetric occupancy network model. However, a major problem with these methods is scalability, as computation and memory grow cubically with the grid resolution. For this reason, later methods have proposed using octrees as a specialized data structure for 3D networks [42, 43]. Finally, there are convolutional occupancy networks [44], which instead of using 3D grids in their convolutions, use three 2D planes to extract features from the object, one plane for each axis.

Point-based methods work directly on the input point cloud. They model each point in the cloud using common Multi-Layer Perceptron (MLP) layers, and then aggregate the features obtained for each point into a global feature for the entire object. PointNet [45] introduced this technique, achieving permutation invariance of the features of each point by aggregating the points with a symmetric function. PointNet++ [46] followed the same idea, but added point features at different levels, allowing it to learn information about local structures in shapes.

Xu Ma et al. [47] point out that “most 3D point cloud applications are still based on the simple PointNet (and PointNet++) or voxel-based methods. This is because sophisticated extractors that explore fine geometric properties have recently been developed. However, due to their computational complexity and memory requirements, they reduce the efficiency in natural scene applications. There are three categories of point cloud representation models: convolution-based, graph-based, and attention-based.

As mentioned earlier, convolution-based methods extract point features by enclosing them in 2D or 3D grids and then applying convolution operations. On the other hand, graph-based methods connect the points in the cloud through graphs, considering each point as a vertex and generating directed edges between them. Graph features are obtained by typical graph-based networks, as shown in the Figure 2.2, which represents the basic operation of DGCNN

[48]. In this way, features are obtained by grouping features by point neighborhoods, a similar approach to PointNet++.

One of the most relevant works in this area is DGCNN [48], which constructs a graph in the feature space and dynamically updates it after each layer. It uses MLP as the learning function on the edges and applies channel-wise symmetric aggregation to obtain neighborhood features. New methods have emerged from this work, focusing on improving DGCNN, relating hierarchical features of each layer [49], building autoencoders [50, 51], simplifying aggregation [52], and reducing model runtime [53].

In terms of attention-based models, the PointTransformer [54] was created inspired by the Transformer [55] models, which have achieved excellent results in natural language processing and image processing tasks. Zhao et al. [54] mention that while there were already models that used attention mechanisms to process point clouds [56–59], these models applied attention to the entire cloud, which required excessive computation, making them inapplicable for learning large 3D scenes. A key advantage of PointTransformer was that it applied self-attention locally, allowing it to scale to large 3D scenes.

Xu Ma et al. [47] introduce Point-MLP, a new model based solely on MLP operations, similar to PointNet and PointNet++. It is a residual feedforward MLP network that aggregates local features extracted by hierarchical MLPs, to which they add a lightweight geometric affine module that transforms the points into a normal distribution. Although their model achieves excellent results, it uses a large number of parameters (12.6M). Therefore, they also present an elite version of Point-MLP with much fewer parameters (0.68M), which achieves similar results with reduced training and inference time.

While these models are known for their point cloud classification version, they have direct applications in other tasks such as segmentation. This can be done by simply changing the network header to a segmentation header, or by more complex mechanisms such as linking the header to previous layers. The working methods of the PointNet, DGCNN, PointTransformer, and Point-MLP models are described below.

## 2.1.1. PointNet

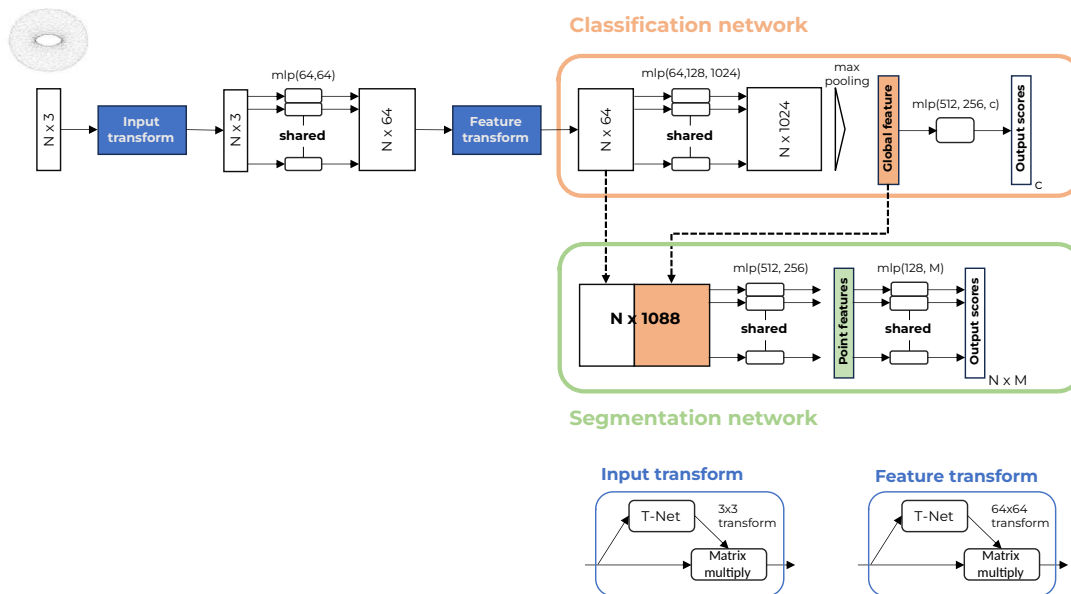


Figure 2.1: **PointNet architecture.** The network takes as input a set of  $N$  points from a given point cloud. The network is designed to perform both sample and feature transformations, which are then aggregated using a max-pooling strategy. This process facilitates the network’s ability to produce a classification score for  $c$  different classes. Building on the basic classification network, the segmentation network is extended to integrate both global and local features through concatenation. In this architecture, ‘mlp’ denotes multi-layer perceptrons, with the subsequent layer sizes given in parentheses. To ensure stability and improve convergence during training, batch normalization is applied across all layers in conjunction with the ReLU activation function.

PointNet [45] focuses on adapting to three main properties of point sets in  $\mathbb{R}^n$ . These are invariance to affine transformations such as rotation and translation of all points, invariance to permutation of the input points, and interaction between nearby points.

The complete network is shown in Figure 2.1. It contains two T-Net [60] networks at the beginning, the first of which is responsible for aligning the point cloud to a canonical space, while the second aligns the point features in later stages of the process. The purpose of these networks is to make the model invariant to affine transformations, like point clouds. As a result, PointNet achieves better performance in classification tasks on ModelNet40.

To extract features from point clouds, we mainly use shared multilayer perceptron (MLP) layers, which transform the cloud of  $N$  points with dimension  $N \times 3$  into a  $d$  dimensional space  $N \times d$ . For classification, a symmetric function, specifically max pooling, is applied over the features, resulting in a global feature embedding of dimension  $N$ . Since max pooling

is a symmetric function, it ensures that the model is invariant to permutations among the points; the result remains unchanged if the points are permuted.

In the case of segmentation, the  $N$  vectors of  $d$  dimensions are concatenated with the global feature vectors of  $d$  dimensions. This results in a feature matrix that includes both the local information extracted before max-pooling and the global information extracted by max-pooling. Both headers are then terminated with MLP layers, which are responsible for generating the output scores for the target task.

While PointNet lacks features that other models include, such as capturing hierarchical features through neighborhoods as PointNet++ [46] does, it remains one of the most widely used networks when working with point clouds. This is due to its ease of implementation, competitive results, and modest computational and memory requirements.

### 2.1.2. DGCNN

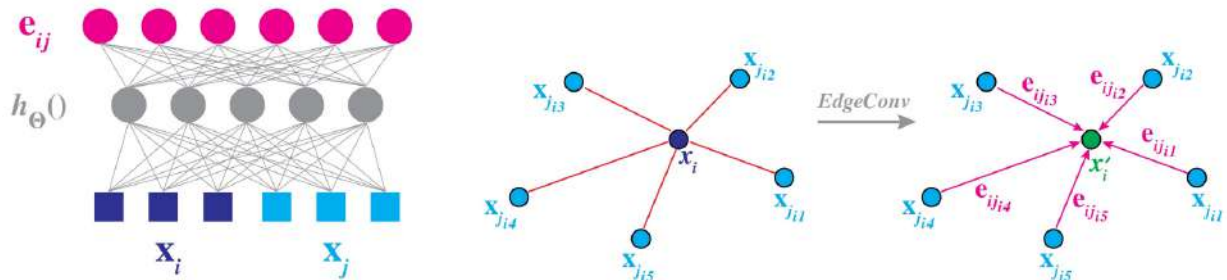


Figure 2.2: **Left: Edge feature computation.** This illustration shows the computation of an edge feature  $e_{ij}$  from a pair of points  $x_i$  and  $x_j$ . The function  $h_{\Theta}()$  denotes the edge features computed over a fully connected layer. **Right: The EdgeConv operation.** This operation aggregates the edge features for each edge starting at point  $x_i$ . Image extracted from DGCNN [48] description.

DGCNN [48] introduces the ability to recover geometric relationships between points, an aspect that PointNet lacks due to its methodology of extracting features from points independently. They propose an operation called EdgeConv, which generates features through edges that describe the relationship between each point and its neighbors. This operation is invariant to the order of neighbors, allowing for permutation invariance in the point cloud.

First, a directed graph of neighborhoods is computed, where each point is connected to its  $K$  nearest neighbors. Edges connect the points, and edge features are defined as a function of trainable parameters. EdgeConv is then defined as a channel-wise symmetric aggregation operation applied to each point, aggregating its corresponding  $K$  edge features. This operation, shown in Figure 2.2, is implemented as a layer of shared MLPs, similar to the components of PointNet.

The graphs needed to compute EdgeConv are dynamically created in feature space after each layer. The network learns how to construct these graphs, with a different graph for each layer. The implementation involves computing a pairwise distance matrix and grouping the  $K$  nearest neighbors for each point in the feature space.

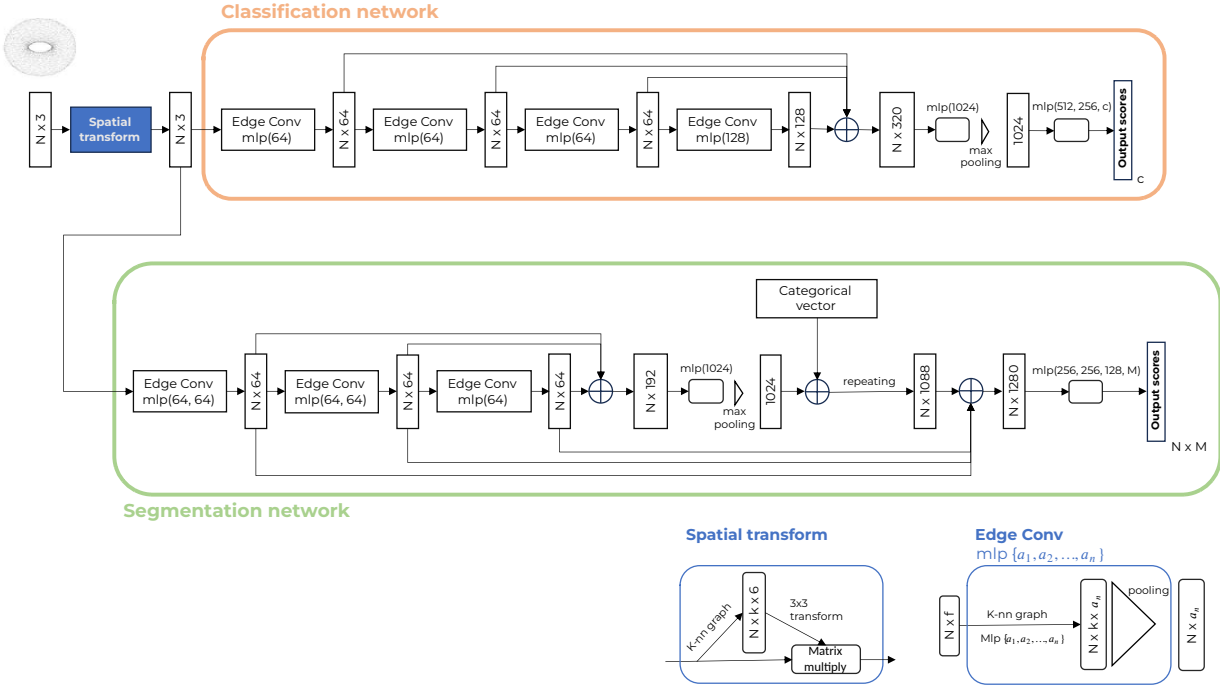


Figure 2.3: **DGCNN architecture.** The network consists of a spatial transformation module that aligns the input to a canonical space. Features are then extracted and aggregated at the end of the network for segmentation (bottom) and classification (top). The symbol  $\oplus$  denotes the concatenation operation. For segmentation tasks, the architecture further incorporates data from the categorical vector of the dataset and concatenates the global feature with local features from previous layers.

For classification tasks, the upper branch of Figure 2.3 is followed, using four EdgeConv layers working with  $K = 20$  neighbors for each point. Fully connected layers are interspersed between the EdgeConv layers to extract features. The features from these layers are concatenated, and a fully connected layer with max pooling is applied to obtain the global features of the point cloud. This is processed through two fully connected layers to obtain a  $c$  dimensional vector of classification scores.

For segmentation, the architecture is modified by adding EdgeConv layers to the model and concatenating the global features with local features from previous layers, similar to PointNet. In addition, they incorporate the categorical vector containing ShapeNetPart [61], the dataset on which segmentation is performed.

### 2.1.3. PointTransformer

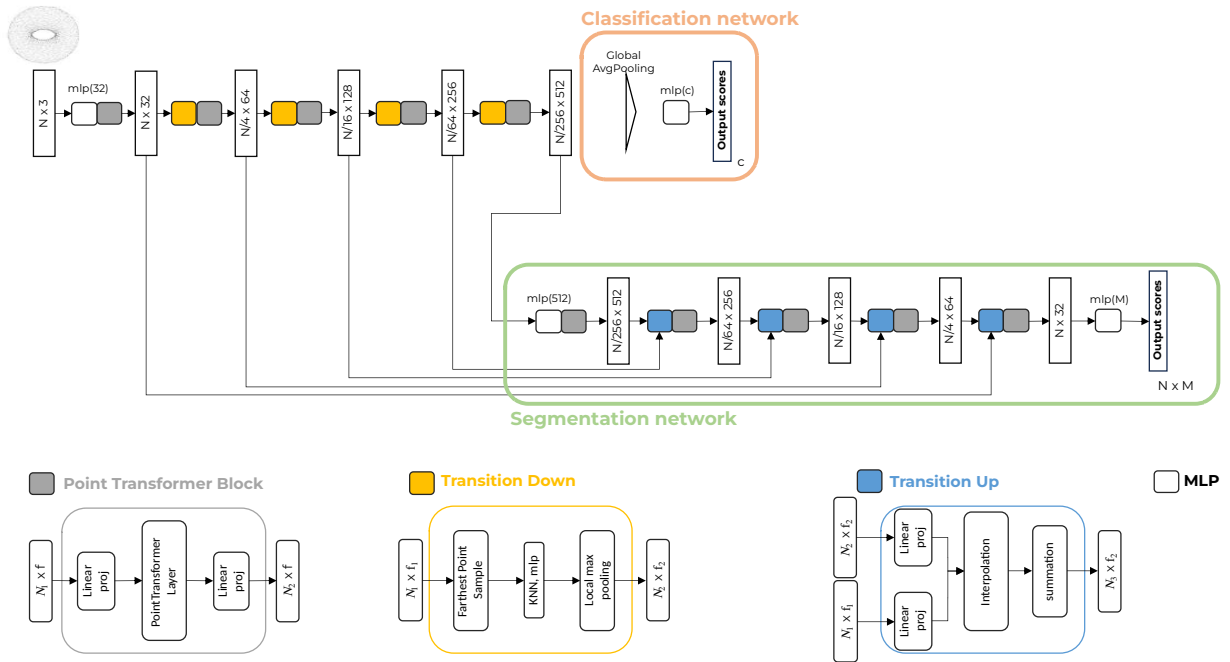


Figure 2.4: **PointTransformer architecture.** The network consists of several PointTransformer layers connected by ‘transition down’ blocks for encoding and ‘transition up’ blocks for decoding. It consists entirely of transformer layers, pointwise transformations, and pooling operations.

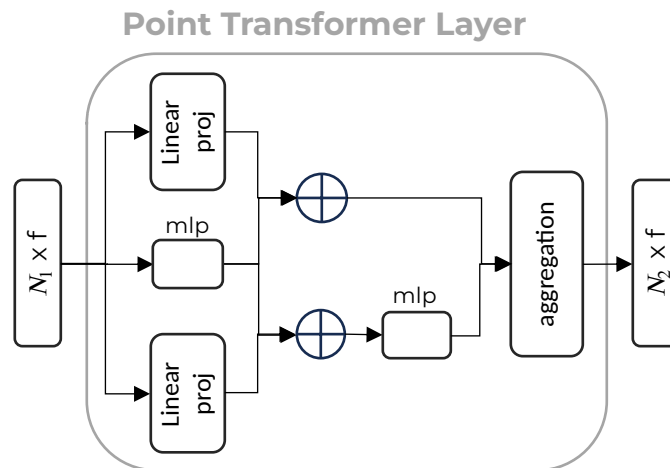


Figure 2.5: **PointTransformer Layer.** This is a self-attention layer applied to an input of  $N_1$  points, each with  $f$  features. It performs linear projections and MLP layers on each input point with respect to its  $k$  nearest neighbors.

The PointTransformer [54] uses the capabilities of self-attention networks to perform classification and segmentation tasks on 3D point clouds. These networks are particularly suited for this type of input because self-attention operates as a set operator, i.e. it is invariant to the permutation of elements. This property is well suited to the primary characteristic of 3D clouds: the invariance of their points.

A PointTransformer layer is developed based on vector self-attention, as shown in Figure 2.5. The set of points  $\mathcal{P}$  is first divided into local neighborhoods centered on each point using KNN (k-nearest neighbors). Then two linear projections are applied to the input and a position coding is obtained by an MLP. These results are fed into a new MLP layer and finally aggregated to produce a result that preserves the number of input features  $f$  but can change the number of points  $N$ .

From the PointTransformer layer, the PointTransformer architecture shown in Figure 2.4 is created. The encoder progressively downsamples the input and implements a PointTransformer layer at each level. For classification, global average pooling and a simple MLP at the end of the encoder are used to obtain the output. For segmentation, transition up layers are applied in conjunction with PointTransformer layers, resulting in a feature vector for each point in the set. The resulting structure for segmentation follows the U-Net [62] model, connecting previous layers at each upsampling level.



## 2.1.4. Point-MLP

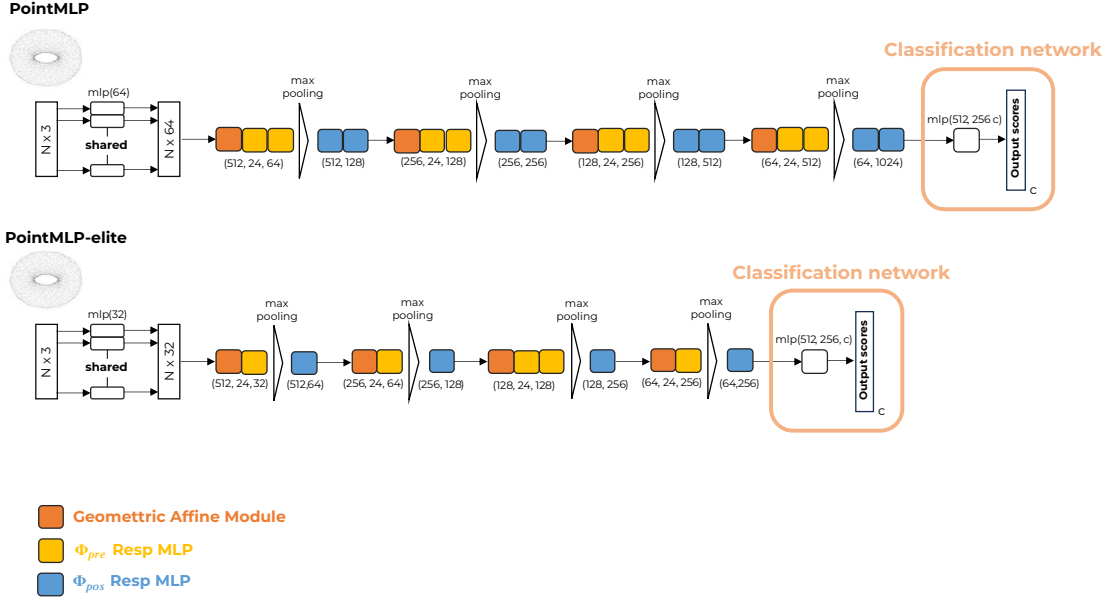


Figure 2.6: **Point-MLP (top) and Point-MLP elite (bottom) architectures.** The structure is divided into several point MLP stages, each consisting entirely of a geometric affine module, multilayer perceptrons, and max-pooling operations.

Point-MLP [47] contends that the development of more sophisticated feature extractors has become oversaturated, offering no significant performance improvements relative to the computational and storage costs involved. They present an architecture based primarily on feature extraction by residual MLP blocks. The core operation of Point-MLP is formulated according to the Equation (2.1):

$$g_i = \Phi_{pos}(\mathcal{A}(\Phi_{pre}(f_{i,j})|j = 1, \dots, K)) \quad (2.1)$$

Where  $\Phi_{pre}(\cdot)$  and  $\Phi_{pos}(\cdot)$  represent residual MLP blocks.  $\Phi_{pre}(\cdot)$  is a shared block that learns weights from a local region, while  $\Phi_{pos}(\cdot)$  is tasked with extracting deep aggregated features. The aggregation function  $\mathcal{A}$  that they use is the max-pooling operation. The operation from Equation (2.1) refers to a Point-MLP stage, which is recursively repeated for  $s$  stages. To select the  $k$  nearest neighbors of each point, the KNN algorithm is employed.

To deal with density variations within point clouds, a geometric affine module is applied that transforms the point features of a local region into a normal distribution. This module, combined with the MLP block stages, enables Point-MLP to achieve exceptional results in classification and regression tasks with much simpler feature extractors compared to other

state-of-the-art models.

In addition to the original Point-MLP structure, they propose a lightweight version called Point-MLP elite. This architecture reduces the number of residual MLP blocks, reduces the embedding dimension from 64 to 32, and introduces a bottleneck structure that significantly reduces the number of parameters in the network. By combining these features, they are able to reduce the number of parameters from 12.6M to only 0.68M, while still achieving competitive results on various benchmarks. Both structures are shown in Figure 2.6.

For segmentation tasks, the network header can be modified by replacing the MLP network with continuous interpolation layers and a lightweight pointNet. This modification allows the network to produce a segmentation output of dimension  $N \text{ times } M$ , where  $N$  is the number of points and  $M$  is the number of classes. Additionally, at each interpolation level, skip-link concatenations are added with outputs from previous point-MLP stages.

## 2.2. Self-supervised learning

In self-supervised learning (SSL), Liu, Xiao, et al. [63] present two main methods for training SSL models: a) generating classification labels from insights automatically derived from the data, and b) predicting data subsets by analyzing different data segments. These methods act as pretense tasks to avoid the need to manually label data sets. They also identify three categories of SSL:

- Generative: Involves training an encoder network to translate an input  $x$  into a distinct feature vector  $z$  that encapsulates the features extracted from the input data, along with a decoder network tasked with reconstructing  $x$  from  $z$ .
- Contrastive: Focuses on training an encoder network to encode an input  $x$  into a feature vector  $z$ , similar to generative models. However, the  $z$  vector is used to detect similarities between different entities during the training process.
- Generative-Contrastive (Adversarial): Involves training an encoder-decoder network to produce false data instances, and a discriminative network that attempts to determine whether its inputs are from the original data set or the creations of the encoder-decoder network.

### 2.2.1. Autoencoders

Autoencoders [64] are a fundamental generative method in self-supervised learning (SSL). They aim to reconstruct the input data as closely as possible to the original input, often using mean square error as the loss function for images. Denoising Autoencoders (DAE) increase robustness by intentionally introducing noise, such as pixel masking. [65–67] or colour channel removal [68].

### 2.2.1.1. Masked autoencoders

Masked autoencoders (MAE), as introduced by Pascal Vincent et al. [65], hide parts of the input and train the model to reconstruct the whole from the remaining data. This concept is used in BERT [69] for natural language processing, where random tokens are masked for the autoencoder to predict. In computer vision, techniques such as SimMIM [70] use similar principles with image patches. Meanwhile, in three dimensions, there are different types of entities, such as three-dimensional networks, where Yaqian Liang et al. [71] proposes to adapt MAE to this type of sample, considering all the challenges of its structure composed of vertices and faces with no specific order.

### 2.2.2. Siamese networks

Siamese networks [72] are a type of neural network architecture used for metric learning, assessing the similarity or dissimilarity between two entities. Within this subset are contrastive networks [73], which are particularly effective for image representation [74–80]. They work by bringing closer the representations of altered versions of the same image (positive pairs) and distancing the representations of different images (negative pairs).

To train Siamese networks, positive pairs can be generated by modifying inputs, such as applying rotations or translations to create variations of the same image. Contrastive learning, which is widely used in image-based SSL, typically requires a substantial number of negative samples and mechanisms to maintain them throughout training iterations. SimCLR [81], for example, requires large batch sizes, while MoCo [82] uses a queue of negative samples and a momentum encoder for consistency.

Negative pairs, ideally very different from the original, can be more difficult to generate and are typically needed in larger numbers than positive pairs. Recent advances aim to reduce the dependence on large data sets. In particular, BYOL [27] and SIMSIAM [28] have shown promising results without the use of negative pairs in their training processes. These innovations represent a shift towards more data efficient training models in SSL.

### 2.2.2.1. BYOL

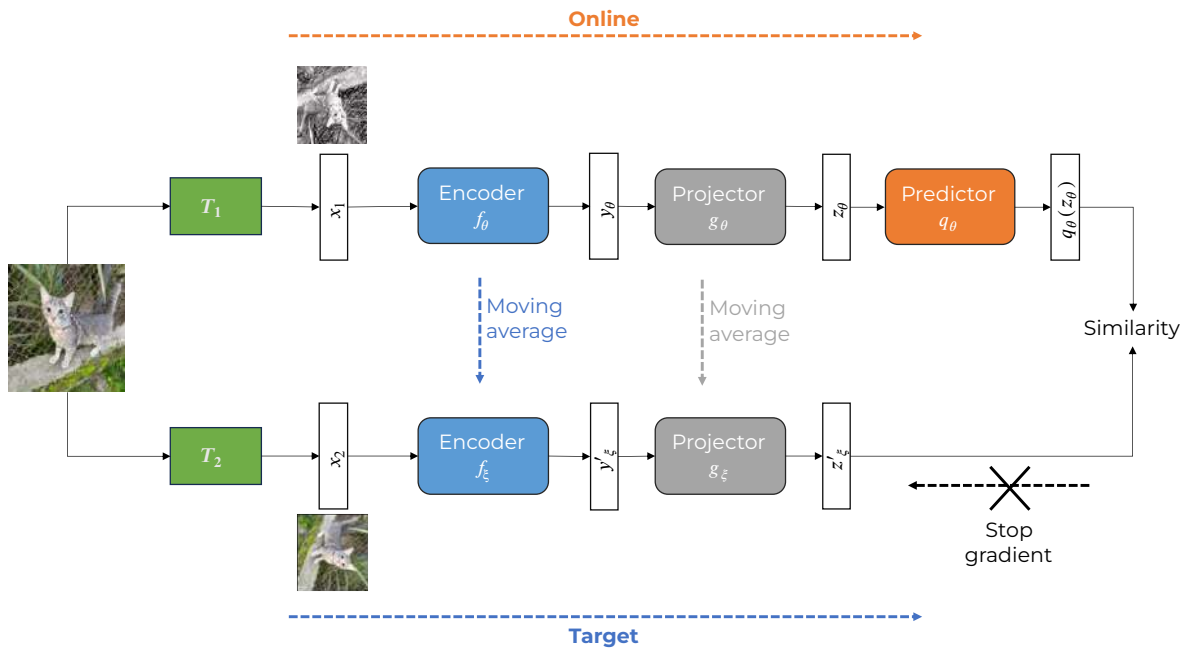


Figure 2.7: **BYOL architecture.** It trains to increase the similarity between  $q_\theta(z_\theta)$  and the result of applying a stop-gradient operation [27] at  $z'_\xi$ . Where  $\theta$  are the trained weights and  $\xi$  is an exponential moving average of  $\theta$ . At the end of training, all but  $f_\theta$  are discarded and  $y_\theta$  is used as the image representation.

BYOL [27] is a Siamese neural network architecture that uses two neural subnetworks, target and online, that interact and learn from each other. Figure 2.7 describes the model in detail. To train, it modifies views of an image and trains the online network to predict the representation that the target network will produce from the same input sample but with different modifications. Meanwhile, the weights learned by the target network follow a moving average of the online network.

BYOL is unique in that it does not require negative pairs, but only augmented versions of the input images. This feature drastically reduces the amount of data and computation needed for these networks to learn, given that using negative pairs requires each example to be compared to many others in order to perform well.

### 2.2.2.2. SIMSIAM

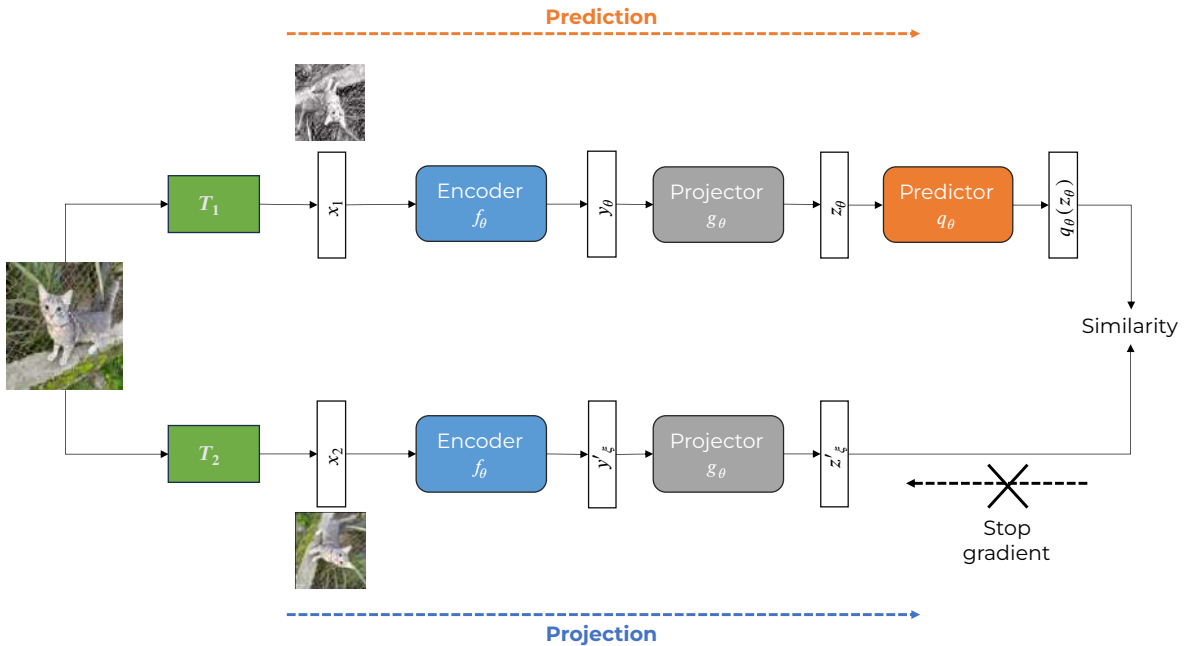


Figure 2.8: **SIMSIAM architecture**. It consists of two instances of an encoder network  $f$ , a projector network  $g$ , a predictor network  $q$ , and a gradient stopping process.

SIMSIAM [28] is a neural network that performs work similar to BYOL, but without the need for an momentum encoder represented by the target network. As shown in Figure 2.8, SIMSIAM processes two modified views of an image over the same network called the encoder  $f$ , which consists of a backbone network, commonly ResNet [83]. Then a multilayer perceptron  $g$ , a simpler neural network, projects the output into a feature vector. Finally, another multilayer perceptron  $q$  transforms the output of one of the projectors  $g$  and tries to match it with the result of the other instance after applying a gradient stopping operation.

The training aims to increase the similarity between the output of the gradient-stopped projection and the output of the predictor. The two views share the weights of the encoder  $f$  and the projector  $g$  during training, which is different from the BYOL idea. The negative cosine between  $p_1 = q(g(f(x_1)))$  and  $z_2 = g(f(x_2))$  is minimized by the formula of Equation (2.2) to increase the similarity between the two views.

$$\mathcal{D}(p_1, z_2) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{z_2}{\|z_2\|_2} \quad (2.2)$$

Equivalent to the mean squared error in  $l_2$  normalized vectors. Then define a symmetrized loss according to the formula of the Equation (2.3) with  $p_2 = q(g(f(x_2)))$  and  $z_1 = g(f(x_1))$ , which will be used to find the similarity of the outputs.

$$(L) = \frac{1}{2}\mathcal{D}(p_1, z_2) + \frac{1}{2}\mathcal{D}(p_2, z_1) \quad (2.3)$$

### 2.3. SSL on point clouds

While the development of new 3D shape capture technologies has increased, machine learning architectures that process such data face a significant bottleneck in obtaining large amounts of labeled data [84, 85]. This challenge has led to the adoption of self-supervised learning (SSL) techniques to achieve better results for tasks with insufficient labeled data.

Xiao et al. [84] and Fei et al. [85] have presented comprehensive reviews summarizing the state of the art in SSL applied to 3D point clouds. They explain that based on SSL ideas in images and text, there are several proposals to apply SSL to 3D point clouds, including point cloud self-reconstruction and contrastive learning, among others.

In point cloud self-reconstruction, autoencoders are considered representative of the method where the point cloud itself serves as ground truth. Models are trained to produce outputs that closely resemble the input. The encoder learns to generate low-dimensional representations that capture the most important information from the input while ignoring insignificant data that may be noise. From this generated representation, the decoder then attempts to reconstruct an output that closely matches the network’s input.

Building on the success of masked autoencoders in text, the concept of recovering masked regions of the input as a pretext task has been explored in 3D point clouds. Point-BERT [86] and Point-MAE [23] are the main proposals. These pre-train transform nets in a manner analogous to what BERT [69] and MAE [22] do for images. They use an autoencoder model to generate tokens from object patches, and randomly hide some tokens to train the network to recover the original point tokens. Finally, the decoder is discarded and the encoder is used to generate representations for downstream tasks.

### 2.3.1. Point-MAE

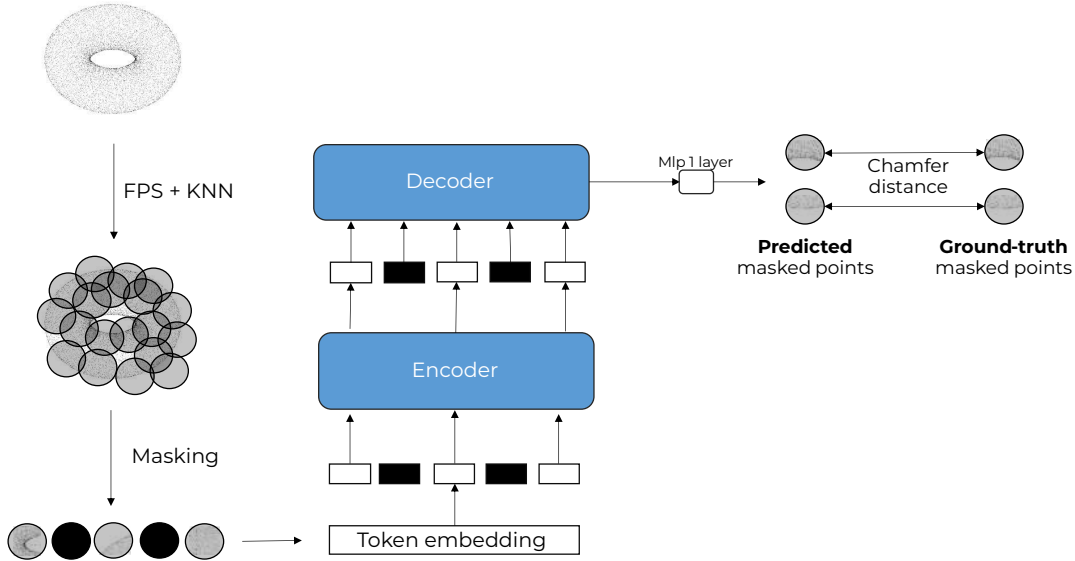


Figure 2.9: **Point-MAE architecture.** Consists of three main stages: masking, embedding, and autoencoding. First, the point cloud is partitioned into patches that are randomly masked and then embedded. This is followed by a pre-training phase of the autoencoder, where the encoder is fed only the visible tokens, while the decoder receives both the mask tokens and the embeddings of the visible tokens. This design allows the model to learn to reconstruct the full input from a partially observed dataset, thereby capturing the underlying structure of the point cloud data.

Point-MAE [23] introduces a masked autoencoder structure that consists of three components: a masking module, an embedding module, and an autoencoder module. The point cloud masking module begins by using the farthest point sampling algorithm [87] to select points  $C$  that are as far apart as possible. These points  $C$  serve as centers for the cloud patches. Then, the kNN algorithm is applied to select the  $k$  nearest neighbors for each previously obtained point  $C$ , forming patches  $P$  that partition the input cloud.

Since patches  $P$  may overlap, point clouds are masked by separately masking the patches. Randomly, between 60% and 80% of the generated patches are classified as ‘masked’  $P_m$ , while the rest are classified as ‘visible’  $P_v$ . The visible patches  $P_v$  then pass through an embedding module consisting of lightweight point nets (essentially shared MLPs and max-pooling layers), producing embeddings  $E_v$ .

The network proceeds by introducing the visible token embeddings  $E_v$  into an encoder composed of standard transformer blocks, resulting in a low-dimensional representation vector. Using this representation and the original mask tokens  $P_m$ , the decoder is tasked with

generating decoded mask tokens that are input to a final MLP prediction layer. The sole function of this layer is to generate predicted masked dot patches using a linear MLP layer and a reshaping operation.

The ability of the network to recover the coordinates of the points in the masked patches is then evaluated. By comparing the predicted point patches  $P_{pred}$  and the ground truth  $P_m$ , the reconstruction loss is calculated using the  $l_2$  chamfer distance [88], according to the Equation (2.4). This process ensures that the network learns to accurately approximate the geometry of the original point cloud.

$$L = \frac{1}{|P_{pred}|} \sum_{a \in P_{pred}} \min_{b \in P_m} \|a - b\|_2^2 + \frac{1}{|P_m|} \sum_{b \in P_m} \min_{a \in P_{pred}} \|a - b\|_2^2 \quad (2.4)$$

Using the Point-MAE methodology as a foundation, new methods have emerged that follow its core ideas. These proposals introduce enhancements aimed at exploiting the capabilities of masked autoencoders in learning point clouds. Among them is Point-M2AE [89], which, similar to PointNet++ [46], adds a hierarchical learning factor to the pre-training of the network, creating a multi-scale masking that can be used to train the encoder-decoder structure in a level-wise manner, following a U-net architecture [62]. In addition, Point-MA2E [90] proposes not only to use masking transformations, but also to apply linear transformations before the input enters the autoencoder.

### 2.3.2. Contrastive learning

Contrastive learning methods, similar to autoencoder-based approaches widely studied for 3D point clouds [20, 23–25, 86, 91], have been proposed for learning representations of synthetic object scenes [92–95]. PointContrast [96] was a pioneer in teaching networks to learn point cloud scene representations, demonstrating that such pre-training can improve performance on high-level 3D tasks such as classification, segmentation, and detection. DepthContrast [97] further advanced the field by handling single-view data.

Despite these advances, the techniques based on Siamese networks for images have not been extensively explored for 3D point clouds. Huang et al. [98] introduced a framework based on spatio-temporal representation learning using BYOL [27] to extract spatial and temporal features from point cloud sequences. This approach treats two consecutive frames as positive pairs and minimizes the mean squared error between their learned representations. Conclu [99] applies self-supervised learning with a SIMSIAM-based architecture, although it requires an additional clustering loss to prevent collapse. While several studies have extended BYOL to 3D point clouds [100–102], none have yet focused on the general ability of Siamese networks to generate representations of 3D point clouds, benchmarked against works such as Point-MAE or Point-BERT.

A significant challenge for Siamese networks on non-image data is the identification of appropriate input transformations. Point clouds, which are simply sets of points in three



dimensions, cannot undergo color-based transformations. Furthermore, they are invariant to permutation, which limits the applicability of transformations that would disrupt the point order. Huang et al. [98] use successive frames as positive pairs in training, naturally generating augmentations that could otherwise be synthetically produced by linear transformations, if the appearance and disappearance of objects in the scene is ignored.

Another hurdle is the linear transformation invariance as proposed by PointNet [45]. If an encoder is invariant to such transformations, then applying them to the input point clouds has no effect on the Siamese networks since both encoder outputs would be identical, resulting in a loss too small to facilitate further network learning.

Despite attempts to adapt BYOL from 2D data to 3D points, as of the current knowledge, no studies have examined other Siamese networks, like the more fundamental SIMSIAM [28], on 3D representations. Hence, it remains uncertain whether Siamese networks inherently allow for effective representation learning of 3D point clouds.

### **2.3.3. Formula-driven supervised learning**

A distinct self-supervised learning approach is formula-driven supervised learning [103, 104], which leverages the creation of synthetic data sets for pre-training neural networks. The term "formula-driven" refers to the generation of synthetic data based on mathematical formulations.

This pre-training method offers several advantages, including complete control over object features, freedom from data rights issues, and the potential for continuous improvement of the generated objects. By controlling the features of the generated objects, patterns that contribute most to the pre-training of neural networks can be identified, isolated, and used. Because the patterns are mathematically generated, the risk of copyright infringement is greatly reduced. In addition, synthetic data generation allows continuous improvement of image or model quality and the generation of as many training samples as needed.

This technique has been extensively studied for images, with results comparable to supervised training [103–107]. In the study of 3D point clouds, Yamada et al. [108] proposed the PC-FractalDB pre-training model based on fractal geometry. This dataset is created by defining fractal categories based on a variance threshold and instance augmentation with FractalNoiseMix. A 3D scene is then generated by selecting 3D fractal models and translating them from the origin. The dataset is then used for object detection as a pre-training task for the models. The results of PC-FractDB suggest that the use of formula-driven supervised learning on 3D point clouds could be a novel tool to address the bottleneck of lack of labeled data in this area of research.

### 2.3.3.1. SimpleShape

SimpleShape [29], a recently introduced dataset, is tailored for detecting symmetries in 3D point clouds. It uses closed plane curves as guides to generate three-dimensional shapes such as cylinders and cones. While the dataset initially contains 69,000 simple three-dimensional shapes, its mathematical formulation allows for the creation of an even larger number of samples.

The dataset generation process begins by randomly selecting a closed plane curve and setting random parameters for its creation. The point cloud is then generated by conical or cylindrical extrusion. The final step is to apply noise, translation, and rotation transformations to the generated shapes.

Although the dataset is primarily designed for geometry recognition, its formulation can be chosen to create synthetic point clouds. These synthetic clouds could be used in self-supervised learning on point cloud representations, benchmarked against common standards. This could help to assess the potential of using synthetic data in the study of 3D point clouds.

## 2.4. Research gaps in SSL on point clouds

As recent surveys show [84, 85], the main focus in recent years has been on pre-training autoencoder type networks with transformer type encoders in SSL on 3D point networks. This is understandable since this type of network is known to give the best results compared to other proposals. However, when looking for studies on other ideas, it is clear that there is a gap of proposals with other deep learning model structures.

Siamese networks have only been explored by Huang, et al. [98], which uses BYOL on 3D point clouds and under specific transformations for their dataset. While formula-driven supervised learning has also been understudied, most work has assumed that ShapeNet is the only or best choice of dataset for pre-training, without questioning it too much.

Because of these gaps we have identified in the state of the art, it becomes interesting to study Siamese networks with a wide variety of encoders, not just transformers. And pre-training on other types of datasets, such as SimpleShape. To explore new ways to help neural networks on 3D points to get better results in different target tasks.

# Chapter 3

## Methodology

The purpose of this chapter is to present the methodology that guided the upcoming experiments. To this end, it first identifies the problems (3.1) and research questions (3.2) to be addressed. It articulates our hypothesis regarding the questions posed (3.3) and formalizes the goals of the research (3.4). Finally, the last section stipulates the scope and assumptions of this research (3.5)

### 3.1. Problem statement

The development of autonomous devices using surface sensors, such as LiDAR sensors, requires machine learning models to maximize their accuracy, precision, and reliability. This type of sensor works with data structures based on three-dimensional point clouds, for which there is not a large variety of labeled datasets that allow training large and complex data models with traditional machine learning techniques.

Self-supervised learning is currently the main alternative for pre-training neural networks without labeled data, which reduces the dependence on large amounts of labeled data. However, state-of-the-art proposals in 3D SSL are mainly composed of autoencoders, complex structures that allow pre-training neural networks for 3D models. These are composed of transformer networks, which are also known to require large amounts of samples for training.

An essential feature sought in self-supervised learning models is that they generalize to different models of subsequent tasks, i.e. that they are useful for pre-training models aimed at different tasks. For this, it is essential that the backbone network, after pre-training, can extract features that facilitate the work of the new network head.

Among the most recent SSL proposals are Siamese networks, with architectures much simpler than autoencoders, which have achieved excellent results. However, they have been mainly applied to 2D models, so it is easy to consider them as pre-training model alternatives for 3D models working with point clouds.

Siamese networks have a special case that must be avoided to perform proper training of

the networks: the collapse of the network to trivial solutions. For this purpose, the contrastive proposals use negative pairs, and the non-contrastive models use several alternatives, such as BYOL, which requires a momentum encoder, and SIMSIAM, which uses gradient stopping during training. Therefore, if Siamese network architectures are to be considered as an alternative, it is essential to verify that the network does not collapse during its pre-training.

The process of transforming an architecture that works with 2D models into a new structure that works with 3D models is mainly based on changing the encoder, since it is the module that receives the input. ResNet [83] is the main backbone used in 2D structures, and PointNet [45] is the main backbone used in 3D. Unfortunately, PointNet is not a generalization of ResNet; it has a significantly different structure.

Since PointNet, the most widely used encoder network, is invariant to rigid transformations and rotations of the entire point cloud, one of our main challenges was to generate a set of transformations that allow for extended data differentiation. To address this challenge, we propose the creation of a new module  $T$  that performs data augmentation at the beginning of the training cycle. This module includes transformations that can be applied to point clouds to enable accurate distortion of the data, thus facilitating encoder learning during training.

Building self-supervised models that work with point clouds is a challenging problem with direct applications in autonomous devices. The main goal is to eliminate the bottleneck of the need for more training data. This work aims to consider the latest advances in 2D and 3D learning to generate a new self-supervised learning model that extends the capabilities of neural networks when performing tasks on 3D point clouds.

Another approach developed for self-supervised learning is formula-driven supervised learning. This method relies on the generation of synthetic datasets that, through general SSL techniques, allow the pre-training of encoders with a large amount of data generated from mathematical formulas. In addition to the advantages of controlling the shapes of the data used for pre-training, it also has the advantage of being free of copyright issues, thus encouraging the development of new copyright-free techniques.

The discovery of datasets created from mathematical formulations that allow neural networks to be pre-trained, with results comparable to or better than commonly used datasets such as ShapeNet [10], remains a state-of-the-art challenge. If successful, it would greatly benefit research on self-supervised learning techniques on 3D representations.

The concept of using synthetic datasets as a basis for self-supervised learning has been explored recently, with a focus on neural networks for image processing. However, the point cloud domain has seen only marginal benefits from this technique. The main problem has been the lack of synthetic 3D point cloud datasets capable of generating feature-rich samples that can effectively support neural networks during pre-training.

## 3.2. Research questions

The main research questions are: How can self-supervised learning be extended to generate representations that facilitate the pre-training of 3D point cloud classification and segmentation networks, and what are the limitations in knowledge generation due to point cloud transformations? These five research questions are considered together:

1. Can Siamese networks, when used in a self-supervised learning context, serve as an effective approach for pre-training encoders on 3D point cloud data to generate useful data representations?
2. What strategies are most successful in preventing Siamese networks from converging to trivial solutions during the self-supervised pre-training phase?
3. Which data augmentation techniques prove to be the most beneficial in enhancing the self-supervised pre-training of Siamese networks for 3D point cloud processing?
4. In terms of developing feature representations suitable for classification and segmentation tasks with point cloud data, how do Siamese networks compare to other state-of-the-art methods in terms of efficiency and accuracy?
5. How effective is pre-training of self-supervised learning networks on mathematically generated datasets, and how does this approach compare to traditional pre-training datasets?

These revised questions are structured to facilitate a focused investigation, ensuring a comprehensive exploration of the potential and challenges of Siamese networks in the context of self-supervised learning for 3D point cloud data.

## 3.3. Hypothesis

We hypothesize that the 3D self-supervised learning Siamese network technique, when synergized with carefully selected backbone networks and data augmentation techniques, will significantly improve the performance of downstream point cloud tasks, including classification and segmentation. It is also hypothesized that the use of formula-driven datasets for pre-training neural networks on 3D point clouds has the potential to equal or exceed the results achieved by traditional pre-training on synthetic datasets, thus providing a promising path to superior representation quality.

## 3.4. Goals

1. To identify a set of point cloud encoders capable of generating point cloud characterization representations that can be used in classification and segmentation tasks.

2. To identify a set of 3D transformations  $T$  that allow different point cloud encoders to learn to represent figures under the strategy of Siamese networks. These transformations should take into account the increased degrees of freedom provided by the third dimension.
3. To evaluate the ability to improve the pre-training of 3D point cloud encoders through different Siamese network-based pre-training strategies. The previously identified encoders and  $T$  transformations will be compared to the state of the art using established benchmark datasets such as ModelNet and ScaObjectNN.
4. To investigate the generalization ability of neural networks developed from pre-training with the ShapeNet database in subsequent deep learning tasks.
5. To perform an ablation study of the 3D self-supervised learning architecture to selectively and strategically improve its components.
6. To evaluate the ability of the SimpleShape dataset to generate high information content figures that support self-supervised learning strategies in characterizing clouds in such a way that downstream tasks benefit from this pre-training.
7. Provide an implementation that can be easily integrated with other 3D deep learning frameworks.

### 3.5. Scope and assumptions

This research evaluates the effectiveness of Siamese networks, an architecture for pre-training 2D neural networks, on 3D point clouds self-supervised learning. It employs standard benchmarks to assess the quality of pre-training performed by different architectures and to compare them with the latest state-of-the-art work, which comprises different pre-training models and architectures.

Additionally, the potential of SimpleShape as a pre-training dataset for self-supervised learning on 3D point clouds will be evaluated. To this end, state-of-the-art models and architectures in SSL will be tested, with only the pre-training dataset varying. These pre-trainings will be evaluated using the same standard benchmarks as the Siamese network.

While some of the experiments focused on standard benchmarks may not fully reflect real-world scenarios, as they utilize datasets generated from CAD models, the shape and structure learning obtained through pre-training can be directly extrapolated to real 3D point clouds. This will enable future work to evaluate the same models in terms of their performance on datasets such as object recognition in autonomous vehicle programs.

The following assumptions are made for the realization of these experiments

- Data Accessibility Assumption: It is assumed that there is an increasing availability of

3D point cloud data that can be used to train neural networks, although obtaining large amounts of labeled data remains a challenge.

- Siamese network effectiveness: It is assumed that training techniques based on Siamese networks, such as BYOL or SIMSIAM, applied to neural networks for 3D point cloud processing are effective in improving network performance in subsequent tasks without relying on labels.
- Self-Supervised Learning Generalization: It is assumed that self-supervised learning methods previously developed and tested on 2D images will be equally effective when applied to 3D data, especially in classification, segmentation, and object recognition tasks.
- Synthetic dataset creation: It is assumed that the use of a synthetic dataset, such as SimpleShape, for pre-training could be as effective as real, conventional datasets, providing a viable and scalable way to improve the learning of representations without relying on the acquisition of new labeled data.
- Evaluation metrics: The thesis assumes that established evaluation metrics, such as classification accuracy and mIoU for segmentation, are suitable for measuring the success of representations learned through self-supervised learning techniques.

# Chapter 4

## Proposal

Our proposal is divided into two parts: On the one hand, to evaluate the capacity of Siamese networks to pre-train encoders focused on deep learning on 3D point clouds. On the other hand, to evaluate the potential of SimpleShape to be used as a pre-training dataset for the same type of networks. The ultimate goal of pre-training in both cases is that the network learns to recognize shapes without the need for labels, thus improving its performance in real tasks, so-called downstream tasks.

In this chapter, we describe both approaches. First, we describe the PointSIMSIAM and PointBYOL models, which allow us to evaluate the capacity of Siamese networks in self-supervised learning (4.1). And second, it describes the creation of SimpleShape, the dataset to be used for pre-training self-supervised learning under the idea of formula-driven supervised learning (4.2).

### 4.1. Siamese networks proposed models

The proposed model is inspired by the SIMSIAM [28] and BYOL [27] frameworks, hereafter referred to as PointSIMSIAM and PointBYOL, respectively. Its structure is analogous to that used in 2D image processing, with adaptations mainly in the encoder used and in the  $T$  transformations applied to the point cloud before input to the Siamese networks.

#### 4.1.1. Data augmentation

In the transition from 2D image processing to 3D point cloud analysis, certain transformations used in the original domain of Siamese networks require adaptation, while others may be less relevant or even inapplicable. However, this shift also introduces new transformation possibilities for data augmentation in neural network pre-training.

Generalizable transformations include translation, flipping, cropping, rotation, scaling, and noise, all reinterpreted for 3D points  $\mathcal{P} = \{x, y, z\}$  instead of 2D pixels. Although these



transformations remain essentially the same, neural networks processing 3D data must adapt to be invariant to such changes. This adaptation could be achieved by augmenting the data during training, by using symmetric functions such as max-pooling, or by using innovative techniques that promote network invariance to these transformations.

When adapting 2D image processing techniques to 3D point cloud analysis, several transformations commonly used in 2D contexts become less relevant or inappropriate. Color-based modifications, including color jitter, grayscale, and adjustments to brightness, contrast, saturation, and hue, are less applicable to 3D point cloud processing due to the typically color-agnostic nature of point clouds. Similarly, enhancements such as blur, which are integral to 2D image enhancement, lose their utility in the 3D domain.

Morphological operations, such as erosion and dilation, which are commonly employed in two-dimensional image processing [109], are not directly applicable to three-dimensional point clouds due to the fundamental differences in their structural and representational characteristics. Furthermore, two-dimensional operations, such as rotation and flipping, scaling and cropping, perspective distortion, and filters, such as Gaussian blur, require adaptations for their application from two-dimensional images to three-dimensional point clouds. These discrepancies necessitate a reevaluation of the most appropriate transformations for effective 3D point cloud processing.

Furthermore, new transformations that are unique to the 3D domain are introduced. These include point cloud sampling, which adjusts the density of point clouds by randomly adding or removing points. Additionally, random point cloud deletion or masking is included, which randomly eliminates sections of a point cloud to force the model to learn from partially incomplete data. In addition, 3D shearing, a transformation that distorts the shape of the point cloud in 3D space by changing the angles between points, is a technique that has no direct equivalent in 2D imaging.

These new transformations are critical to increasing the robustness and versatility of models designed for 3D object recognition and analysis. Huang et al. [98] distinguish between temporal and spatial transformations. The authors propose a series of transformations, including random rotation, random translation, and random scaling, which are designed to simulate the changes that occur between two consecutive frames in a sequence of 3D model captures from a sensor. In contrast, their proposed spatial transformations include random cropping, random cropping, random jittering, and random drop-out.

In this study, a sequence of transformations is proposed that does not focus on simulating data sequences to justify similarity search in the Siamese network. Instead, the objective is to demonstrate that two point clouds derived from the same figure, albeit with different transformations applied, should be recognized as the same object by an encoder, thus yielding similar representations.

This transformation sequence consists of random scaling, random rotation, random translation, point cloud jitter, an additional layer of scaling and translation, and finally, inspired

by the masked autoencoder pipeline and similar to Point-MA2E [90], a random mask operation that eliminates 60% of the total points was applied to the transformed point cloud. The complete transformation pipeline is shown in Figure 4.1, and a specific example using the ModelNet [11] dataset is shown in Figure 4.2.

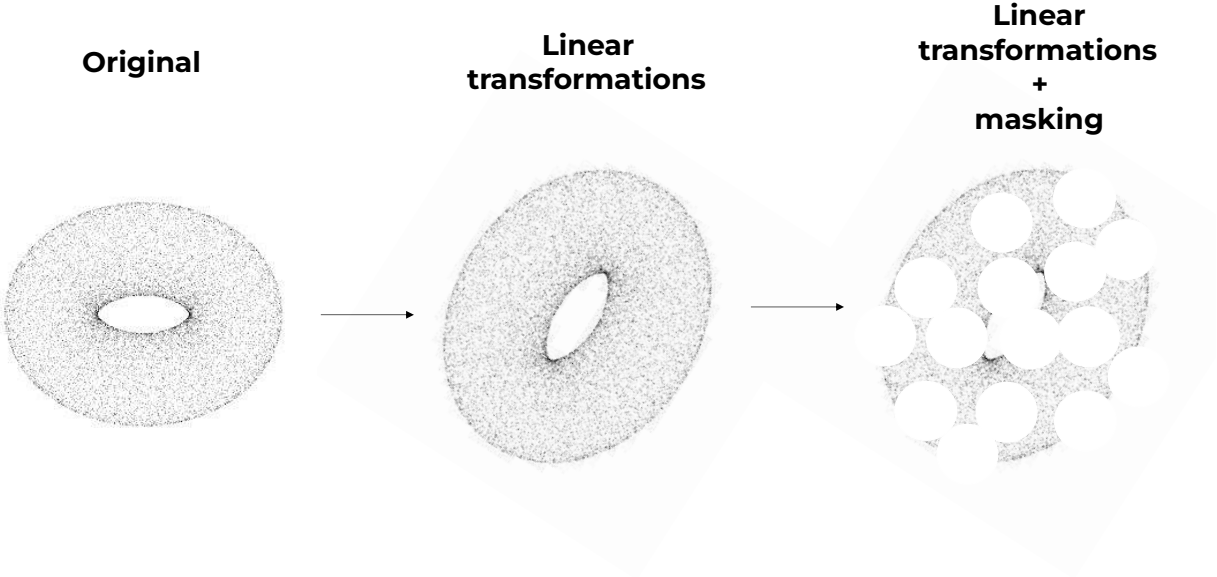


Figure 4.1: Linear transformation and masking pipeline illustration.

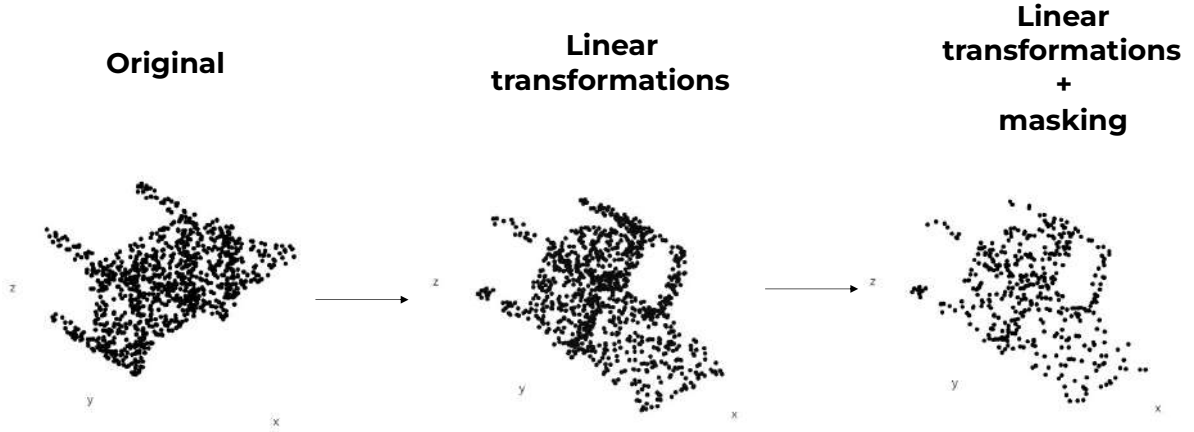


Figure 4.2: Linear transformation and masking on ModelNet [11] chair sample. Masking ratio is setted to 60% of the total points in the model.

The idea of applying random masking to the point cloud is considered as one of the principal contributions to the study of Siamese networks on 3D point clouds in this research. This transformation follows the structure of masked autoencoders but with natural differences due to the structure of the models. Since point-wise networks operate directly on the 3D coordinates of points, this transformation involves removing sections of the point clouds in the 3D domain, differing from the work of autoencoders that use transformer networks as encoders and apply the random mask operation to the tokens generated from the point cloud, masking tokens, that represents a group of points, rather than points.

Although this transformation is applied in a different layer of the training pipeline, it follows the same principle as the other transformations. In essence, a uniformly masked point cloud should be correctly identified by the encoder as a pair of another masked version of the same object. This assertion is exemplified in cases such as a table missing a leg, which remains essentially the same original table, or an airplane losing a wing, or a door with holes.

While basic transformations involving masking on point clouds, such as the clipping proposed by Huang et al. [98], are capable of uniformly removing a portion of the cloud, masking in masked autoencoders operates uniformly across the cloud, removing multiple tokens that represents groups of points, simultaneously, with a uniform masking density throughout the image.

Pang et al. [23] demonstrated that masking between 60% and 80% of the tokens generated for a point cloud is the optimal range for pre-training a Transformer neural network. This implies that for an input of 1024 points, between 204 and 410 points are sufficient for an

encoder to recognize and correctly characterize the type of figure. This is due to the fact that the semantic value of the points is not uniform. The corners and curves of figures contain a significantly greater amount of information than intermediate points, which allows for the accurate characterization of point clouds based solely on the most significant points.

The masking of masked autoencoders works with the 32 farthest-point-sample points of the cloud and their 32  $k$  nearest neighbors, resulting in a situation where most of the removed points have little semantic value. Since the masking is done uniformly, the majority of the removed points are likely to be in the center of the figure, leaving the critical points that define an object intact.

In the case of using transformer-type backbone networks, in addition to applying the 3D spatial masking transformation. The application of the token masking algorithm used by Point-MAE [23] was also explored in its pre-training to determine which of the two transformations is more beneficial for such networks when pre-trained with the methodology of Siamese networks.

These transformations were applied to the PointSIMSIAM and PointBYOL networks in conjunction with various encoders. These encoders may benefit more or less from the applied transformations and the final masking, providing insights into the effectiveness of these methods in improving the learning capabilities of the networks.

#### 4.1.2. Encoders

In addition to experimenting with the two networks, PointSIMSIAM and PointBYOL, and applying the previously detailed transformations, a subset of neural networks was also selected to serve as the backbone of the structure. These were of different types: PointNet [45] as a pointwise network, DGCNN [48] as a graph-based network, a Transformer [55] backbone as proposed by Point-BERT [86], and Point-MLP [47], which is also pointwise but stands out as one of the models with the best results in the state of the art. Point-MLP is particularly noteworthy because it avoids specialized encoders and consists entirely of MLP-type networks.

Our interest is not only in examining the encoders for their ultimate capability in classification and segmentation tasks, but also in the specific ability of Siamese network methods to teach each type of encoder to produce higher quality representations of 3D point clouds. Additionally, following the issues raised by Ma et al. [47] regarding the specialization of large networks to extract specific features from point clouds, the number of parameters in the networks used in downstream tasks and the inference time for each were also compared. This approach aims to provide a comprehensive evaluation of the effectiveness and efficiency of these different coding strategies in the context of Siamese network-based learning.

### 4.1.3. Prediction head networks

In describing the architecture of Multilayer Perceptron (MLP) networks, the notation  $(m,n,p)$  is employed. Here,  $m$  denotes the number of neurons in the input layer.  $n$  indicates the number of neurons in the hidden layer. Should there be multiple hidden layers, each subsequent number following  $m$  and preceding  $p$  represents the number of neurons in each additional hidden layer. Lastly,  $p$  corresponds to the number of neurons in the output layer.

The proposed models contain networks that serve as prediction heads. These were composed of four MLP layers with the dimensions  $(d_{in}, 256, 256, d_{out})$ , where the input was the output vector  $v_{in} \in \mathbb{R}^{d_{in}}$  from the pre-trained encoder. The output of these layers is the classification vector  $v_{out} \in \mathbb{R}^{d_{out}}$  for the following task. This four-layer structure is based on the design proposed in Point-MAE [23], where a four-layer head is also used for the final predictions. For segmentation tasks, the original segmentation versions of each model were implemented, which are typically an extension of the original model, such as PointNet [45], as shown in Figure 2.1.

This approach aims to ensure that the prediction head is adequately tuned to the specific characteristics of the encoded representations, allowing for effective classification or segmentation. The use of a standardized MLP structure across different encoder types also facilitates direct comparison of their performance in the context of the same downstream task.

#### 4.1.4. General proposed model

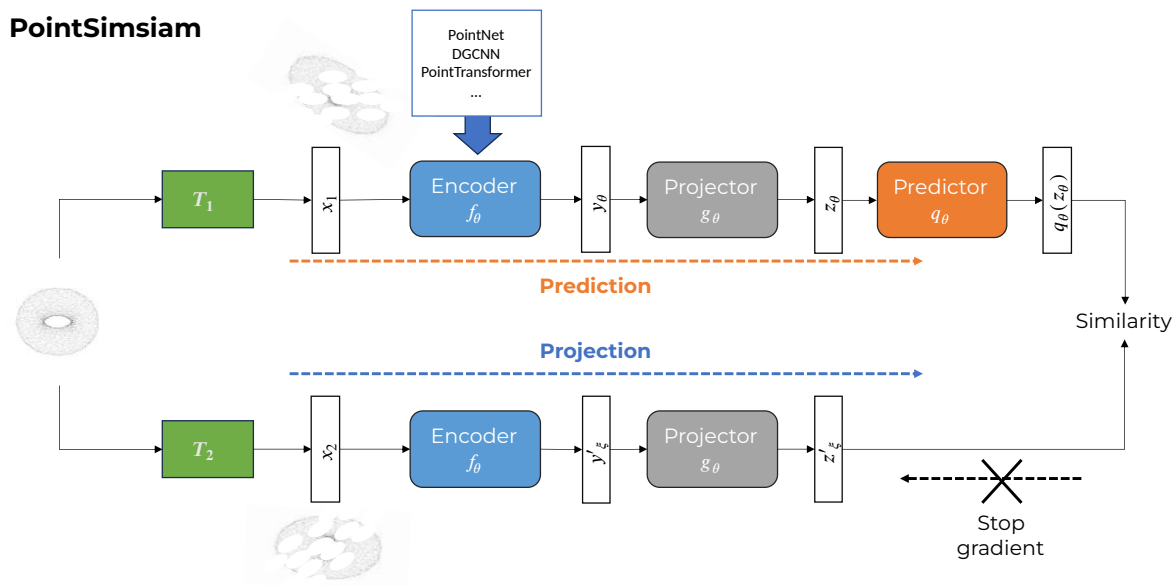


Figure 4.3: **PointSIMSIAM architecture.** It is the same architecture than SIMSIAM [28] for 2D images, but it applies 3D linear transformation and masking to the input. It can be built to pre-train any 3D encoder.

In summary, the use of Siamese networks with encoders that directly process 3D point clouds is proposed. The models are PointSIMSIAM, shown in Figure 4.3, and PointBYOL, a version of BYOL for 3D point clouds. PointNet, DGCNN, Transformer, and Point-MLP encoders were pre-trained using a series of linear transformations and point masking. These combinations were evaluated for their ability to improve results in downstream tasks and for their effectiveness in teaching the encoders to generate high-quality representations.

It is expected that a thorough analysis of the various proposed combinations of transformations, encoders, and pre-training methods will not only identify the most effective combination for downstream tasks, but also provide insight into which backbone networks are suitable and efficient for self-supervised learning using Siamese networks. In addition, this study aims to understand the impact of different 3D transformations on network learning and their potential to prevent the collapse of Siamese networks during pre-training. This research could significantly contribute to the field of 3D point cloud processing by providing new perspectives and techniques for effective and efficient learning.

## 4.2. SimpleShape for formula-driven supervised learning

To evaluate the effectiveness of SimpleShape [29] as a pre-training dataset for self-supervised learning (SSL) on 3D point clouds, Point-MAE [23] was pre-trained using its autoencoder-based approach, replacing the standard ShapeNet55 [10] with SimpleShape. For a fair comparison, the official implementation of Point-MAE<sup>1</sup> was used to pre-train and evaluate this model under the same conditions as the other experiments.

To evaluate SimpleShape, systematic variations of the dataset were created with different numbers of figures, named according to the number of objects created. The dataset variants for evaluation included sets of 1K, 5K, 10K, 15K, 50K, and 100K figures.

It is expected that the versions with fewer figures (1K and 5K) would yield lower results compared to the other variants, since Transformer networks [55] require a large amount of data for training, and these sets have less than 10% of the figures provided by ShapeNet55 [10]. In addition, benchmark performance is expected to improve proportionally as the number of figures in the dataset increases.

### 4.2.1. Dataset creation

To create the different datasets, the same configuration as the original benchmark was followed, only varying the number of samples generated. The methodology of Sipiran et al. [29] was used to create the datasets, keeping only the generated point clouds and omitting their symmetry planes. Examples of the generated dataset, as shown in Figure 4.4, are presented below.

---

<sup>1</sup> <https://github.com/Pang-Yatian/Point-MAE>

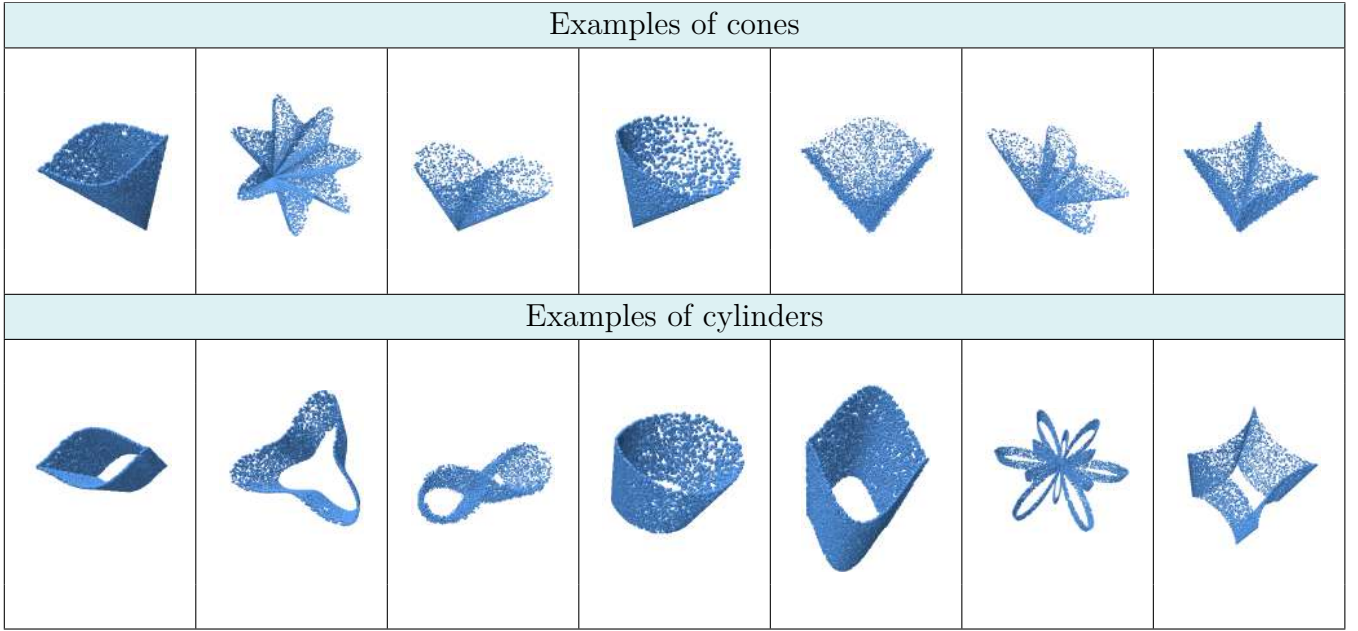


Figure 4.4: A visual illustration of point clouds in SimpleShape datasets.

The generation of point clouds is done in three steps:

1. **Selection of a Known Symmetry Curve Family:** A curve is selected from a family of curves with known symmetry. The basic curve families are detailed in the Table 4.1.
2. **Generation of a Cylinder or Cone:** A cylinder or cone is generated using the selected curve as the directrix.
3. **Application of Translations and/or Rotations:** Finally, translations and/or rotations are applied to the generated shape.




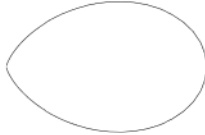
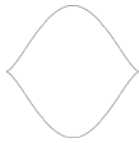

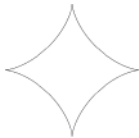
Point clouds are created by randomly selecting one of these curves and assigning random values to the parameters  $a$ ,  $b$ , and  $m$  within certain constraints. The parameters for each curve are assigned as follows:

- Citrus:  $a = 1$  and  $b$  randomly chosen in the interval  $[1, 13]$ ,  $b \in \mathbb{N}$ .
- m-Convexities:  $a$  randomly chosen in the interval  $[0.5, 1.1]$ ,  $a \in \mathbb{R}$ ,  $b$  randomly chosen in the interval  $[0.2, 0.9]$ ,  $b \in \mathbb{R}$ , and  $m$  randomly chosen in the interval  $[3, 9]$ ,  $m \in \mathbb{N}$ .
- Geometric petal:  $a$  randomly chosen in the interval  $[1.0, 2.0]$ ,  $a \in \mathbb{R}$ ,  $b$  randomly chosen in the interval  $[1, 6]$ ,  $b \in \mathbb{N}$ , and  $m$  randomly chosen in the interval  $[1, 6]$ ,  $m \in \mathbb{N}$ .
- Lemniscate of Bernoulli:  $a = 1$ .
- Egg of Keplero:  $a = 1$ .



- Mouth curve:  $a = 1$ .
- Astroid:  $a = 1$ .


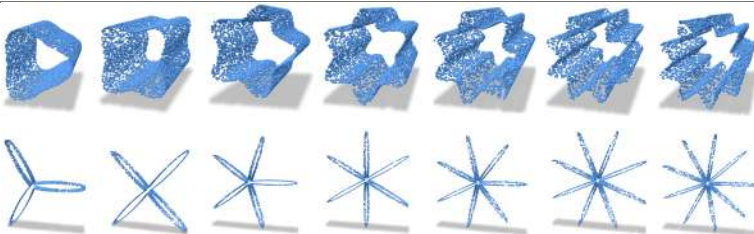
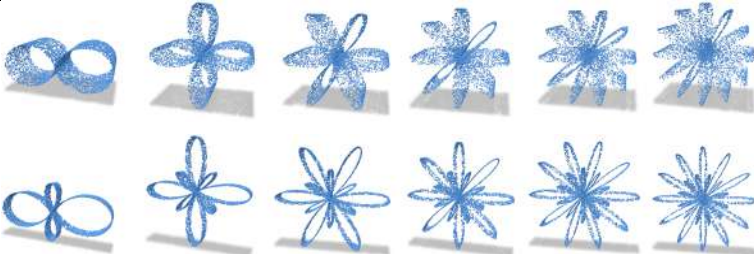
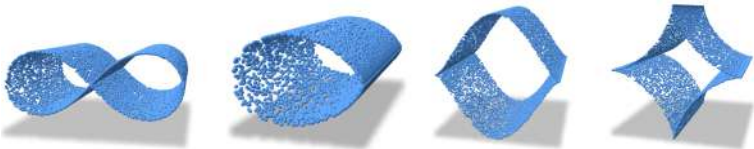
Table 4.1: Families of plane curves used as directrix of the cylinders and cones in SimpleShape datasets.

Family	Example	Formula
Citrus curve		$\mathbf{P}(t) := \begin{cases} x = t - \frac{a}{2} \\ y = \pm \sqrt{\frac{(a-t)^3 t^3}{a^4 b^2}} \end{cases}$
m-convexities		$\mathbf{P}(t) := \begin{cases} x = \frac{a}{1+b \cos(mt)} \cos t \\ y = \frac{a}{1+b \cos(mt)} \sin t \end{cases}$
Lemniscate of Bernoulli		$\mathbf{P}(t) := \begin{cases} x = a \frac{\sin t}{1+\cos^2 t} \\ y = a \frac{\sin t \cos t}{1+\cos^2 t} \end{cases}$
Egg of Kepler		$\mathbf{P}(t) := \begin{cases} x = \frac{a}{(1+t^2)^2} \\ y = \frac{at}{(1+t^2)^2} \end{cases}$
Mouth curve		$\mathbf{P}(t) := \begin{cases} x = a \cos t \\ y = a \sin^3 t \end{cases}$
Geometric petal		$\mathbf{P}(t) := \begin{cases} x = (a + b \cos nt) \cos t \\ y = (a + b \cos nt) \sin t \end{cases}$
Astroid		$\mathbf{P}(t) := \begin{cases} x = a \cos^3 t \\ y = a \sin^3 t \end{cases}$

To generate the point clouds, cones or cylinders are created with a rotation axis randomly aligned with the z-axis. Translation and rotation transformations are applied with a probability of 0.8 each. The Table 4.2 shows examples of shapes generated by SimpleShape. Then, each point cloud is randomly subjected to final transformations. These are classified into 5 types according to [29], including uniform noise, Gaussian noise, undersampling, and combinations of these three transformations.

Finally, to create the datasets used in this research, these guidelines are followed to generate point cloud sets of 1K, 5K, 10K, 15K, 50K, and 100K samples. Although originally designed to create figures with known symmetries, versions of this dataset focused on increasing the variability of local and global features could be developed for self-supervised learning techniques.

Table 4.2: Examples of shapes generated by SimpleShape [29].

<p style="text-align: center;"><b>Citrus</b>  <math>a = 1</math>  <math>b \in [1, 13], b \in \mathbb{N}</math></p>	
<p style="text-align: center;"><b>m-Convexities</b>  Top: <math>a = 0.5, b = 0.2</math>  Bottom: <math>a = 1.1, b = 0.9</math>  <math>m \in [3, 9], m \in \mathbb{N}</math></p>	
<p style="text-align: center;"><b>Geometric petal</b>  Top: <math>a = 1.0, b = 1</math>  Bottom: <math>a = 2.0, b = 6</math>  <math>m \in [1, 6], m \in \mathbb{N}</math></p>	
<p style="text-align: center;"><b>Lemniscate Bernoulli</b>  <b>Egg of Keplero</b>  <b>Mouth curve</b>  <b>Astroid</b></p>	

# Chapter 5

## Datasets

The datasets used in the experiments are described below. Following standard benchmarks, these are Shapenet[10], ShapeNetPart[61], ModelNet[11], and ScanObjectNN[35]. It also reveals the structure of SimpleShape[29], which was used for the second stage of our experiments.

### 5.1. ShapeNet

ShapeNet [10] is a comprehensive repository of 3D CAD models that includes a diverse set of semantic categories structured according to the WordNet [110] taxonomy. This database is notable not only for its collection of models, but also for its extensive annotations, which provide semantic insights critical to 3D geometric understanding and shape analysis. With over three million shapes, approximately 220,000 of which are meticulously classified into over three thousand categories, ShapeNet is recognized as one of the most comprehensive and diverse datasets available.

Designed to evolve, ShapeNet regularly integrates new models and community-contributed annotations. Models are drawn from public online repositories as well as established research datasets, providing a wide range of shapes from common object categories to complex scenes.

The ShapeNet initiative includes subsets such as ShapeNetCore, consisting of 51,300 unique 3D models with manually verified category and orientation annotations, and ShapeNet-Sem, consisting of 12,000 models with 270 categories, which provides a denser subset of models.

Furthermore, ShapeNet facilitates user interaction with its data through tools and a web API <sup>2</sup> that allows researchers to perform specific queries and batch downloads through an easy-to-use web interface.

One of the challenges in building ShapeNet was the methodology for capturing and val-

---

<sup>2</sup> <https://shapenet.org>

idating annotations with high accuracy. When full verification isn't possible, a confidence metric is provided for each annotation, increasing the reliability of the dataset for various applications.

Annotations in ShapeNet range from linguistic attributes based on WordNet to geometric details such as rigid alignments, parts, keypoints, and symmetries. It also includes functional and physical annotations such as surface material properties and weight, which are critical for applications involving physical simulation and structural analysis.

Yu, Xumin, et al. [111] present ShapeNet-55, an advanced benchmark derived from the ShapeNetCore dataset. This benchmark uses all 55 object categories from ShapeNet, with the goal of testing the capabilities of neural networks with a dataset that maximizes diversity. They partition the original ShapeNet dataset, allocating 80% of the objects from each category to the training set and reserving the remaining 20% for evaluation purposes. This allocation results in a substantial collection of 41,952 models for training and 10,518 models for testing. For each object in the dataset, 8,192 points are randomly sampled from its surface to construct the corresponding point cloud, providing a rich and diverse set of data points for robust neural network training and evaluation.

This particular subset has been widely adopted as the standard for pre-training datasets used by self-supervised learning frameworks for point cloud models [23, 86]. For pre-training, 1024 of the available 8192 points are used for each object.

This specific subset has increasingly become the default standard for pre-training datasets used in self-supervised learning frameworks, especially for point cloud models, as evidenced by its adoption in studies such as Point-BERT [86] and Point-MAE [23]. For the pre-training phase, these frameworks typically use a subset of 1,024 points from the available 8,192 points for each object, providing a representative yet computationally manageable sample of the entire point cloud for each model in the dataset. This approach balances the need for detailed representation of each object with the practical constraints of processing and learning from large point cloud data.

## 5.2. ShapeNetPart

ShapeNetPart [61] represents an advanced segment within the ShapeNetCore [10] database, specifically designed to facilitate in-depth analysis of 3D shapes by providing meticulous per-point annotations. The dataset meticulously catalogs over 31,963 models, meticulously dissecting them into 16 basic shape categories, annotated with 50 parts in total. This granular approach to annotation is critical for tasks requiring precision, such as semantic segmentation and object recognition, where understanding the specific parts and features of an object is essential.

The creation of ShapeNetPart is based on an active learning methodology, a semi-automated process that significantly reduces the labor-intensive nature of manual annotation. First, hu-

man annotators label a subset of points on the 3D models, focusing on specific areas or features of interest, such as the arms of a chair or the wheels of a car. These manual annotations serve as seeds for an automated propagation system that extrapolates the initial labels to similar shapes within the dataset, effectively augmenting the initial human effort.

To ensure the fidelity of the propagated labels, a subsequent verification phase is implemented where both human-generated and algorithmically propagated annotations are meticulously reviewed. This verification step is not only a quality control measure; it also provides critical data that is used to fine-tune the propagation algorithms, increasing their precision and reliability.

In practice, this framework processes each shape by uniformly sampling thousands of points across its surface, assigning each point a boolean label that confirms or denies its association with the specified region. Once labeled, these points serve as proxies for the shape’s surface, and their annotations are eventually transferred to the actual faces of the model, ensuring that the dataset reflects the complex topography of each object.

### 5.3. ModelNet

ModelNet [11] is a significant dataset designed to support the training of deep 3D shape representations capable of capturing intra-class variance among objects. It was constructed by aggregating a large number of 3D CAD models from various online resources, including the 3D Warehouse <sup>3</sup> and Yobi3D <sup>4</sup>, which indexes several CAD model websites. The dataset was carefully curated to include a wide range of common object categories from the SUN database [112], each with a minimum of 20 instances, for a total of 660 different categories. Additional models were obtained from the Princeton Shape Benchmark [113].

To ensure the quality and relevance of the dataset, downloaded models underwent a rigorous review process. Incorrectly categorized models were filtered out using Amazon Mechanical Turk <sup>5</sup>, where workers reviewed thumbnails of the models to confirm category accuracy. The authors also performed manual inspections to remove any irrelevant objects or misclassifications from each CAD model, ensuring that each mesh represented a single object belonging to its labeled category. Superfluous elements such as ground thumbnails or bystanders were also excluded.

The final dataset is extensive, containing 151,128 3D CAD models across 660 different object categories, which significantly exceeds the scope of previous datasets in this area. Specifically, a subset known as ModelNet40 was created for classification tasks. This subset contains 12,311 CAD models in 40 different categories, further divided into 9,842 training samples and 2,468 test samples. This division not only provides a robust framework for train-

---

<sup>3</sup> <https://3dwarehouse.sketchup.com>

<sup>4</sup> <http://yobi3d.com>

<sup>5</sup> <https://www.mturk.com>

ing and evaluating classification algorithms, but also ensures a diverse and comprehensive set of models, making ModelNet40 an important resource in the field of 3D object classification.

## 5.4. ScanObjectNN

ScanObjectNN [35] is a real-world point cloud object dataset derived from indoor scene scans, specifically designed to challenge current point cloud classification methods. It was created from two notable scene mesh datasets, SceneNN [114] and ScanNet [12], from which objects were manually selected and categorized into 15 common groups. This selection process involved refining the initial instance segmentation from the scene datasets to ensure accuracy and consistency across the collection.

The dataset is notable for its complexity, reflecting the clutter and partial views often found in real-world environments due to occlusion. It contains 2,902 objects across its categories, with each object represented by a list of points that include both global and local coordinates, normals, color attributes, and semantic labels. The objects in ScanObjectNN reflect a more challenging and realistic environment than their synthetic counterparts, as they are often noisy and incomplete, providing a more rigorous test for classification algorithms.

To further enhance the usefulness of the dataset, several variants were created to represent different levels of difficulty and to test the robustness of the classification methods against more extreme real-world conditions. These variants are

- **OBJ ONLY:** This baseline variant contains only the segmented objects, free of additional noise or background, allowing a direct comparison with the performance of the synthetic dataset.
- **OBJ-BG:** Objects include background data, simulating common real-world scenarios where objects may not be isolated. This variant helps to test how well classification methods can distinguish between the object of interest and its surroundings.
- **Perturbed Variants (PB):** By introducing translations, rotations, and scaling perturbations to the ground truth bounding boxes, these variants create objects with varying degrees of background inclusion and bias, presenting up to 14,510 perturbed objects to further challenge classification methods. Four variants are distinguished according to their level of difficulty: PB\_T25, PB\_T25\_R, PB\_T50\_R, and PB\_T50\_RS. Where T25 and T50 refer to translations between 25% and 50% of their size. R and S refer to rotation and scaling respectively.

The ScanObjectNN dataset provides an invaluable resource for advancing point cloud classification techniques and is publicly available to the research community, encouraging the development and benchmarking of new methods that can handle the intricacies of real-world data.

## 5.5. SimpleShape

SimpleShape [29] is a data set specifically designed for analyzing and detecting symmetry in 3D point clouds. The dataset consists of a set of closed plane curves that serve as the basis for generating surfaces of simple geometric shapes, such as cylinders and cones. These shapes are designed to test the algorithms' ability to detect reflective planes and the number of symmetries within a given shape.

The dataset consists of 69,000 simple three-dimensional shapes represented as point clouds, divided into training and test sets of 60,000 and 9,000 point clouds, respectively. Each point cloud is generated by selecting a closed plane curve from a predefined family of curves characterized by their unique shapes and parametric representations. This curve, which lies in the XY plane, is then extruded along the Z axis to form the desired 3D shape, either cylindrical or conical.

The point clouds in SimpleShape are subjected to various perturbations to mimic real-world conditions and test the robustness of the symmetry detection algorithms. These perturbations include clean point clouds with no noise, point clouds with uniform noise of varying intensities, point clouds with Gaussian noise, undersampled point clouds, and combinations of noise and undersampling.

The benchmark evaluates algorithms based on how effectively they can handle these perturbations and still accurately detect symmetries. The goal is to provide a comprehensive dataset that can be used to develop and test algorithms on simple shapes, pushing the boundaries of what is possible in symmetry detection in point cloud data. In addition to serving as a tool for advancing the field of 3D shape analysis, the dataset provides a robust platform for researchers to benchmark and validate their symmetry detection methods.

# Chapter 6

## Evaluation metrics

The metrics used to evaluate the quality of the pre-training resulting from the application of both proposals of this research are described below: Siamese networks as a pre-training method and replacing ShapeNet55 with SimpleShape as the pre-training dataset in self-supervised learning models. These metrics include linear probing (6.1), fine-tuning for classification (6.2), few-shot learning (6.3), and segmentation (6.4). Additionally, qualitative metrics (6.5) and retrieval metrics (6.6) are added.

### 6.1. Linear probing

Linear probing serves as a central evaluation technique in the field of self-supervised learning, particularly when investigating the quality of representations derived from unsupervised pre-training phases. It is based on the premise that a well-trained self-supervised model should have learned representations that capture the underlying structure of the dataset, even without the guidance of explicit labels.

The essence of linear probing is simple yet profound: a simple linear classifier (e.g., SVM or KNN classifier) is trained on the frozen representations obtained from a self-supervised model. The model itself is kept static, its weights untouched during this phase, ensuring that the classifier's performance depends solely on the quality of the embeddings it receives.

When using linear probing, the workflow begins once a self-supervised model has been trained. This model, having digested unlabeled data, now acts as a feature extractor. As new data is passed through the model, it outputs embeddings (a distilled essence of the input data as seen through the lens of the model's learned parameters).

These embeddings are then used as input to the linear classifier. The classifier's task is to map the embeddings to the correct output labels, even though these labels were never part of the model's initial training. Because of its simplicity, the classifier is a litmus test for the embeddings: If the representations are strong, even this simple classifier should be able to achieve high accuracy.



The evaluation of the classifier, typically using a labeled validation set, provides direct feedback on the representational power of the pre-trained model. The achieved accuracy becomes a measure of the quality of the embeddings and the effectiveness of the self-supervised learning approach. This metric is particularly important for comparing different self-trained algorithms or for fine-tuning their hyperparameters.

Linear probing thus plays a critical role in the self-supervised learning evaluation ecosystem, providing a clear, quantitative metric that correlates the intricacies of unsupervised representation learning with tangible downstream performance. It bridges the gap between the abstract representations of unsupervised learning and the practical needs of labeled tasks, validating the value of self-supervised pre-training in a supervised context.

## 6.2. Classification accuracy

Classification accuracy is a critical metric for measuring the performance of models in accurately identifying and categorizing 3D point cloud data into predefined classes. This metric is especially important in environments where models are fine-tuned on datasets such as ModelNet, following self-supervised pre-training on datasets such as ShapeNet.

To compute classification accuracy, a neural network trained on ModelNet is tested on a labeled dataset of different 3D model classes. The task of the network is to accurately predict the correct class for each input 3D point cloud. Accuracy is computed by comparing the network’s class predictions for each point cloud with the actual labels in the test data.

Mathematically, classification accuracy is expressed as:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \times 100\% \quad (6.1)$$

This formula reflects the proportion of test instances for which the model correctly predicts the class, with the result expressed as a percentage. A higher percentage indicates a better performing model in terms of correctly classifying the point clouds.

Classification accuracy is a simple yet powerful indicator of a model’s ability to generalize from training to unseen data, highlighting the effectiveness of the learned representations for the 3D shape recognition task. By providing a quantifiable and direct comparison of model performance, it allows researchers to assess the impact of different self-supervised learning approaches on the critical task of 3D object classification.

Furthermore, in accordance with previous research [23, 45, 46], a voting technique is employed to assess the classification accuracy of the model. This technique enhances the accuracy and resilience of the model by averaging the outcomes of multiple inferences, each with variations in the input. Specifically, this evaluation is performed by applying a different scaling and translation transformation to the test set on each of the 10 inferences per test case. The logits of all inferences are concatenated and then averaged to form the "vote,"

where each inference contributes an "opinion." The final score is obtained by averaging all these opinions. The class with the highest average score is selected as the final prediction for each sample.

### 6.3. Few-shot classification accuracy

Few-shot classification on ModelNet40 requires models that accurately identify and categorize 3D point cloud data based on a very limited set of labeled examples for each class. This evaluation scenario is particularly challenging and relevant given the scarcity of large labeled datasets in the 3D shape domain.

After pre-training, the model is exposed to the few-shot learning task. Here, it is given a small number of labeled examples (often only one to five instances) per class. These examples form the ‘support set’, which the model uses to fine-tune or adapt its learned representations to the new specific task of classifying the point clouds into their correct categories.

Few-shot classification typically follows an ‘N-way, K-shot’ methodology, where N is the number of different classes and K is the number of examples per class in the support set. The model must use the limited information from these K shots to make accurate predictions about new, unseen instances from the same N classes.

The model’s performance is then evaluated based on its accuracy in classifying these new instances, which are part of the ‘query set’. Few-shot classification accuracy reflects the model’s ability to quickly adapt its representations to new classes from minimal examples. The high accuracy indicates that the model’s self-supervised pre-training has resulted in a versatile and portable feature space that can effectively accommodate rapid learning and generalization.

Building on previous studies in the area of few-shot learning for the ModelNet dataset, the established ‘K-way N-shot’ framework [23, 86, 115] was followed. In this setup,  $K$  different classes are first randomly selected from the dataset, and for each class,  $N + 20$  object instances are sampled. The model is trained on  $K \times N$  samples, which form the support set, to learn the discriminative features of each class. Performance is then evaluated on the remaining  $20K$  samples, called the query set, to test the model’s ability to generalize from the support set to new, unseen instances. This approach ensures a balanced representation of classes and provides a rigorous test of the model’s few-shot learning capabilities.

### 6.4. Segmentation mIoU

In the field of 3D point cloud analysis, especially for datasets such as ShapeNetPart [61], the mean intersection over union (mIoU) serves as an important metric for measuring the effectiveness of part segmentation tasks. Part segmentation goes beyond simple object classification by requiring each point in a point cloud to be classified into one of several

specific part categories.

mIoU is particularly suited to part segmentation because it measures the degree of overlap between the predicted and actual segments for each part category, relative to their combined area. The calculation involves determining the IoU for each individual part category and then averaging these IoU values across all categories to obtain the mean IoU.

Mathematically, the IoU for a part category is calculated as follows:

$$IoU = \frac{\textit{Area of overlap between predicted and actual segments}}{\textit{Area of union of predicted and actual segments}} \quad (6.2)$$

The mean IoU (mIoU) is then obtained by averaging the IoU values across all part categories:

$$mIoU = \frac{\sum \textit{IoU of each part category}}{\textit{Number of part categories}} \quad (6.3)$$

In a self-supervised learning environment, where models are pre-trained on datasets like ShapeNet without part labels, mIoU becomes an essential measure to evaluate how well these models adapt to part segmentation tasks after fine-tuning. A high mIoU score indicates that the model not only identifies the different parts within the point clouds, but also accurately delineates their boundaries. This accurate segmentation is indicative of the model’s ability to recognize and represent the intricate geometric and structural nuances of different object parts, a key strength in applications requiring detailed 3D object analysis.

## 6.5. Dimensionality reduction evaluation

In the field of self-supervised learning, especially when dealing with complex data structures such as 3D point clouds, it is crucial to employ robust evaluation methods that can effectively capture and illustrate the intricate relationships within the data. Dimensionality reduction techniques, in particular t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP), play a key role in this context.

These methods not only help to visualize high-dimensional data in a comprehensible way, but also provide insight into the quality of feature representations derived from self-supervised learning models. In the following subsections, we discuss how each of these techniques contributes to a deeper understanding of the underlying structure of the data, thereby validating the efficacy of self-supervised learning approaches.

### 6.5.1. t-SNE

In evaluating the effectiveness of pre-training for self-supervised learning, especially for complex datasets such as 3D point clouds, t-Distributed Stochastic Neighbor Embedding (t-SNE) [116] emerges as a powerful tool for visualization and analysis. As a nonlinear dimensionality reduction technique, t-SNE is adept at unraveling the intricate structures of high-dimensional data, thereby facilitating a deeper understanding of the feature spaces developed by self-supervised models.

When applying t-SNE in the context of self-supervised learning, the process typically begins by extracting a high-dimensional feature set from the pre-trained model. These features are representative of the model’s internal understanding of the data and are obtained by processing the input through the network to capture the activations of a selected layer. This process results in a rich, multidimensional embedding of each data point, encapsulating the learned nuances and discriminative features identified by the model.

After obtaining these embeddings, t-SNE is used to project the high-dimensional data into a two- or three-dimensional space. The algorithm works by preserving the local structure of the data, ensuring that points that are close in the original feature space remain close in the reduced space, while those that are dissimilar end up farther apart. The resulting scatterplot generated by t-SNE thus provides a visual map of the distribution and clustering tendencies of the data.

The scatter plot serves as an intuitive medium to qualitatively assess the self-training process. Effective pre-training is indicated by the formation of distinct clusters within the plot, with each cluster representing a collection of similar data points. The degree of separation between these clusters visually conveys the ability of the self-supervised model to recognize and distinguish between different data patterns.

While t-SNE is inherently qualitative, it can be complemented with quantitative methods to objectively measure cluster quality, such as calculating the silhouette score. This dual approach allows for both intuitive visual inspection and statistical evaluation of feature representations, providing a comprehensive picture of model performance.

Ultimately, the value of t-SNE in the evaluation pipeline lies in its ability to elucidate the success of self-supervised pre-training in fostering representations that are meaningful and discriminative. Such visual and quantitative insights are invaluable for fine-tuning the pre-training process and providing a solid foundation upon which further supervised or semi-supervised tasks can be built.

### 6.5.2. UMAP

Uniform Manifold Approximation and Projection (UMAP) [117] is a dimensionality reduction technique that can be used to evaluate the quality of embeddings learned through self-supervised learning. Like t-SNE, UMAP is used to visualize high-dimensional data in

two or three dimensions, but it relies on different mathematical foundations and often offers several practical advantages.

To evaluate self-supervised pre-training with UMAP, the process typically involves the following steps:

**Embedding generation:** After a model has been pre-trained in a self-supervised manner, it is used to generate high-dimensional embeddings from the data. These embeddings should encapsulate the features that the model has learned to identify as significant during pre-training.

**Dimensionality reduction:** UMAP is applied to these embeddings to project them into a lower dimensional space for visualization. UMAP works by first estimating the manifold on which the data lies using a nearest neighbor graph, and then optimizing the layout of this graph in low-dimensional space to reflect the high-dimensional structure as closely as possible.

**Visualization and evaluation:** The low-dimensional embeddings generated by UMAP can be plotted, often resulting in a scatter plot that reveals the intrinsic clustering of the data. This visualization can then be used to qualitatively evaluate how well the self-supervised pre-training has captured meaningful structures and relationships within the data.

The main differences between UMAP and t-SNE are as follows

- **Mathematical foundation:** UMAP is based on manifold learning and topological data analysis, while t-SNE is based on probability distributions and stochastic neighbor embedding. UMAP tends to preserve more of the global structure of the data compared to t-SNE, which focuses more on local neighbor relationships.
- **Scalability:** UMAP often scales better to larger datasets than t-SNE, making it more suitable for big data applications. It is generally faster and uses fewer computational resources.
- **Flexibility:** UMAP allows for more customization in its optimization process, such as adjusting the balance between local and global structure, which can be beneficial depending on the specific characteristics of the data.
- **Consistency:** UMAP tends to produce more consistent results across different runs and parameter settings, which can make the interpretation of visualizations more reliable.

## 6.6. Retrieval techniques

A key technique for evaluating the effectiveness of models in self-supervised learning is retrieval. Retrieval is the process of finding and selecting relevant or similar data from a larger data set. This section focuses on two critical evaluation methods in the retrieval process: Nearest Neighbor (NN) and Mean Average Precision (mAP).

### **6.6.1. Nearest neighbor (NN)**

The nearest neighbor technique is used to find the closest instance (or instances) to a given data point in the feature space. In the realm of 3D point clouds, this means identifying the point clouds that are most similar to a given query, based on the learned representations of the model. The success of this technique depends heavily on the quality of the representations generated by the self-supervised learning model. These representations should encapsulate the essential characteristics of the 3D point cloud data and ensure that similar points are clustered within the feature space.

### **6.6.2. Mean average precision (mAP)**

Mean Average Precision (mAP) is a standard metric used to assess the quality of search results in retrieval tasks. It is calculated by averaging the average accuracies obtained at each recall level over a set of queries. In the context of 3D point clouds, the mAP provides a quantitative measure of how well the model retrieves relevant instances under different similarity thresholds. A high mAP indicates that the model not only retrieves relevant instances, but also classifies them with high accuracy, which is crucial for practical applications where accuracy and relevance are paramount.

For the mAP calculation, a result will be considered relevant if the class of the retrieved object matches the class of the query object. This definition of relevance is critical as it ensures that the metric accurately reflects the model’s ability to identify and prioritize the most similar objects based on their structural characteristics. The effectiveness of retrieval is gauged by the proportion of relevant objects that appear at the top of the retrieved list, with particular focus on the initial few results, where precision has a pronounced impact on the mAP metric.

The implementation of retrieval techniques such as NN (Nearest Neighbor) and mAP is pivotal in self-supervised learning of 3D point clouds, as it provides a means of objectively evaluating the quality of the retrieval results.

# Chapter 7

## Experiments to perform

Both hypotheses were evaluated through several experiments. On the one hand, to evaluate the ability of Siamese networks for pre-training neural networks operating on 3D point clouds, standard state-of-the-art benchmarks were followed, qualitatively and quantitatively evaluating the features learned by the encoder during pre-training and its ability to use them for classification and segmentation tasks.

On the other hand, to evaluate the capability of SimpleShape [29] as a dataset for formula-driven supervised learning, the original model proposed by Point-MAE [23] was used and pre-trained on different variations of the SimpleShape dataset with artificially generated samples of 1K, 5K, 10K, 15K, 50K, and 100K. Once trained, the model was evaluated using the same benchmarks as the Siamese network proposal.

This chapter describes how the data used were prepared following the standard methodology in all cases (7.1). It continues with the description of the combinations between models to be compared and their respective encoders (7.2). And finally it describes each of the benchmarks through which the proposed models will be compared with the state of the art (7.3).

### 7.1. Data preparation

The datasets to be used are those previously described in chapter 5. Specifically, the version of the ShapeNet55 dataset presented by Yu, Xumin et al. [111] was used to pre-train the networks. To ensure fair comparisons with other state-of-the-art models, the setup followed by Point-MAE was adopted, sampling 1024 points from each cloud using the Farthest-Point-Sampling algorithm.

In the case of Modelnet, the ModelNet40 version was used to evaluate the model’s ability to classify shapes. Following the experiments of Zhang et al. [89] for linear probing, the same version of ModelNet was used where 1024 points were sampled from each 3D model. For few-shot learning, the dataset was divided according to the experiments performed by Point-

BERT [86]; 10 independent experiments were performed and the average results along with their standard deviation are reported in chapter 8. The benchmarks "5way 10shot", "5way 20shot", "10way 10shot" and "10way 20shot" were followed. For ScanObjectNN, the same variants were chosen as for previous self-supervised learning work: OBJ-BG, OBJ-ONLY, and PB-T50-RS.

## 7.2. Models

In this research, different versions of the PointSIMSIAM and PointBYOL models have been evaluated, each integrated with the encoders proposed in section 4.1. These models undergo the previously detailed linear and masking transformations to produce variants based on PointSIMSIAM and PointBYOL. Encoders include PointNet, DGCNN, Transformer, and Point-MLP, all of which follow the data augmentation strategy based on linear and masking transformations presented in section 4.1.1.

In addition, variations of the original encoders have been experimented with. For example, in PointNet, the T-Net modules were removed, focusing on invariance to linear transformations through spatial transformations. This modification, aimed at enhancing the effects of linear transformations during pre-training, results in a version called ‘PointNet(w/o ST)’.

Regarding the Transformer encoder, two different versions were explored. The first uses the previously defined transformations  $T$  to train the encoder. The second, on the other hand, excludes the last point masking transformation from the set  $T$  and applies the tokenization and token masking technique used by Point-MAE in its pre-training phase. This variant was called Transformer(TM).

Finally, Point-MLP was used as an encoder in two forms: its standard version, also called Point-MLP, and the simplified version known as Point-MLP elite.

Details of the encoders described in this section, including their inference time and number of parameters, are presented in the Table 7.1. The inference time is a crucial aspect of these models, as it determines their applicability in environments that handle 3D models, as pointed out by Xu Ma et al. [47]. The number of parameters, on the other hand, indicates the size of the model and provides insight into the complexity involved in its training and the amount of data required.

Another value indicated in the table is the pre-training time of PointSIMSIAM and Point-BYOL with each of the encoders. This time has an effect on the investigation of the models. However, the most important value is the inference time, which is independent of the pre-training method and will determine how long the model will take in real cases.

From the values shown in the table, it is clear that Transformer [86] has a significantly higher parameter complexity compared to PointNet, DGCNN, and Point-MLP. Therefore, it is desirable to achieve comparable performance to Transformer-based networks while using these simpler network alternatives.



In addition, DGCNN is characterized by a significantly longer inference time, mainly due to intermediate tasks. Despite a relatively small number of parameters, the computational demands imposed by the inference tasks result in a longer execution time. It is worth noting the exceptional inference times achieved by PointNet and Point-MLP elite. These results position them as the most favorable options for practical applications involving 3D point clouds.

Table 7.1: Inference time and pre-training time of proposed methods with different encoders for ModelNet40 train set (subsamped to 1024 points) and the number of trainable parameters for each model’s encoder.

Method	Encoder	Inference time [s]	Pre-training time [h]	# params
PointSIMSIAM	PointNet	11.1	22.0	2.8M
PointBYOL	PointNet	11.1	23.8	2.8M
PointSIMSIAM	PointNet (w/o ST)	5.6	20.7	0.5M
PointBYOL	PointNet (w/o ST)	5.6	20.7	0.5M
PointSIMSIAM	DGCNN	11.5	45.4	1.2M
PointBYOL	DGCNN	11.5	59.2	1.2M
PointSIMSIAM	Transformer	15.3	31.1	22.1M
PointBYOL	Transformer	15.3	39.6	22.1M
PointSIMSIAM	Transformer (TM)	37.0	13.3	22.1M
PointBYOL	Transformer (TM)	37.0	21.6	22.1M
PointSIMSIAM	Point-MLP	27.2	66.2	12.9M
PointBYOL	Point-MLP	27.2	68.8	12.9M
PointSIMSIAM	Point-MLP elite	16.5	29.2	0.6M
PointBYOL	Point-MLP elite	16.5	33.2	0.6M

### 7.3. Standard benchmarks

To evaluate the performance of the proposed models from chapter 4, standard state-of-the-art benchmarks were computed, following previous work on self-supervised learning on 3D point clouds [23, 45–47, 84–86]. Specifically, the ShapeNet dataset [10] was chosen to pre-train different configurations of Siamese networks and then evaluate their generalization ability for classification on Modelnet40, ScanobjectNN, and segmentation on Shapenetpart, following the configurations described in chapter 5 using the metrics outlined in chapter 6. Additionally, the embeddings produced by the different models were qualitatively evaluated using t-SNE and UMAP techniques.

For each experiment, three pre-training and downstream task runs were conducted. In accordance with standard practices in deep learning research, the most favorable results obtained among these three runs are presented.

### 7.3.1. Pre-training

The pre-training setup is performed using the model described in chapter 4. This involves pre-training encoders according to the SIMSIAM and BYOL methods, tailored for 3D point clouds. The ShapeNet55 dataset was used, with the Farthest Point Sample algorithm applied to each model to set the number of points per figure to 1024. The linear transformations and masking described in 4.1.1 are then applied before the models are processed by the network.

For optimization, the AdamW optimizer [118] is used together with the cosine learning rate decay [119]. An initial learning rate of 0.001 is set, with a weight decay of 0.05. The model is pre-trained for 300 epochs with a batch size of 128. Regarding masking, 60% of the total points of each cloud are eliminated.

### 7.3.2. Linear probing

The approach used to evaluate the models by linear probing is similar to that of Point-M2AE [89]: After pre-training each model with ShapeNet, 1024 points are sampled from each 3D model of ModelNet40 [11]. The frozen encoder is then used to extract features, followed by a linear SVM. In addition, a variation of the experiment was performed using a kNN classifier with  $K = 20$  instead of a linear SVM, allowing a broader assessment of the quality of the embeddings produced by the model.

### 7.3.3. Classification

Following the state-of-the-art standard benchmarks, the performance of different models pre-trained on ShapeNet55 [10] was evaluated when fine-tuned over 300 epochs to classify the ModelNet [11] and ScanObjectNN [35] datasets in their three variants. Additionally, few-shot classification experiments were performed on ModelNet40 [11], following the setup of Point-BERT [86], where ModelNet is divided into 4 subsets that are evaluated independently. In all these experiments, the pre-trained encoder is used with different self-supervised learning methods, coupled with an MLP that serves as the classification head of the network.

### 7.3.4. Segmentation

Consistent with other studies on self-supervised learning for 3D point clouds [84, 85], the ability of pre-trained models to generalize acquired knowledge to new tasks was evaluated. For this purpose, ShapeNetPart [61] was chosen as a test dataset for segmentation. The model pre-trained on ShapeNet55 [10] was fine-tuned for the part segmentation task on the new dataset. This task involves classifying each point in the cloud with a label corresponding to the part of the object to which it belongs.

### **7.3.5. Dimensionality Reduction Evaluation**

To evaluate the quality of the representations generated by the pre-trained model, the dimensionality reduction metrics described in section 6.5 were used. Specifically, the feature vector space generated by the model for the objects in ModelNet40 was reduced to two dimensions using t-SNE and UMAP techniques. The purpose of this approach is to determine the ability of the models to generate object class clusters within the feature space.

### **7.3.6. Retrieval Techniques**

Finally, the retrieval techniques described in section 6.6 were applied. This involved creating a  $N \times N$  matrix of the model-generated feature vectors for the  $N$  elements of the ModelNet40 test subset. This matrix captures the cosine similarity between the feature vectors. From this data, the performance of different models was calculated using the NN and mAP metrics.

# Chapter 8

## Results and analysis

This chapter focuses on the analysis of the different results obtained and their contribution to the solution of the problems proposed in this research, namely the exploration of alternatives of self-supervised learning on 3D point clouds.

Following the standard benchmarks described in Chapter 7, the results of PointSIMSIAM and PointBYOL in their different versions for standard benchmarks were examined (section 8.1). The results to be compared from previous state-of-the-art work only consider the values reported by the actual research of each model, as well as reports of similar papers and surveys [84, 85]. Blank values in the results tables indicate that the model result in that comparison has not been officially reported.

Then, the effectiveness of SimpleShape as a pre-training dataset in self-supervised learning techniques was analyzed in comparison to ShapeNet55, the dataset widely used in recent studies (section 8.2).

In benchmarking exercises where results in downstream tasks are compared, a table is included which compares the results obtained when the model is used in its original form (from scratch) to perform the task, with those obtained when it is pre-trained with PointSIMSIAM and POINTBYOL, and then used to perform the same task. The purpose of these tables is to evaluate the difference in performance between the encoder without pre-training and with pre-training.

These experiments were conducted using three GeForce RTX A5000 GPUs, each with 24 GB of video memory.

### 8.1. Siamese networks

In line with current standard benchmarks, the effectiveness of Siamese networks in pre-training encoders for 3D point clouds was assessed, aiming to generate representations that are useful for various downstream tasks.

At the end of each subsection, a comparison was also made with the results of the original models trained from scratch for the evaluated fine-tuning tasks. It is important to note that the models trained from scratch were trained in a supervised manner, using the original labels of the figures. The purpose of this evaluation is to determine whether the proposed pre-training model indeed improves the performance of encoders on downstream tasks with 3D datasets.

### 8.1.1. Linear probing

Following the instructions in section 7.3.2, the linear probing experiments were performed with *KNN* ( $k = 20$ ) and *SVM*, and the classification accuracies are reported in Table 8.1. As shown, among the proposed models, PointBYOL with a DGCNN encoder achieves the best performance in *SVM*, reaching an accuracy of 91.7%. This result is slightly better than ConClu [99], the most effective contrastive learning model, with 91.6%. However, PointMA2E achieves the best overall results in the experiment with *SVM*, with an accuracy of 93.1%. For the *KNN* classifiers, the results show that the PointBYOL model with a DGCNN encoder achieves the best classification accuracy, with an accuracy of 85.5%.

Almost all the proposed models achieve an accuracy higher than 75%, except for PointSIMSIAM with PointNet, which gives lower results. Several conclusions can be drawn from these results:

First, DGCNN emerges as the coder that achieves the best results with both PointSIMSIAM and PointBYOL, outperforming all state-of-the-art contrastive learning models. This suggests that DGCNN is the coder that produces the most effective representations after pre-training.

Second, PointSIMSIAM and PointBYOL show similar results when all encoders are considered. While the best overall result is obtained with PointBYOL for Transformer and Point-MLP, the best individual performance is obtained with PointSIMSIAM.

Third, PointSIMSIAM does not perform well with PointNet, which may be due to the fact that the proposed transformations are not sufficient to prevent the encoder results from collapsing. When training PointNet under the PointSIMSIAM methodology, the transformations aim to sufficiently differentiate the inputs so that the T-net modules for spatial transformations do not quickly lead to a collapse of the results. However, it appears that this goal is not fully achieved, resulting in the encoder not learning effectively during training.

These initial results confirm that the proposed methodology of Siamese networks is suitable for pre-training models working with 3D point clouds. Subsequent experiments were conducted to measure the self-supervised learning methods of the state-of-the-art [84, 85].

Table 8.1: Linear evaluation on ModelNet40 [11] using *SVM* and *KNN* ( $k = 20$ ). Methods are categorized according to their self-supervised learning (SSL) methodology. The best of the proposed models and the best state-of-the-art model are highlighted in bold. Models are organized by methodology: first, denoising autoencoder models; then, contrastive learning-based models; and finally, the proposed models.

Method	Backbone	SVM Acc. (%)	kNN Acc. (%)
Point-DAE	PointNet	89.3	-
Point-DAE	DGCNN	91.9	-
Point-BERT	Transformer	87.4	-
Point-MAE	Transformer	91.0	-
Point-M2AE	Transformer	92.9	-
Point-MA2E	PointNet	89.4	-
Point-MA2E	DGCNN	92.1	-
Point-MA2E	Transformer	<b>93.1</b>	-
DepthContrast	PointNet++	85.4	-
STLR	PointNet	88.3	-
STLR	DGCNN	90.9	-
GISR	DGCNN	90.4	-
DCGLR	3D-ViT	91.3	-
DCGLR	PCT	91.4	-
ConClu	DGCNN	91.6	-
PointSIMSIAM	PointNet	67.9	60.6
PointBYOL	PointNet	85.3	76.9
PointSIMSIAM	PointNet(w/o ST)	88.0	79.0
PointBYOL	PointNet(w/o ST)	88.3	79.7
PointSIMSIAM	DGCNN	89.6	81.7
PointBYOL	DGCNN	<b>91.7</b>	<b>85.5</b>
PointSIMSIAM	Transformer	87.3	82.6
PointBYOL	Transformer	85.7	80.5
PointSIMSIAM	Transformer(TM)	88.3	82.3
PointBYOL	Transformer(TM)	76.0	74.7
PointSIMSIAM	Point-MLP	88.8	83.6
PointBYOL	Point-MLP	88.2	83.7
PointSIMSIAM	Point-MLP elite	87.5	83.2
PointBYOL	Point-MLP elite	88.4	82.7

### 8.1.2. Classification on ModelNet40

Following the standard benchmarks for self-supervised learning on 3D point clouds, PointSIMSIAM and PointBYOL were fine-tuned on the ModelNet40 [11] dataset, after being pre-

trained with ShapeNet55. All model variations proposed in section 4.1 were used and the results are shown in Table 8.2.

Unfortunately, the standard approach is to use the voting method, so the raw accuracy results for many models are unknown. In terms of voting accuracy, several proposed models outperform state-of-the-art contrastive learning models. In terms of autoencoders, both Point-MAE and Point-M2AE achieve better results than Siamese networks, although they remain competitive.

The best performing models among the proposed variants are PointBYOL with a Point-MLP encoder, which achieves 93.6% accuracy with voting. PointBYOL with Point-MLP elite obtains 93% raw classification accuracy. This result matches the performance of ContrastMPCT [120], a model that uses Transformer as the encoder, while Point-MLP elite is a simpler and lighter network.

All models based in contrastive learning evaluated with voting accuracy are outperformed by several proposed models. However, although these models outperform Point-BERT and Point-DAE from the state of the art in voting accuracy, Point-M2AE with a Transformer encoder achieves the highest accuracy at 94.0%.

Each PointSIMSIAM and PointBYOL model was also compared with the original results obtained by their encoders, trained in a supervised manner without pre-training on ModelNet40. The aim is to verify if pre-training with Siamese networks does indeed improve the final classification results on this dataset.

Looking at the results in Table 8.3, PointNet [45], DGCNN [48], and Transformer [86] perform better with the Siamese network method compared to training only on ModelNet40 in a supervised manner. However, Point-MLP [47] and Point-MLP elite are negatively affected by pre-training, with PointSIMSIAM and PointBYOL performing worse than without pre-training. This may be due to the structure of Point-MLP being overly specialized for this benchmark, or the pre-training parameters being suboptimal, thereby distracting the encoder from classifying correctly.

While the pre-trained models do not reach the level of those pre-trained with autoencoders, the proposed Siamese networks significantly improve the results of different encoders. This suggests that further research on the chosen transformations could enable the models to achieve even better results.

Table 8.2: Comparison of pre-trained models vs from scratch performance: Shape classification on ModelNet40. ‘Acc (%)’ denotes overall accuracy and ‘Voting Acc (%)’ denotes voting accuracy. The results are listed starting with autoencoder-based models, followed by models that primarily use contrastive learning, and finally the results of the proposed models.

Method	Backbone	Acc. (%)	Voting Acc. (%)
Point-DAE	PointNet		90.6
Point-DAE	DGCNN		93.3
Point-BERT	Transformer		93.2
Point-MAE	Transformer		93.8
Point-M2AE	Transformer		<b>94.0</b>
DepthContrast [97]	PointNet++		85.4
ContrastMPCT [120]	Transformer	<b>93.3</b>	
STRL [98]	PointNet		88.6
STRL [98]	DGCNN		90.8
PointSIMSIAM	PointNet	91.1	90.9
PointBYOL	PointNet	91.3	91.2
PointSIMSIAM	PointNet(w/o ST)	89.9	90.0
PointBYOL	PointNet(w/o ST)	90.0	90.3
PointSIMSIAM	DGCNN	92.3	92.6
PointBYOL	DGCNN	93.0	93.0
PointSIMSIAM	Transformer	92.5	92.5
PointBYOL	Transformer	92.7	93.1
PointSIMSIAM	Transformer(TM)	91.9	92.5
PointBYOL	Transformer(TM)	92.5	92.9
PointSIMSIAM	Point-MLP	92.9	93.0
PointBYOL	Point-MLP	93.0	<b>93.6</b>
PointSIMSIAM	Point-MLP elite	92.7	92.3
PointBYOL	Point-MLP elite	<b>93.3</b>	93.1



Table 8.3: Shape classification on ModelNet40. ‘Voting Acc (%)’ indicates voting accuracy. Each version of PointSIMSIAM and PointBYOL is distinguished, along with the ‘from scratch’ results of their respective encoders.

Method	Backbone	Voting Acc. (%)
<i>From scratch</i>	PointNet	89.2
PointSIMSIAM	PointNet	90.9
PointBYOL	PointNet	<b>91.2</b>
PointSIMSIAM	PointNet(w/o ST)	90.0
PointBYOL	PointNet(w/o ST)	90.3
<i>From scratch</i>	DGCNN	92.9
PointSIMSIAM	DGCNN	92.6
PointBYOL	DGCNN	<b>93.0</b>
<i>From scratch</i>	Transformer [86]	91.4
PointSIMSIAM	Transformer	92.5
PointBYOL	Transformer	<b>93.1</b>
PointSIMSIAM	Transformer(TM)	92.5
PointBYOL	Transformer(TM)	92.9
<i>From scratch</i>	Point-MLP	<b>94.5</b>
PointSIMSIAM	Point-MLP	93.0
PointBYOL	Point-MLP	93.6
<i>From scratch</i>	Point-MLP elite	<b>94.0</b>
PointSIMSIAM	Point-MLP elite	92.3
PointBYOL	Point-MLP elite	93.1

### 8.1.3. Classification on ScanObjectNN

The results for classification on ScanObjectNN are shown in the Table 8.4. Among the proposed models, the best performance in this task is achieved by PointBYOL with Point-MLP elite as the backbone in the OBJ-ONLY variant, and PointBYOL with Point-MLP as the backbone in the OBJ-BG and OBJ-PB categories of ScanObjectNN.

Among all current models, Point-MA2E [90] achieves the best results in the OBJ-ONLY category with an accuracy of 93.07% and in OBJ-PB with an accuracy of 89.31%. However, the proposed configuration of PointBYOL with Point-MLP backbone outperforms the state-of-the-art proposals in the OBJ-BG category, achieving a classification accuracy of 94.15%. It is noteworthy that according to Fei et al. [85], ContrastMPCT [120] is the only contrastive learning method evaluated under this benchmark, and the models with PointBYOL with Point-MLP and Point-MLP Elite backbones significantly outperform it.

As in the previous section, the results of each backbone are compared with its unpretrained version in Table 8.5. The version trained only on ScanObjectNN with labels is called ‘from scratch’. Although most of the proposed models achieve competitive results, exceeding 85%

accuracy, those using Point-MLP and Point-MLP elite as encoders stand out.

Since there are no official results from these encoders for OBJ-ONLY and OBJ-BG, it remains unclear whether the results are due to the proposed methodology or to the quality of the encoder. To address this, Table 8.5 shows that for the PB-T50-RS variant, the most complex in the dataset, the Point-MLP encoder improves its performance by about 4.6%, demonstrating that it does indeed improve performance thanks to pre-training with PointBYOL.

Regarding the self-supervised learning models using contrastive learning, only ContrastMPCT is identified, which is outperformed in its results. However, it is noteworthy that using Transformer as an encoder, the same as ContrastMPCT, yields worse results in classifying ScanObjectNN. This difference in results suggests that Transformer networks do not benefit as much from Siamese network pre-training as Point-MLP, which is not a major issue since these models require a large amount of data. The performance improvement with Point-MLP over Transformer is positive for both the increased accuracy and the smaller dimension of the encoder, which favors the goal of self-supervised learning to reduce the amount of data required for training.

Table 8.4: Accuracy (%) results of fine-tuning on ScanObjectNN classification. Models are categorized according to their pre-training methodology: first, models based on the autoencoder methodology; then, state-of-the-art models following contrastive learning; and finally, the models proposed in this research.

Method	Backbone	OBJ-ONLY	OBJ-BG	PB-T50-RS
Point-DAE	PointNet		80.2	
Point-DAE	DGCNN		92.1	
Point-BERT	Transformer	88.12	87.43	83.07
Point-MAE	Transformer	88.29	90.02	85.18
Point-M2AE	Transformer	88.81	91.22	86.43
MAE3D	DGCNN			83.21
Point-MA2E	PointNet		80.2	
Point-MA2E	DGCNN		92.1	
Point-MA2E	Transformer	<b>93.07</b>	<b>93.86</b>	<b>89.31</b>
ContrastMPCT [120]	Transformer	90.42	90.15	85.50
PointSIMSIAM	PointNet	77.97	78.31	69.00
PointBYOL	PointNet	82.44	81.58	69.81
PointSIMSIAM	PointNet(w/o ST)	79.69	78.83	69.04
PointBYOL	PointNet(w/o ST)	80.89	79.00	68.98
PointSIMSIAM	DGCNN	88.81	88.98	81.57
PointBYOL	DGCNN	89.15	88.81	82.27
PointSIMSIAM	Transformer	85.71	86.57	81.57
PointBYOL	Transformer	88.64	89.33	84.73
PointSIMSIAM	Transformer(TM)	87.95	88.64	83.48
PointBYOL	Transformer(TM)	86.92	85.37	80.22
PointSIMSIAM	Point-MLP	90.19	92.08	87.95
PointBYOL	Point-MLP	90.71	<b>94.15</b>	<b>89.00</b>
PointSIMSIAM	Point-MLP elite	89.67	91.37	85.60
PointBYOL	Point-MLP elite	<b>91.74</b>	91.05	87.02

Table 8.5: Comparison of pre-trained models vs from scratch performance: Accuracy (%) results of fine-tuning ScanObjectNN classification, comparing each Siamese network model with its backbone results when trained from scratch.

Method	Backbone	OBJ-ONLY	OBJ-BG	PB-T50-RS
<i>From scratch</i>	PointNet	73.3	79.2	68.0
PointSIMSIAM	PointNet	77.97	78.31	69.00
PointBYOL	PointNet	<b>82.44</b>	<b>81.58</b>	<b>69.81</b>
PointSIMSIAM	PointNet(w/o ST)	79.69	78.83	69.04
PointBYOL	PointNet(w/o ST)	80.89	79.00	68.98
<i>From scratch</i>	DGCNN	86.2	82.8	78.1
PointSIMSIAM	DGCNN	88.81	<b>88.98</b>	81.57
PointBYOL	DGCNN	<b>89.15</b>	88.81	<b>82.27</b>
<i>From scratch</i>	Transformer	79.86	80.55	77.24
PointSIMSIAM	Transformer	85.71	86.57	81.57
PointBYOL	Transformer	<b>88.64</b>	<b>89.33</b>	<b>84.73</b>
PointSIMSIAM	Transformer(TM)	87.95	88.64	83.48
PointBYOL	Transformer(TM)	86.92	85.37	80.22
<i>From scratch</i>	Point-MLP	-	-	85.4 ± 0.3
PointSIMSIAM	Point-MLP	90.19	92.08	87.95
PointBYOL	Point-MLP	<b>90.71</b>	<b>94.15</b>	<b>89.00</b>
<i>From scratch</i>	Point-MLP elite	-	-	83.8 ± 0.6
PointSIMSIAM	Point-MLP elite	89.67	<b>91.37</b>	85.60
PointBYOL	Point-MLP elite	<b>91.74</b>	91.05	<b>87.02</b>

#### 8.1.4. Few-shot classification

The primary goal of the few-shot classification benchmark is to evaluate the quality of pre-trained encoders in terms of their ability to quickly adapt to new target datasets.

The results are presented in Table 8.6. As in previous experiments, the autoencoder-based models are reported first, followed by those using contrastive learning, and finally the models proposed in this research. The reported results pertain to the mean and standard deviation of the accuracy of the model across the 10 independent experiments defined by Sharma et al. [115]. It is notable that Point-M2AE [89] and ContrastMPCT [120] achieve the best results in their respective groups.

Comparing the proposed models with the best state-of-the-art, PointBYOL with DGCNN achieves the best results in the ‘5w10s’ and ‘5w20s’ benchmarks with  $97.1\% \pm 2.1\%$  and  $98.9\% \pm 1.4\%$  accuracy, respectively. In other benchmarks, the proposed networks rank third in ‘10w10s’ and second in ‘10w20s’. Beyond the achieved performance, it is noteworthy that convolutional networks outperform Transformer models pre-trained with autoencoders and

contrastive learning.

Each version of the Siamese network is also compared to its encoder trained from scratch in the Table 8.7. It is observed that all encoders benefit from pre-training with PointBYOL and PointSIMSIAM. Unfortunately, Point-MLP does not report its results in this benchmark, so the results are only reported in the last two sections of the table.

Analyzing the results in Table 8.7, it is clear that encoders trained from scratch are significantly less efficient than those pre-trained. The most striking example is DGCNN, which achieves an accuracy of only 16.9% under the ‘10w20s’ method, while pre-training with PointBYOL increases its accuracy to 95.1%, resulting in an approximate increase of 78.2% due to the Siamese network pre-training alone.

Table 8.6: Few-shot classification results on ModelNet40. Mean accuracy (%) and standard deviation (%) from 10 independent experiments are reported.

Method	Backbone	5w10s	5w20s	10w10s	10w20s
Point-DAE	PointNet	93.0 ± 3.7	94.9 ± 3.3	86.7 ± 5.8	92.1 ± 4.6
Point-DAE	DGCNN	96.7 ± 2.5	97.7 ± 1.6	93.0 ± 3.8	<b>95.6 ± 2.6</b>
Point-BERT	Transformer	94.6 ± 3.1	96.3 ± 2.7	91.0 ± 5.4	92.7 ± 5.1
Point-MAE	Transformer	96.3 ± 2.5	97.8 ± 1.8	92.6 ± 4.1	95.0 ± 3.0
Point-M2AE	Transformer	<b>96.8 ± 1.8</b>	98.3 ± 4.5	92.3 ± 4.5	95.0 ± 3.0
MAE3D	Transformer	95.2 ± 3.1	97.9 ± 1.6	91.1 ± 4.6	94.2 ± 3.8
ContrastMPCT [120]	Transformer	96.5 ± 1.7	<b>98.5 ± 1.7</b>	<b>93.0 ± 2.4</b>	95.2 ± 2.0
Cover-tree [115]	PointNet	63.2 ± 10.7	68.9 ± 9.4	49.2 ± 6.1	50.1 ± 5.0
Cover-tree [115]	DGCNN	60.0 ± 8.9	65.7 ± 8.4	48.5 ± 5.6	53.0 ± 4.1
PointSIMSIAM	PointNet	90.8 ± 6.6	93.8 ± 5.1	84.3 ± 6.4	90.8 ± 5.1
PointBYOL	PointNet	93.3 ± 3.4	96.4 ± 1.9	87.9 ± 5.2	92.6 ± 4.5
PointSIMSIAM	PointNet(w/o ST)	93.2 ± 3.4	94.9 ± 3.2	87.5 ± 6.2	92.1 ± 4.1
PointBYOL	PointNet(w/o ST)	93.7 ± 3.0	94.9 ± 3.0	87.7 ± 5.9	92.0 ± 4.2
PointSIMSIAM	DGCNN	96.8 ± 2.3	<b>98.8 ± 1.4</b>	92.6 ± 4.5	95.3 ± 2.8
PointBYOL	DGCNN	<b>97.1 ± 2.1</b>	98.4 ± 0.9	92.3 ± 4.0	<b>95.5 ± 2.8</b>
PointSIMSIAM	Transformer	94.7 ± 3.1	97.0 ± 2.5	90.2 ± 5.2	94.3 ± 3.0
PointBYOL	Transformer	95.7 ± 3.6	98.4 ± 1.2	92.4 ± 4.5	95.1 ± 2.8
PointSIMSIAM	Transformer(TM)	95.5 ± 2.8	96.9 ± 1.9	91.2 ± 4.6	94.5 ± 2.4
PointBYOL	Transformer(TM)	93.8 ± 2.7	95.7 ± 2.5	88.9 ± 5.2	93.1 ± 4.0
PointSIMSIAM	Point-MLP	95.2 ± 3.0	98.3 ± 1.6	91.9 ± 4.8	94.9 ± 2.6
PointBYOL	Point-MLP	96.7 ± 2.3	98.0 ± 1.9	<b>92.6 ± 4.1</b>	95.1 ± 2.5
PointSIMSIAM	Point-MLP elite	95.1 ± 1.8	98.1 ± 1.8	92.0 ± 4.9	94.5 ± 3.0
PointBYOL	Point-MLP elite	95.8 ± 2.8	97.4 ± 2.3	91.7 ± 4.3	95.0 ± 2.5

Table 8.7: Comparison of pre-trained models vs from scratch performance: Few-shot classification results on ModelNet40, comparing each Siamese network model with its backbone results when trained from scratch. Mean accuracy (%) and standard deviation (%) from 10 independent experiments are shown.

Method	Backbone	5w10s	5w20s	10w10s	10w20s
<i>From scratch</i>	PointNet	52.0 ± 12.2	57.8 ± 15.5	46.6 ± 13.5	35.2 ± 15.3
PointSIMSIAM	PointNet	90.8 ± 6.6	93.8 ± 5.1	84.3 ± 6.4	90.8 ± 5.1
PointBYOL	PointNet	93.3 ± 3.4	<b>96.4 ± 1.9</b>	<b>87.9 ± 5.2</b>	<b>92.6 ± 4.5</b>
PointSIMSIAM	PointNet(w/o ST)	93.2 ± 3.4	94.9 ± 3.2	87.5 ± 6.2	92.1 ± 4.1
PointBYOL	PointNet(w/o ST)	<b>93.7 ± 3.0</b>	94.9 ± 3.0	87.7 ± 5.9	92.0 ± 4.2
<i>From scratch</i>	DGCNN	36.1 ± 2.8	40.8 ± 4.6	19.9 ± 2.1	16.9 ± 1.5
PointSIMSIAM	DGCNN	96.8 ± 2.3	<b>98.8 ± 1.4</b>	<b>92.6 ± 4.5</b>	95.3 ± 2.8
PointBYOL	DGCNN	<b>97.1 ± 2.1</b>	98.4 ± 0.9	92.3 ± 4.0	<b>95.5 ± 2.8</b>
<i>From scratch</i>	Transformer	87.8 ± 5.2	93.3 ± 4.3	84.6 ± 5.5	89.4 ± 6.3
PointSIMSIAM	Transformer	94.7 ± 3.1	97.0 ± 2.5	90.2 ± 5.2	94.3 ± 3.0
PointBYOL	Transformer	<b>95.7 ± 3.6</b>	<b>98.4 ± 1.2</b>	<b>92.4 ± 4.5</b>	<b>95.1 ± 2.8</b>
PointSIMSIAM	Transformer(TM)	95.5 ± 2.8	96.9 ± 1.9	91.2 ± 4.6	94.5 ± 2.4
PointBYOL	Transformer(TM)	93.8 ± 2.7	95.7 ± 2.5	88.9 ± 5.2	93.1 ± 4.0
<i>From scratch</i>	Point-MLP	-	-	-	-
PointSIMSIAM	Point-MLP	95.2 ± 3.0	<b>98.3 ± 1.6</b>	91.9 ± 4.8	94.9 ± 2.6
PointBYOL	Point-MLP	<b>96.7 ± 2.3</b>	98.0 ± 1.9	<b>92.6 ± 4.1</b>	<b>95.1 ± 2.5</b>
<i>From scratch</i>	Point-MLP elite	-	-	-	-
PointSIMSIAM	Point-MLP elite	95.1 ± 1.8	<b>98.1 ± 1.8</b>	<b>92.0 ± 4.9</b>	94.5 ± 3.0
PointBYOL	Point-MLP elite	<b>95.8 ± 2.8</b>	97.4 ± 2.3	91.7 ± 4.3	<b>95.0 ± 2.5</b>

### 8.1.5. Point cloud segmentation

Another key aspect of a successful self-supervised learning method is its ability to generalize to different types of tasks. To evaluate this property, the downstream task of segmenting the ShapeNetPart [61] dataset is used. The goal of this experiment is to evaluate the generalization ability of the pre-trained encoders in feature generation, extending the downstream tasks beyond simple classification. The segmentation heads originally proposed for each backbone were adopted.

As reported in Table 8.8, the Siamese network versions with the best results are PointSIMSIAM and PointBYOL with the Point-MLP encoder, reaching a  $MIoU_I$  of 85.9. However, they do not surpass the results of ContrastMPCT [120] and Point-M2AE [89], which are the leading models in contrast learning and autoencoders, respectively.

Similar to previous experiments, Table 8.9 reports the results of each variant of PointSIMSIAM and PointBYOL compared to the original version of their encoders trained from scratch. From this table, we can see that PointNet [45], DGCNN [48], and the Transformer encoder [86] benefit from pre-training for segmentation. On the other hand, Point-MLP [47]

reduces its  $mIoU_I$  from 86.1 to 85.9, and its simplified version, Point-MLP elite, reaches 85.8 with PointBYOL. This last result is not compared to Ma et al. [47] as they do not report their results in this benchmark.

These results suggest that the generalization capability of pre-training with Siamese networks is less efficient than masked autoencoder methods. Although the segmentation results are not the best compared to other pre-training methods, the possibility of improving the transformations used or using other Siamese network methods, such as SwAV or SimCLR, to achieve better performance is not ruled out.

Table 8.8: Part Segmentation on ShapeNetPart [61]. ‘ $mIoU_I$ ’ denotes the mean IoU of the model over all instances in the dataset. The results are categorized based on autoencoder models, contrastive models, and proposed models.

<b>Method</b>	<b>Backbone</b>	$mIoU_I$
Point-DAE	PointNet	84.7
Point-DAE	DGCNN	85.9
Point-BERT	Transformer	85.6
Point-MAE	Transformer	86.1
Point-M2AE	Transformer	<b>86.5</b>
Point-MA2E	PointNet	84.8
Point-MA2E	DGCNN	86.0
Point-MA2E	Transformer	86.4
PointContrast [96]	SR-UNet	85.1
ContrastMPCT [120]	Transformer	86.2
ConClu [99]	DGCNN	85.4
PointSIMSIAM	PointNet	84.4
PointBYOL	PointNet	84.6
PointSIMSIAM	PointNet(w/o ST)	84.4
PointBYOL	PointNet(w/o ST)	84.3
PointSIMSIAM	DGCNN	85.3
PointBYOL	DGCNN	85.3
PointSIMSIAM	Transformer	85.6
PointBYOL	Transformer	85.6
PointSIMSIAM	Transformer(TM)	85.6
PointBYOL	Transformer(TM)	85.7
PointSIMSIAM	Point-MLP	<b>85.9</b>
PointBYOL	Point-MLP	<b>85.9</b>
PointSIMSIAM	Point-MLP elite	85.6
PointBYOL	Point-MLP elite	85.8

Table 8.9: Comparison of pre-trained models vs from scratch performance: Part segmentation on ShapeNetPart [61], comparing each Siamese network model with its backbone results when trained from scratch. ‘ $mIoU_I$ ’ denotes the model’s mean IoU across all instances in the dataset. The results are categorized based on autoencoder models, contrastive models, and proposed models.

Method	Backbone	$mIoU_I$
<i>From scratch</i>	PointNet	83.7
PointSIMSIAM	PointNet	84.4
PointBYOL	PointNet	<b>84.6</b>
PointSIMSIAM	PointNet(w/o ST)	84.4
PointBYOL	PointNet(w/o ST)	84.3
<i>From scratch</i>	DGCNN	85.2
PointSIMSIAM	DGCNN	<b>85.3</b>
PointBYOL	DGCNN	<b>85.3</b>
<i>From scratch</i>	Transformer	85.1
PointSIMSIAM	Transformer	85.6
PointBYOL	Transformer	85.6
PointSIMSIAM	Transformer(TM)	85.6
PointBYOL	Transformer(TM)	<b>85.7</b>
<i>From scratch</i>	Point-MLP	<b>86.1</b>
PointSIMSIAM	Point-MLP	85.9
PointBYOL	Point-MLP	85.9
<i>From scratch</i>	Point-MLP elite	-
PointSIMSIAM	Point-MLP elite	85.6
PointBYOL	Point-MLP elite	<b>85.8</b>

### 8.1.6. Dimensionality reduction evaluation

As mentioned in chapter 6, t-SNE and UMAP are qualitative evaluation methods that allow a visual assessment of the quality of representations generated by a pre-trained encoder. To evaluate the proposed models, after pre-training on ShapeNet55, the encoder is used to generate embeddings of each ModelNet40 model. These embeddings are then used to generate two-dimensional representations using t-SNE and UMAP.

Effective pre-training of the encoder should visually cluster objects of the same class close together, while separating those of different classes as much as possible, forming one cluster for each class. For this purpose, the results of these evaluation techniques are presented on a 2D plane, where the points are colored according to the label of each object transformed into an embedding.

Based on this, a qualitative analysis of the generated clusters is performed, evaluating the quality of the pre-training by examining the relationship between the generated clusters and



the classes of the dataset. This benchmark is related to linear probing in that it measures the ease of classifying classes based solely on the features generated by the pre-trained model.

Comparing the t-SNE results from Figures 8.1 and 8.2 with the UMAP results from Figures 8.3 and 8.4, the first noticeable difference is the scale of the results. UMAP projects the results into a two-dimensional range with a maximum value of less than 20, while t-SNE reaches values close to 75. This means that, mathematically, the different classes of ModelNet40 appear closer on the plane in UMAP than in t-SNE, but this observation can be misleading because these numerical values are not the main focus of these benchmarks.

The results with t-SNE are consistent across the board. In particular, PointSIMSIAM with PointNet does not generate specific clusters for the classes, but groups them all into one cluster. In the remaining images, several distinct groups can be identified, along with a central cloud of points composed of elements from different classes. It's significant that PointNet without spatial transformations discriminates a large number of classes, unlike PointNet with spatial transformations, suggesting that while T-Net modules may improve the performance of PointNet, they make pre-training with the proposed combination of transformations more difficult.

UMAP produces figures similar to t-SNE, but on a smaller scale. With PointSIMSIAM and PointNet, UMAP also discriminates only one class from the rest, while PointNet (without ST) produces more defined clusters. UMAP not only reduces the range of values taken by the features, but also accentuates the differences between clearly differentiated classes and brings those with similar feature vectors closer together. It is observed that PointBYOL increases the grouping of classes in the case of PointNet, Transformer and Point-MLP. These results are consistent with those obtained in previous benchmarks, where pre-training with Siamese networks effectively helps the coders to better differentiate elements of a data set.

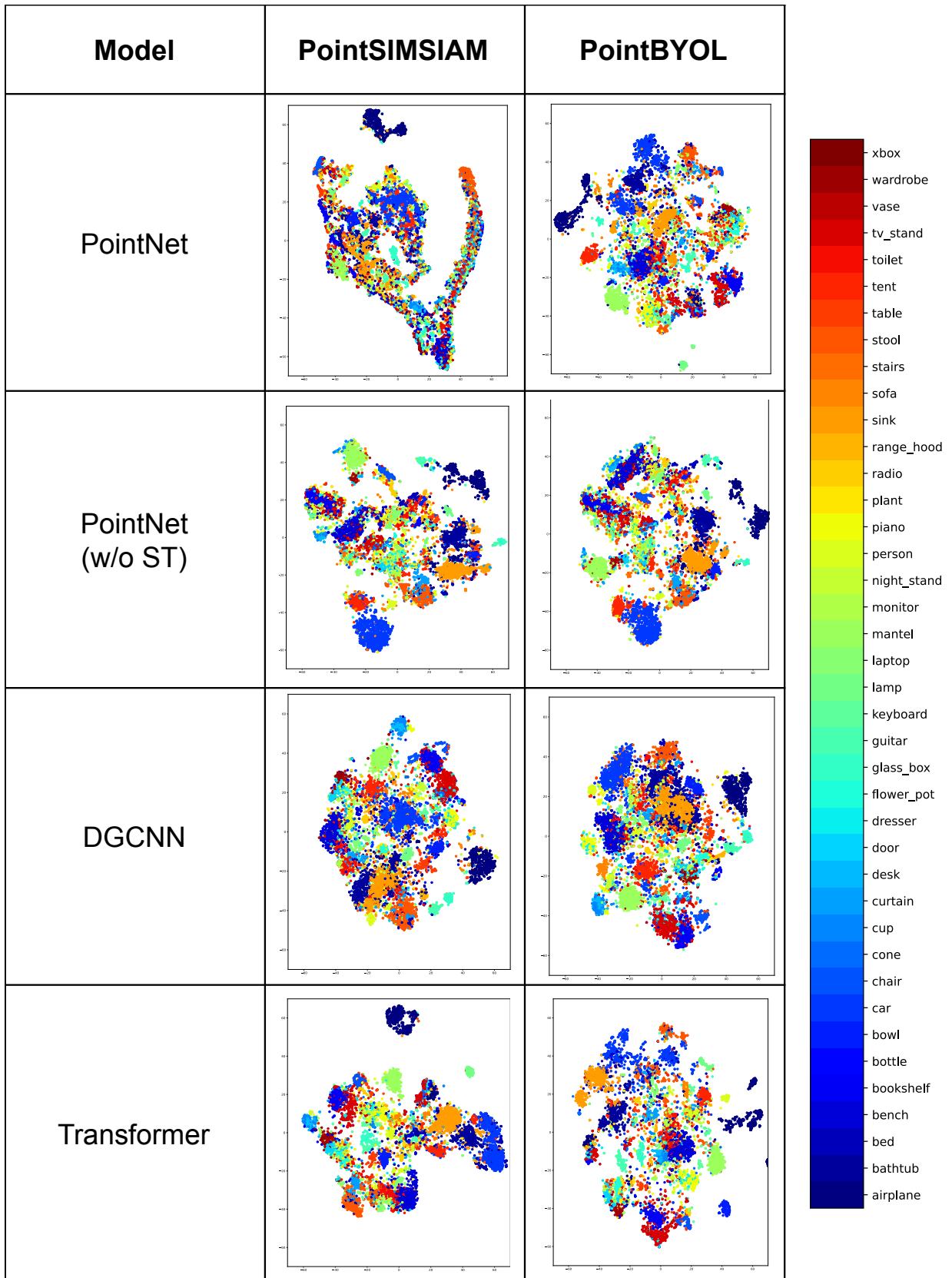


Figure 8.1: **Siamese networks t-SNE results part 1.** t-SNE is used to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-70, 70].

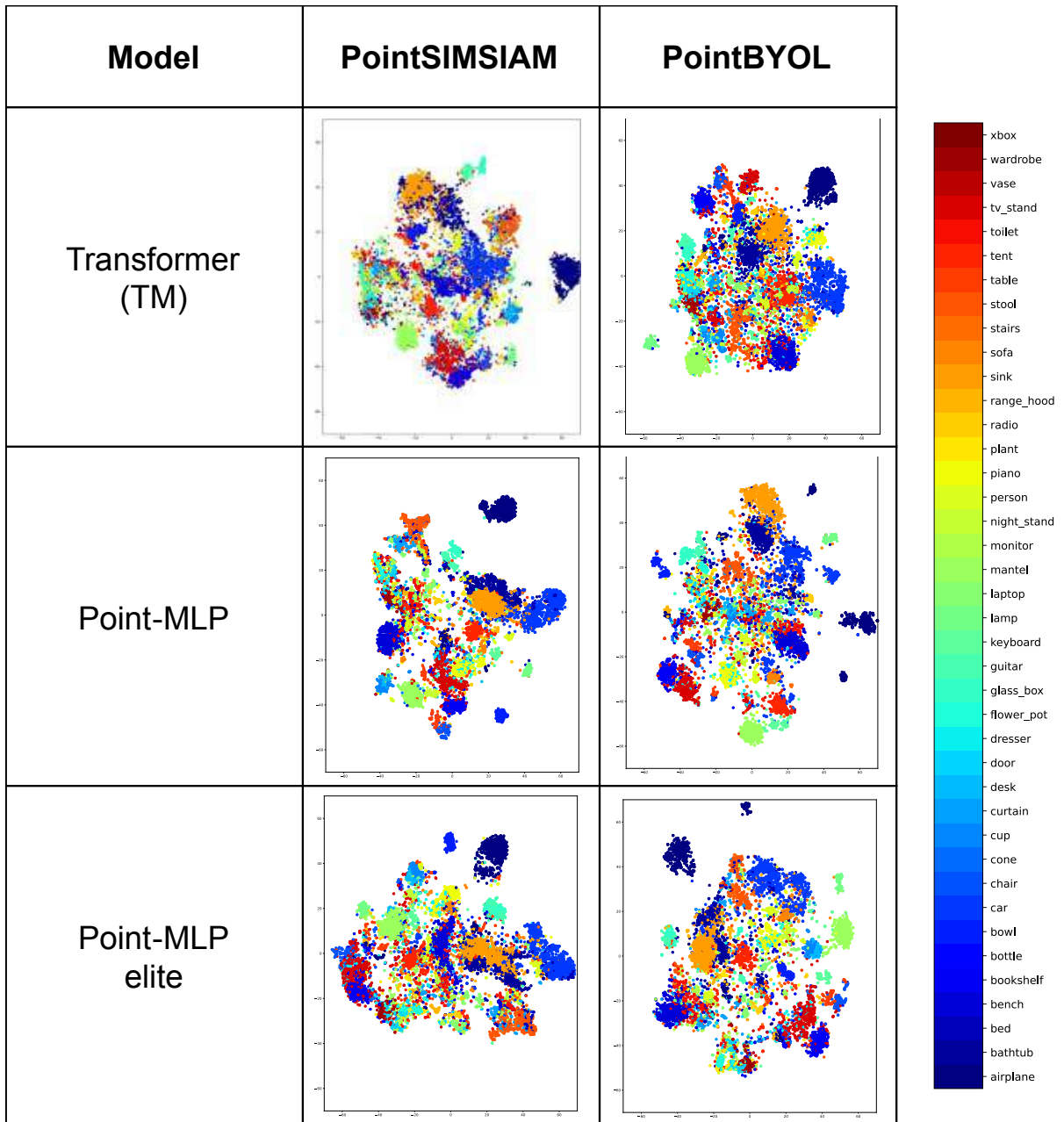


Figure 8.2: **Siamese networks t-SNE results part 2.** t-SNE is used to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from [-70, 70].

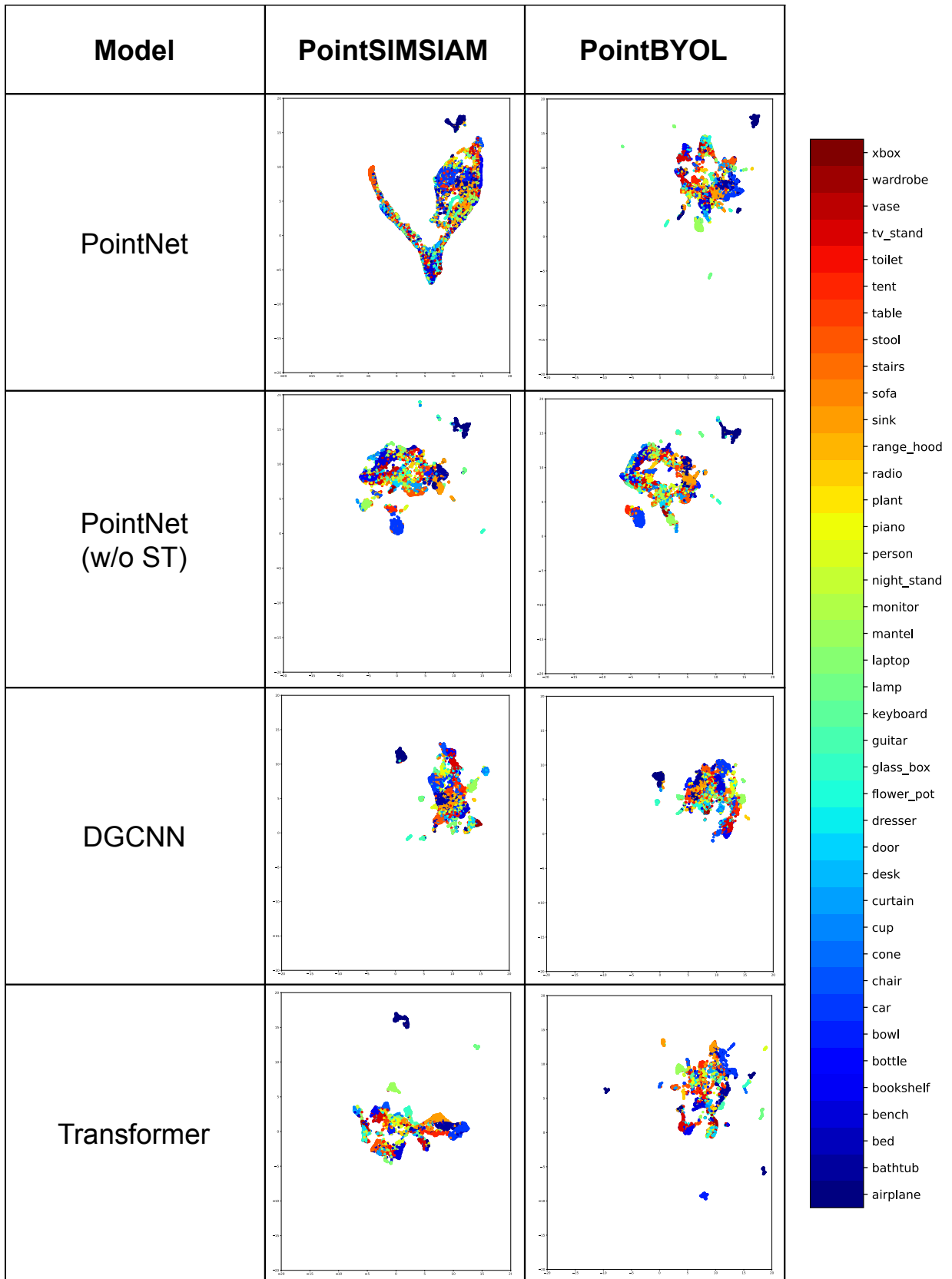


Figure 8.3: **Siamese networks UMAP results part 1**. UMAP is applied to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from  $[-20, 20]$ .

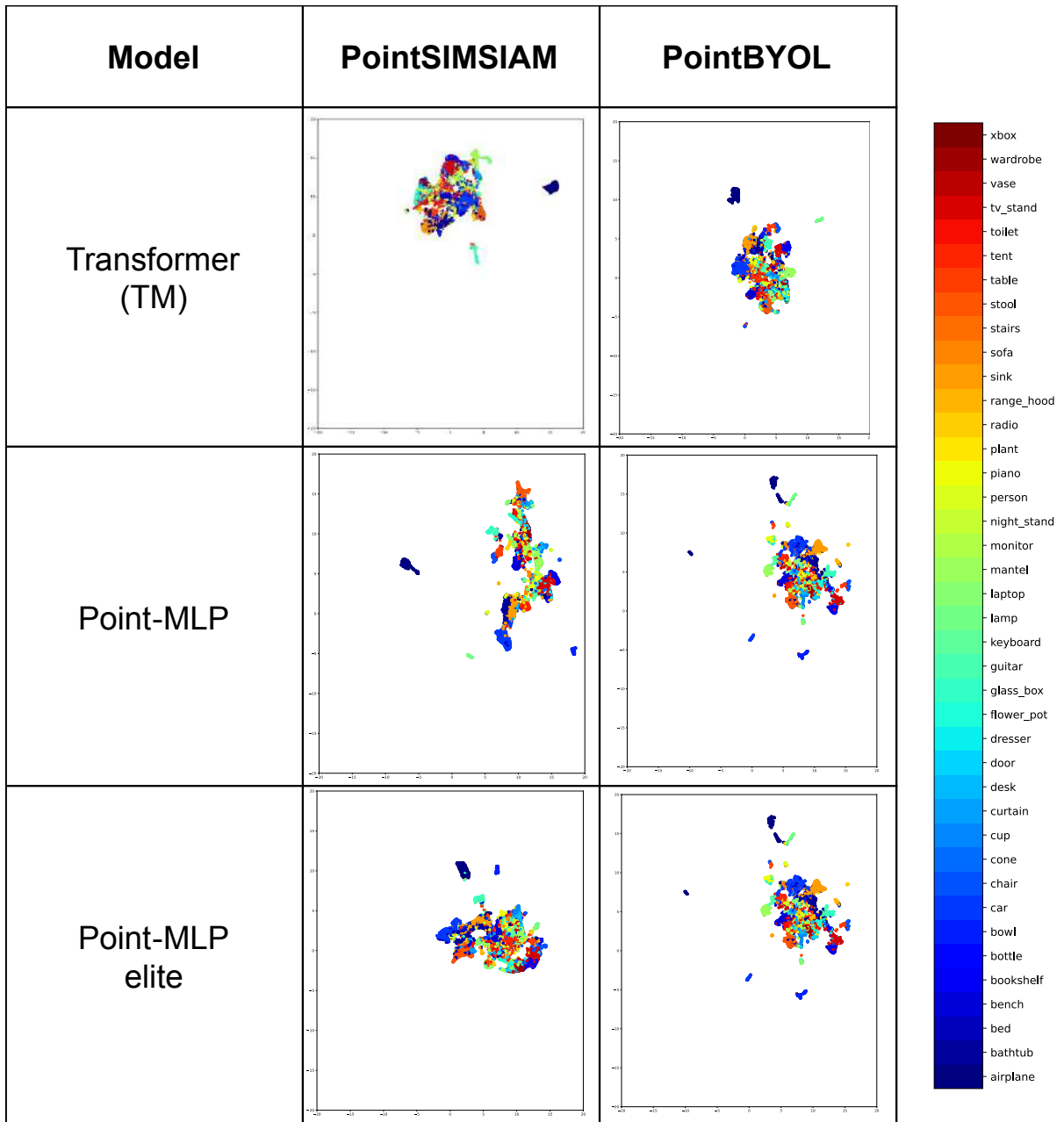


Figure 8.4: **Siamese networks UMAP results part 2**. UMAP is applied to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from  $[-20, 20]$ .

### 8.1.7. Retrieval techniques

Table 8.10 shows the results of the NN and mAP techniques applied to the representations from the test subset of ModelNet40 generated by the PointBYOL with a pre-trained Point-MLP encoder model. It can be seen from the table that the performance of NN significantly exceeds that of mAP. The pre-trained model with the best results is PointSIMSIAM with a Point-MLP encoder, which achieves 78.8% in the NN metric and 45.1% in the mAP metric. This results suggest that the model is effective at identifying the nearest neighbor to a given data point within the feature space. For a single example or query, the model is able to accurately identify the most similar example from its training set.

Conversely, a lower performance in mAP, implies that while the model may be proficient in identifying the most similar case (as evidenced by NN), it is not as efficient or accurate in ranking a broader set of relevant examples in terms of similarity or relevance. mAP considers not only the accuracy in identifying relevant examples, but also how those examples are ranked. A low mAP indicates that the model finds relevant examples, but does not necessarily rank them effectively.

According to these results, the model may exhibit limited generalization, suggesting that it could be overfitted to specific data characteristics. This allows for accurate nearest neighbor identification but fails to generalize this capability across a wider data set. Furthermore, the feature representations learned by the model might be effective in identifying direct similarities but less so in capturing a range of more subtle or complex similarities needed for high mAP ranking. Another possibility is that the model might be biased towards certain types of data or characteristics, resulting in accurate nearest neighbor identification but poor overall relevance ranking.

A thorough analysis of Figure 8.5 reveals the model’s proficient ability to discern common features among similar objects. Although there are instances of misclassification, such as incorrectly associating ‘bed’ more closely with ‘sofa’ and ‘desk’ than with its own category, it is understandable considering that this particular model of ‘bed’ might be significantly different from others in the dataset. A similar phenomenon is observed with ‘bookshelf’, ‘wardrobe’, and ‘tv\_stand’, items that are often conflated even in everyday language.

In summary, the results of high NN but low mAP in retrieval tasks suggest that the model is likely to be very good at identifying the most similar example for a specific query, but less able to effectively classify a wider range of relevant examples. This suggests areas where the model may need improvement, such as generalizing its learning capabilities or developing more robust feature representations.

Table 8.10: **Retrieval results.** ‘NN (%)’ denotes the percentage of instances where the model accurately identified the nearest neighbor. ‘mAP (%)’ signifies the model’s mean average precision in its retrieval tasks across various levels of recall.

Method	Backbone	NN (%)	mAP (%)
PointSIMSIAM	PointNet	60.4	26.7
PointBYOL	PointNet	73.8	31.2
PointSIMSIAM	PointNet(w/o ST)	76.8	36.9
PointBYOL	PointNet(w/o ST)	78.2	38.2
PointSIMSIAM	DGCNN	74.8	39.4
PointBYOL	DGCNN	75.2	39.2
PointSIMSIAM	Transformer	77.1	41.1
PointBYOL	Transformer	77.2	38.3
PointSIMSIAM	Transformer(TM)	73.7	37.9
PointBYOL	Transformer(TM)	68.1	32.2
PointSIMSIAM	Point-MLP	<b>78.8</b>	<b>45.1</b>
PointBYOL	Point-MLP	77.5	37.3
PointSIMSIAM	Point-MLP elite	77.2	42.8
PointBYOL	Point-MLP elite	77.9	39.6

Additionally, the results of the 3D object retrieval can be visually examined in Figure 8.5. This figure shows randomly selected objects from the ModelNet dataset next to point clouds that, according to the PointSIMSIAM with Point-MLP pre-trained model, have the most similar representations to the selected figures. In this case, similarity is determined based on the cosine similarity of the generated feature vectors.

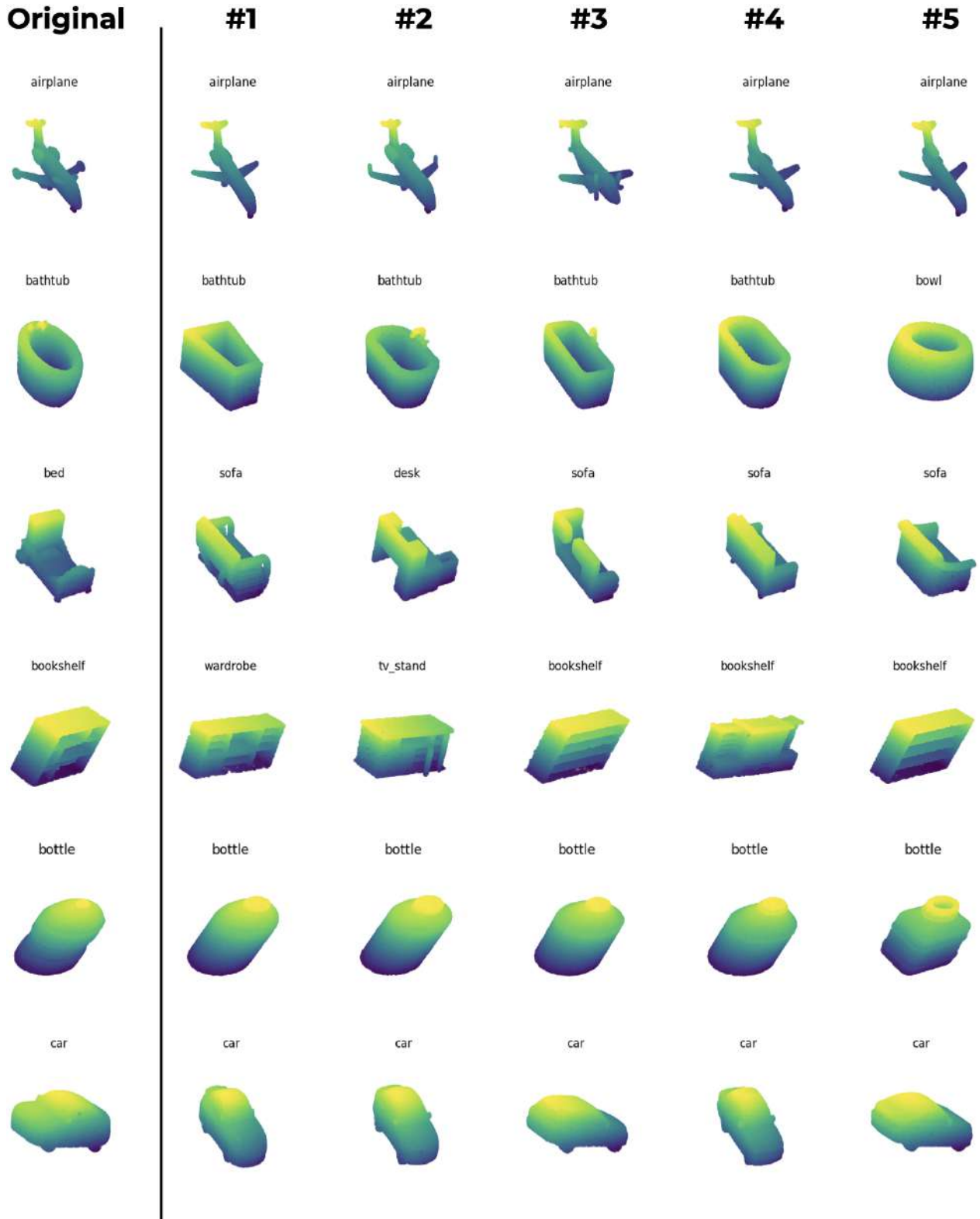


Figure 8.5: **Visual Representation of retrieval.** Objects randomly selected from ModelNet are displayed alongside their 5 nearest elements in the feature space, according to output from PointSIMSIAM with PointMLP encoder.



## 8.2. Formula-driven supervised learning

Unlike the previous experiments, the formula-driven supervised learning approach focuses on improving self-supervised learning methods by changing the dataset used, rather than the model or methodology. To evaluate the results of this approach, all pre-training and model evaluation parameters are kept constant, with the dataset being the only varying factor.

To fairly compare the added value of pre-training with SimpleShape on 3D point cloud networks to pre-training with ShapeNet55, Point-MAE [23] was used as the deep learning model. Since Point-MAE has been shown to improve results on 3D point cloud networks when pre-trained with self-supervised learning on ShapeNet55, it effectively assesses the impact of SimpleShape. The results in each benchmark were compared when Point-MAE’s Transformer was pre-trained on the SimpleShape1K, SimpleShape5K, SimpleShape10K, SimpleShape15K, SimpleShape50K, and SimpleShape100K datasets.

The results were compared with those published by Pang, et al. [23] and with the Transformer encoder [86] trained on the same datasets using labels but without pre-training. In addition, the official implementation of Point-MAE was followed to obtain results in each subsequent task, reported as ‘Point-MAE (ours)’.

In this section the models are evaluated under the linear probing metric (8.2.1). Then the results are reported under the standard downstream tasks metrics (8.2.2), which are analyzed as a whole in its last subsection (8.2.2.5). Finally, the results and analysis of dimensionality reduction evaluation (8.2.3) and retrieval (8.2.4) are presented.

### 8.2.1. Linear probing

The linear probing results are detailed in Table 8.11, where it is observed that the representations generated by Point-MAE, after pre-training with SimpleShape, facilitate the accurate classification of a significant portion of the samples by *SVM* and *KNN*.

The most remarkable results are achieved with SimpleShape10K, which reaches an accuracy of 88.2% with linear *SVM* and 81.7% with the *KNN* classifier. Beyond this point, accuracy decreases progressively as the number of SimpleShape samples is increased or decreased. Compared to pre-training with ShapeNet, the latter shows better results with linear *SVM*, reaching an accuracy of 91.0%, but with *KNN*, pre-training with SimpleShape10K outperforms ShapeNet, reaching an accuracy of 81.7%.

### 8.2.2. Downstream tasks

#### 8.2.2.1. Classification on ModelNet40

Regarding the fine-tuning evaluations for ModelNet40 classification shown in Table 8.12, SimpleShape5K outperforms pre-training with ShapeNet55 in joint accuracy, but does not

reach its level when using the voting method. The best voting accuracy achieved with pre-training on SimpleShape5K is 92.99%, while Point-MAE achieved 93.8% with pre-training on ShapeNet55. However, the accuracy of the Transformer model trained from scratch with labels is lower than all results obtained with SimpleShape, highlighting Point-MAE’s ability to improve its encoder’s results even with a different dataset than ShapeNet55.

Table 8.11: Linear evaluation on ModelNet40 [11] by SVM and kNN.

Method	Pre-training dataset	SVM Acc. (%)	kNN Acc. (%)
Point-MAE	ShapeNet55	<b>91.0</b>	
Point-MAE (our)	ShapeNet55	90.0	<b>76.9</b>
Point-MAE	SimpleShape1K	56.7	61.8
Point-MAE	SimpleShape5K	87.9	80.6
Point-MAE	SimpleShape10K	<b>88.2</b>	<b>81.7</b>
Point-MAE	SimpleShape15K	87.2	80.8
Point-MAE	SimpleShape50K	87.0	80.1
Point-MAE	SimpleShape100K	86.0	78.8

Table 8.12: Shape classification on ModelNet40: ‘Acc (%)’ denotes overall accuracy and ‘Voting Acc (%)’ indicates voting accuracy.

Method	Pre-training dataset	Acc. (%)	Voting Acc. (%)
Transformer	<i>from scratch</i>		91.4
Point-MAE	ShapeNet55		<b>93.8</b>
Point-MAE (our)	ShapeNet55	<b>92.7</b>	93.4
Point-MAE	SimpleShape1K	92.4	92.9
Point-MAE	SimpleShape5K	<b>92.7</b>	<b>93.0</b>
Point-MAE	SimpleShape10K	<b>92.7</b>	92.8
Point-MAE	SimpleShape15K	92.4	92.7
Point-MAE	SimpleShape50K	92.3	92.9
Point-MAE	SimpleShape100K	92.1	92.8

Table 8.13: Few-shot classification results on ModelNet40. Mean accuracy (%) and standard deviation (%) from 10 independent experiments are reported.

Method	Pre-training dataset	5w10s	5w20s	10w10s	10w20s
Transformer	<i>from scratch</i>	87.8 $\pm$ 5.2	93.3 $\pm$ 4.3	84.6 $\pm$ 5.5	89.4 $\pm$ 6.3
Point-MAE	ShapeNet55	96.3 $\pm$ 2.5	97.8 $\pm$ 1.8	92.6 $\pm$ 4.1	95.0 $\pm$ 3.0
Point-MAE (our)	ShapeNet55	<b>97.1 <math>\pm</math> 2.2</b>	<b>98.3 <math>\pm</math> 1.3</b>	<b>93.1 <math>\pm</math> 3.8</b>	<b>95.1 <math>\pm</math> 3.3</b>
Point-MAE	SimpleShape1K	91.7 $\pm$ 3.1	94.6 $\pm$ 4.0	87.2 $\pm$ 6.6	90.8 $\pm$ 5.2
Point-MAE	SimpleShape5K	95.0 $\pm$ 2.2	97.2 $\pm$ 2.0	89.5 $\pm$ 5.0	92.9 $\pm$ 4.3
Point-MAE	SimpleShape10K	<b>95.5 <math>\pm</math> 2.3</b>	97.1 $\pm$ 2.0	<b>91.0 <math>\pm</math> 4.8</b>	<b>93.3 <math>\pm</math> 3.3</b>
Point-MAE	SimpleShape15K	95.5 $\pm$ 2.4	97.3 $\pm$ 1.7	90.5 $\pm$ 4.9	93.2 $\pm$ 4.2
Point-MAE	SimpleShape50K	95.0 $\pm$ 2.8	97.4 $\pm$ 2.6	90.8 $\pm$ 4.8	93.3 $\pm$ 4.0
Point-MAE	SimpleShape100K	94.8 $\pm$ 2.7	<b>97.6 <math>\pm</math> 2.0</b>	90.5 $\pm$ 4.8	93.3 $\pm$ 4.5

### 8.2.2.2. Classification on ScanObjectNN

Continuing with downstream task benchmarks, the performance of Point-MAE on ScanObjectNN after pre-training with different versions of SimpleShape was also evaluated, with results presented in Table 8.14. Similar to the ModelNet40 classification results, it does not reach the levels of Point-MAE, but a significant improvement is observed compared to the Transformer trained from scratch. In this case, unlike the previous benchmarks, the best results are achieved with the SimpleShape100K dataset, with 87.26%, 86.57%, and 81.85% in the OBJ-ONLY, OBJ-BG, and PB-T50-RS tests, respectively.

Table 8.14: Accuracy (%) results from fine-tuning on ScanObjectNN classification.

Method	Pre-training dataset	OBJ-ONLY	OBJ-BG	PB-T50-RS
Transformer	<i>from scratch</i>	79.86	80.55	77.24
Point-MAE	ShapeNet55	<b>90.02</b>	<b>88.29</b>	<b>85.18</b>
Point-MAE (our)	ShapeNet55	88.30	87.60	83.07
Point-MAE	SimpleShape1K	85.71	85.20	80.22
Point-MAE	SimpleShape5K	85.20	86.23	81.23
Point-MAE	SimpleShape10K	<b>86.74</b>	<b>86.91</b>	<b>82.69</b>
Point-MAE	SimpleShape15K	85.71	86.05	81.89
Point-MAE	SimpleShape50K	86.57	85.54	81.99
Point-MAE	SimpleShape100K	87.26	86.57	81.85

### 8.2.2.3. Few-shot classification

Concluding the downstream classification tasks, the results of pre-training with SimpleShape in the few-shot classification task were evaluated. These are detailed in the Table 8.13. Here, the best results are spread between pre-training with SimpleShape5K to SimpleShape100K, showing a significant improvement of the Transformer’s performance compared

to training from scratch, although not surpassing the results obtained when pre-training with ShapeNet55.

#### 8.2.2.4. Point Cloud segmentation

To conclude the evaluation of the downstream tasks, Table 8.15 shows the results of the point MAE in the segmentation task. While the results are very similar to those obtained with pre-training on ShapeNet55, they are not quite the same. The best performance is achieved by pre-training with SimpleShape100K, reaching 86.0  $mIoU_I$  compared to the 86.1  $mIoU_I$  achieved with ShapeNet55 pre-training.

Table 8.15: Part segmentation on ShapeNetPart [61], comparing each Siamese network model with its backbone results when trained from scratch. ‘ $mIoU_I$ ’ denotes the model’s mean IoU across all instances in the dataset. The results are categorized based on autoencoder models, contrastive models, and proposed models.

Method	Pre-training dataset	$mIoU_I$
<i>From scratch</i>	Transformer	85.1
Point-MAE	ShapeNet55	<b>86.1</b>
Point-MAE (our)	ShapeNet55	<b>86.1</b>
Point-MAE	SimpleShape1K	85.6
Point-MAE	SimpleShape5K	85.6
Point-MAE	SimpleShape10K	85.9
Point-MAE	SimpleShape15K	85.9
Point-MAE	SimpleShape50K	85.6
Point-MAE	SimpleShape100K	<b>86.0</b>

#### 8.2.2.5. Downstream tasks results analysis

For few-shot learning on ModelNet40, ScanObjectNN classification, and ShapeNetPart segmentation, shown in the tables 8.13, 8.14, and 8.15, pre-training with SimpleShape again produces results inferior to those of ShapeNet55, but superior to training with labels from scratch, even on a dataset generated from only 1000 samples. This suggests that while ShapeNet55 may contain objects with greater variability and feature richness, the difference in results with SimpleShape is not as pronounced as when using the model without pre-training.

Analyzing the performance of Point-MAE with different sizes of the SimpleShape dataset, it is evident that in most cases the most significant performance difference occurs between pre-training with SimpleShape1K and SimpleShape5K. On the other hand, the best performance results are obtained with different numbers of samples in the dataset: for ModelNet40 it is 5K, for ScanObjectNN it is 10K and 100K, for few-shot learning it is 10K, 50K and 100K, and for segmentation it is 100K. This could be due to the fact that increasing the number of

samples in the pre-training set does not directly affect the results of the subsequent tasks, as long as a minimum number of figures is reached.

Once all datasets, except SimpleShape1K, reach this minimum number of figures necessary for the Transformer model to learn through masked autoencoders, the difference in using one dataset over another becomes minimal. An interesting finding is the increase in  $mIoU_I$  with SimpleShape100K in the Table 8.15. This improved performance in segmentation may indicate that the number of samples is important for improving performance. It would be worthwhile to continue this experiment in the future and test with a version of SimpleShape containing 500K or more elements.

From these results, it can be concluded that SimpleShape is sufficient to achieve high levels of performance in downstream tasks, suggesting that using ShapeNet55 as a standard for pre-training is not necessarily the best option. This dataset is used under the assumption that it provides enough features to teach a deep learning model to discriminate between classes and identify the most relevant features of each 3D model. However, because ShapeNet55 was created from CAD models, it has limited variability, well-defined edges, and a lack of randomness in its features.

Nevertheless, based on these experiments, ShapeNet55 remains the better choice for pre-training Point-MAE for classification and segmentation tasks, especially if performance is the main goal. However, SimpleShape offers other advantages, such as avoiding copyright issues, and its 5000-sample version is sufficient to achieve competitive results with less than 10% of the total samples of ShapeNet55.

### 8.2.3. Dimensionality reduction evaluation

Similar to the dimension reduction experiments performed for PointSIMSIAM and Point-BYOL in section 8.1.6, the quality of the representations generated by Point-MAE can be assessed after their pretraining on the SimpleShape dataset. Following the same methodology, after pre-training on SimpleShape1K, 5K, 10K, 15K, 50K, and 100K, the encoder is used to create embeddings for each model from ModelNet40. These embeddings are then used to generate two-dimensional representations using t-SNE and UMAP.

The results are shown in Figure 8.6 and 8.7 for t-SNE and Figure 8.8 and 8.9 for UMAP. It is immediately apparent that, similar to PointSIMSIAM and PointBYOL, t-SNE allows a clearer differentiation of the dataset’s class clusters. Although the results are not perfect, there are clear class clusters in the figures.

The results obtained with t-SNE, as shown in Figures 8.6 and 8.7, show cluster groupings with considerable distances between them, but with intermediate elements that do not clearly define any particular class. This is with the exception of the pre-training with SimpleShape1K, which confirms previous analyzes where Point-MAE pre-trained with this dataset gave inferior results compared to the others. Given that the only variable among the datasets is the number of figures that make up the generated dataset, it can be inferred that one thou-

and figures is not enough to pretrain such a network.

At the same time, it is assumed that the images generated by the model are indicative of its ability to discriminate object classes in datasets different from the pretraining dataset. This would be considered successful pretraining, suggesting that SimpleShape has potential for pretraining such networks if further explored for this purpose.

UMAP, on the other hand, presents less striking results in Figures 8.8 and 8.9. SimpleShape1K fails to distinguish any class and ends up with a single cluster combining all elements of the dataset. The other variants of SimpleShape manage to distinguish small clusters from the main group, but these separations are not as clear as with t-SNE, and none is clearly superior to the others.

While it might be expected that increasing the number of samples in the pre-training dataset would improve performance, this is not observed in these cases. Therefore, even though it is possible to create a dataset with 1 million points, this alone is not sufficient. It becomes necessary to explore other parameters of the dataset during its creation, such as the variability among its figures and the functions used to create the figures.

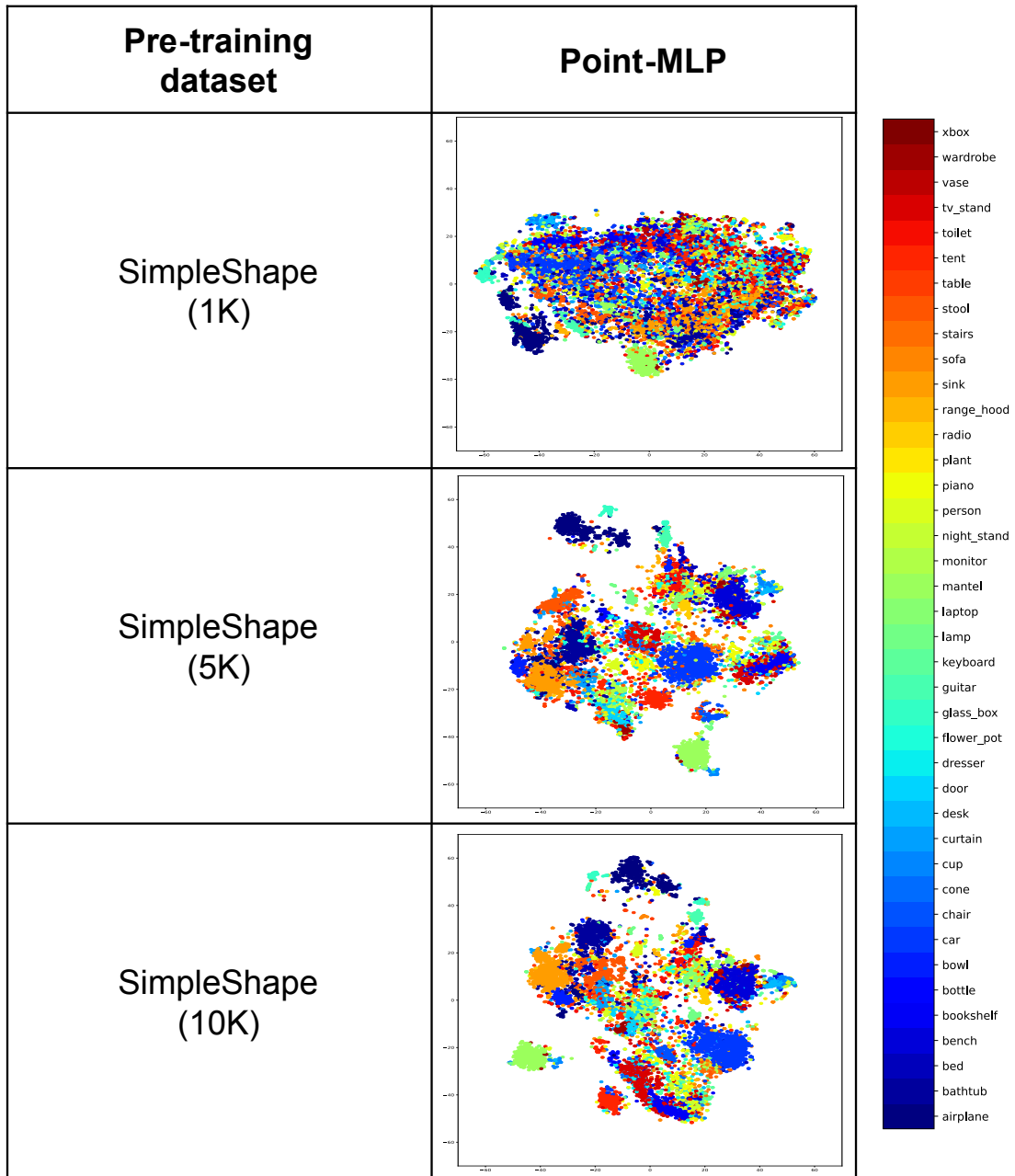


Figure 8.6: **SimpleShape t-SNE results part 1.** t-SNE is used to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from  $[-70, 70]$ .

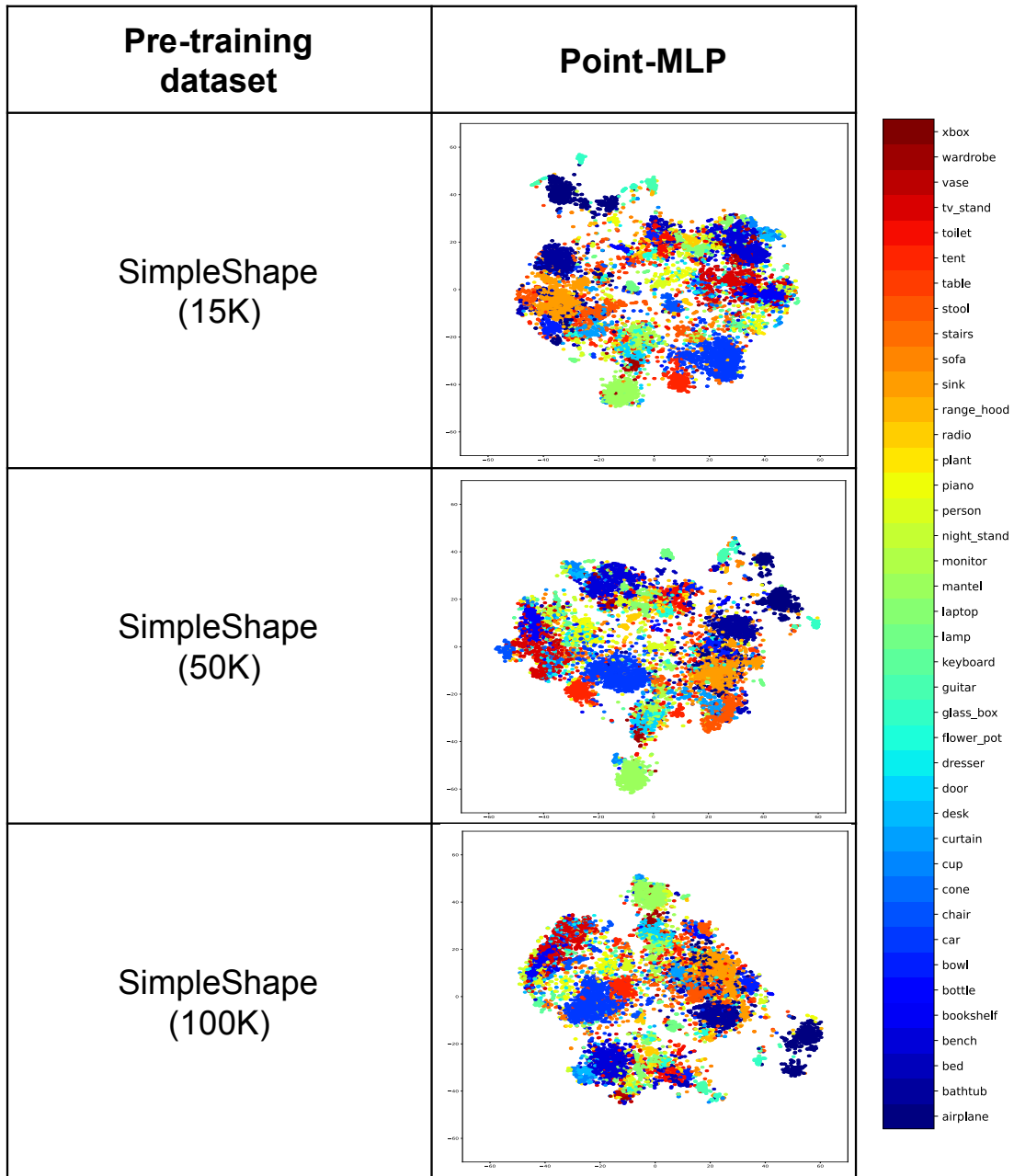


Figure 8.7: **SimpleShape t-SNE results part 2.** t-SNE is used to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from  $[-70, 70]$ .



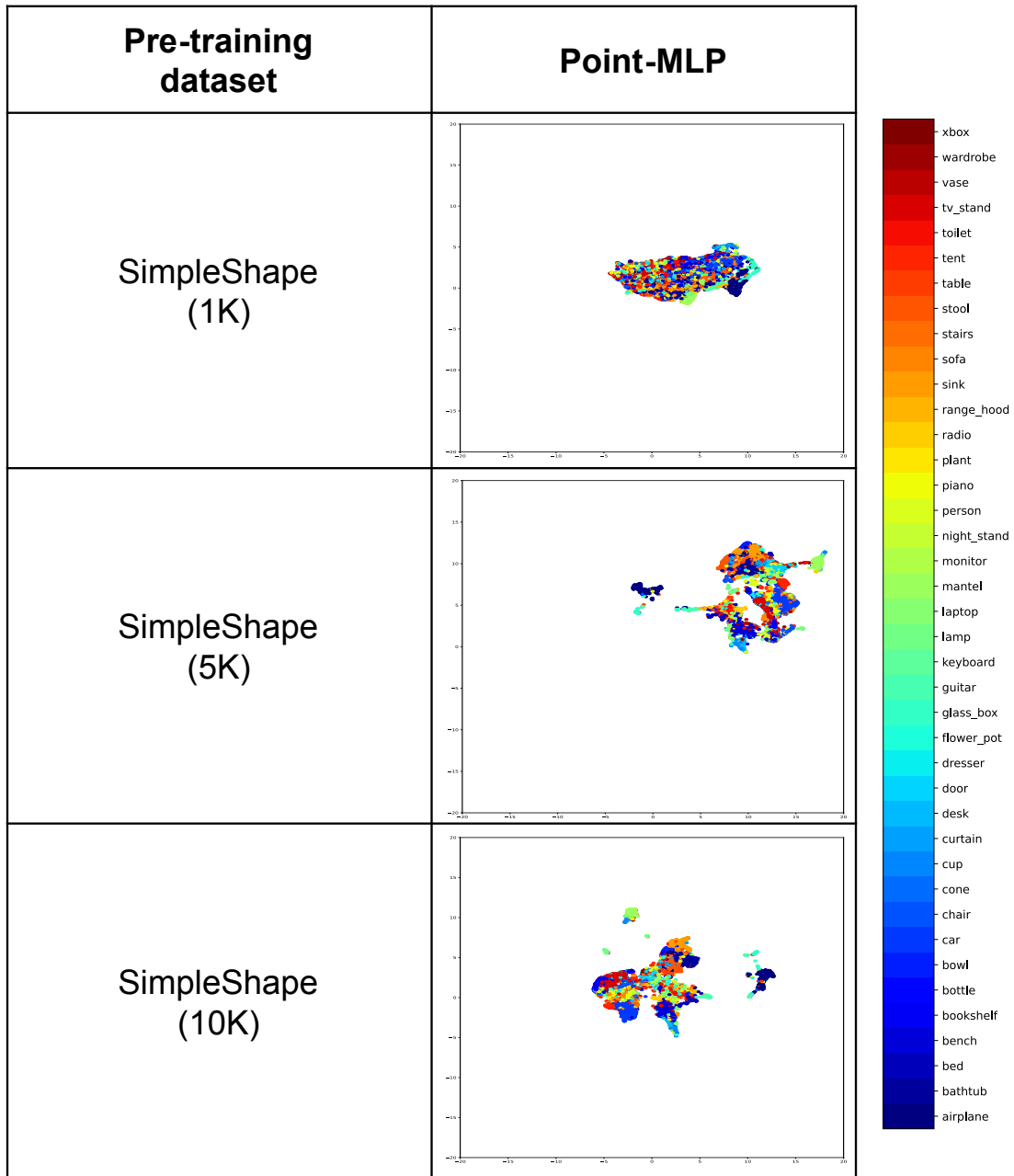


Figure 8.8: **SimpleShape UMAP results part 1**. UMAP is applied to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from  $[-20, 20]$ .

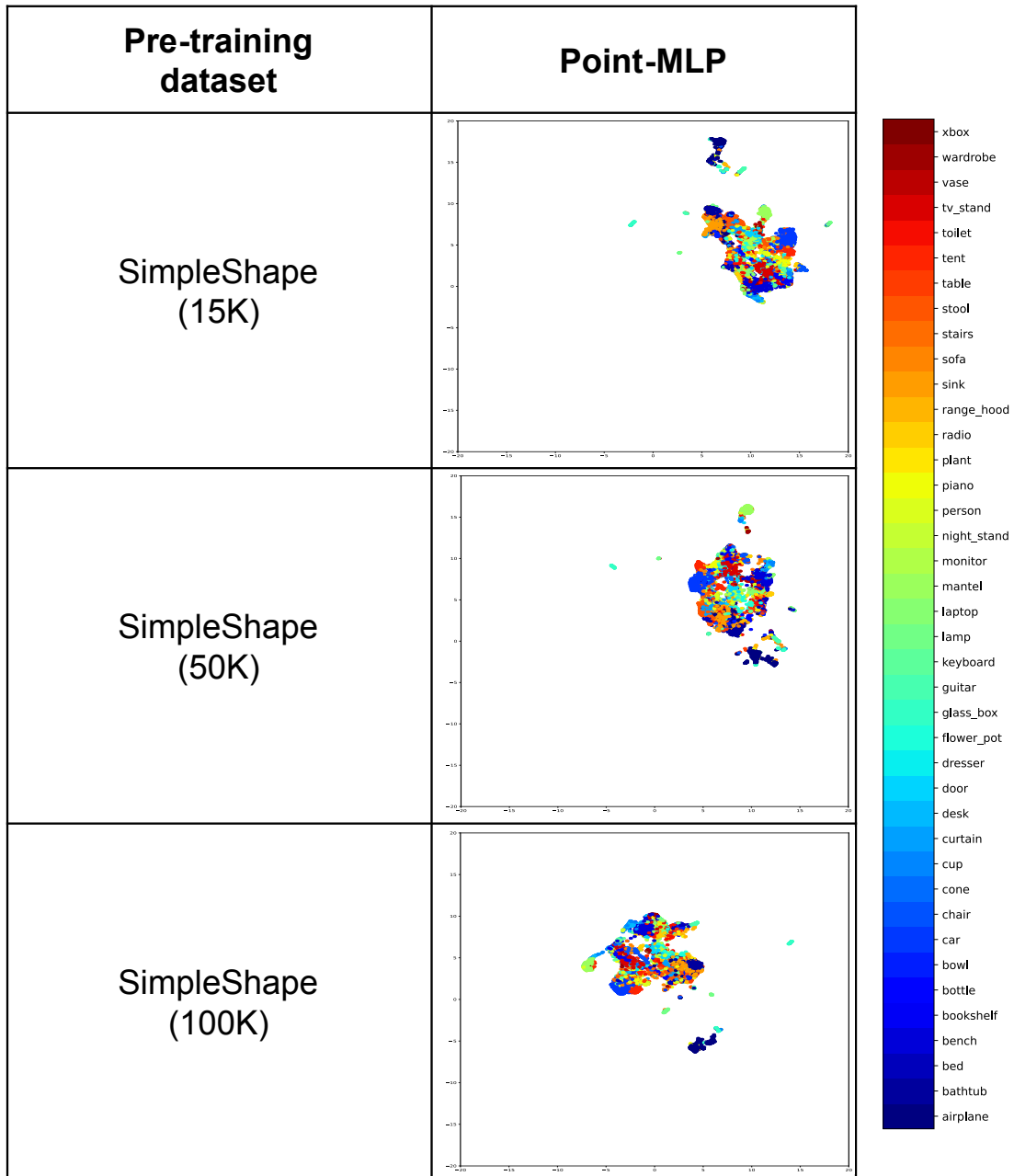


Figure 8.9: **SimpleShape UMAP results part 2**. UMAP is applied to project the representations generated by pre-trained encoders into 2D. In all images, the x and y axes range from  $[-20, 20]$ .

## 8.2.4. Retrieval techniques

Continuing with the experiments performed, Table 8.16 shows that the SimpleShape15K and SimpleShape50K versions even surpass the results obtained by Point-MAE when pre-trained with ShapeNet55. It is noticeable that SimpleShape1K delivers significantly lower results than the others due to its limited number of samples. While there is a peak in performance between the 15K and 50K variants, the 100K sample variant shows a drop in performance, which could be attributed to several factors, including possible overfitting on SimpleShape, which contains simple figures.

Similar to the Siamese network results, the model may have limited generalization. This is due to its high performance in finding neighbors with NN compared to the results in mAP. However, it can be observed that ShapeNet55 also detects this difference between the benchmarks. This indicates that the results obtained by pre-training with ShapeNet55 could be equal or even worse than those of SimpleShape. The superior results obtained in all standard benchmarks, where ShapeNet55 outperforms SimpleShape in all its variants, can be attributed more to the great adaptability of the Transformer model to new datasets than to the type of dataset used for pretraining.

Furthermore, it is crucial to highlight the efficiency gains achieved by having fewer samples in the training process. Reducing the number of samples directly translates into faster computation, as the model needs to process a smaller dataset. This approach not only conserves computational resources, but also speeds up the overall training and retrieval process. Despite this accelerated methodology, the accuracy of the model remains robust, as demonstrated by its ability to distinguish between closely related object classes in Figure 8.10. The occurrence of small classification errors, such as in the sink and car models, does not affect the overall efficiency and effectiveness of the model’s performance. By balancing the amount of training data with the need for computational speed, the model provides a pragmatic solution for large-scale object recognition tasks.

Table 8.16: **SimpleShape Retrieval results.** ‘NN (%)’ denotes the percentage of instances where the model correctly identified the nearest neighbor. ‘mAP (%)’ denotes the average precision of the model in its retrieval tasks at different recall levels.

Method	Pre-training dataset	NN (%)	mAP (%)
Point-MAE	ShapeNet55	72.2	33.3
Point-MAE	SimpleShape1K	47.4	18.5
Point-MAE	SimpleShape5K	73.3	36.5
Point-MAE	SimpleShape10K	72.6	37.3
Point-MAE	SimpleShape15K	<b>73.5</b>	36.8
Point-MAE	SimpleShape50K	72.2	<b>37.5</b>
Point-MAE	SimpleShape100K	69.8	33.9

The results of the 3D object retrieval can be visualized again in Figure 8.10. This figure shows the same objects selected from the ModelNet dataset as used in the experiments with PointSIMSIAM and PointBYOL in section 8.1.7. It contains, in order, the five objects whose representations are closest in cosine similarity to the feature vector of the main object according to the version pre-trained with SimpleShape15K.

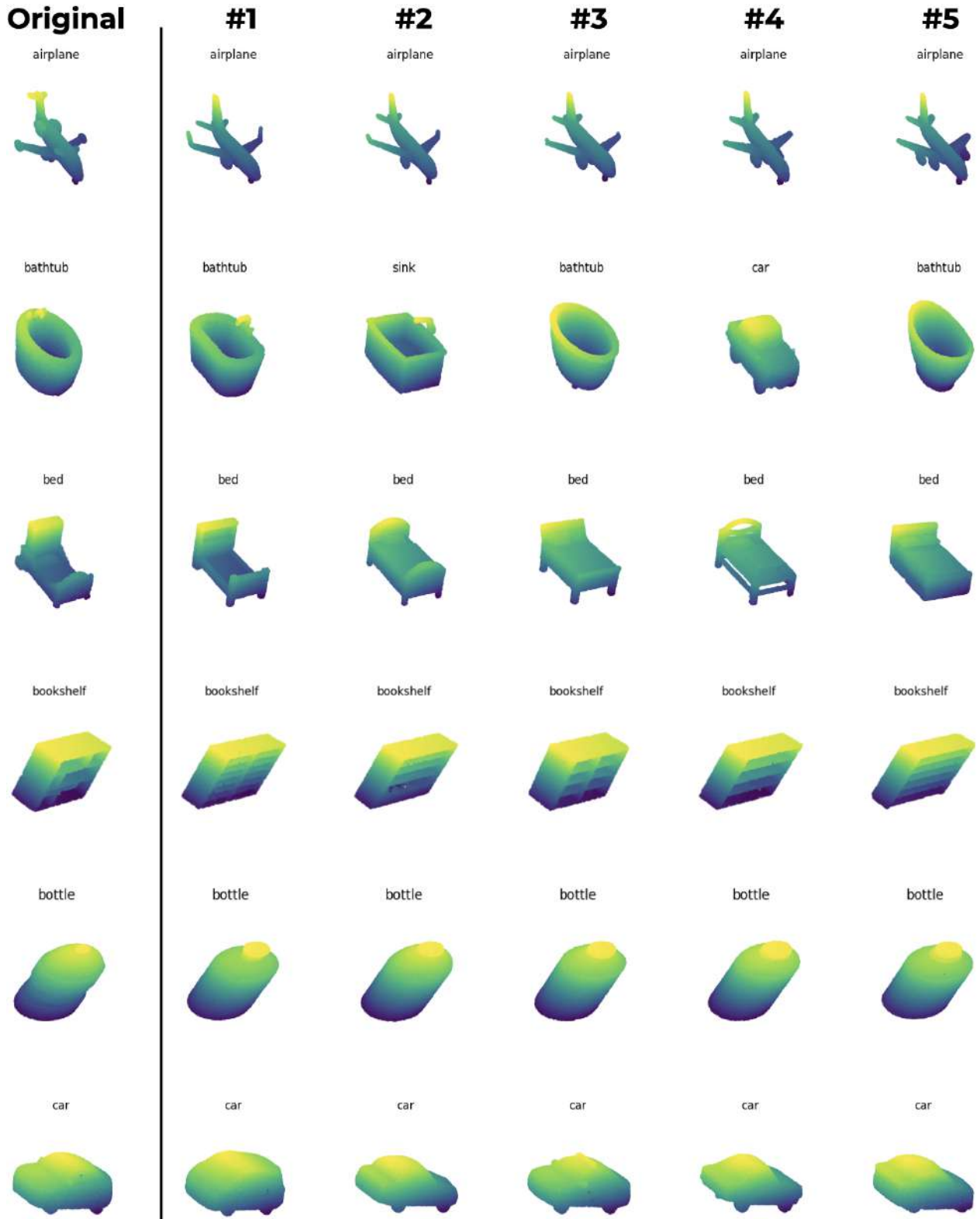


Figure 8.10: **Visual Representation of Retrieval for SimpleShape.** Objects randomly selected from ModelNet are displayed next to their 5 nearest elements in feature space, according to the output of Point-MAE pre-trained with SimpleShape15K.

# Chapter 9

## Conclusions

The experiments conducted in this research show that the Siamese network method is effective for pre-training learning models on 3D point clouds. The proposed models performance reach state-of-the-art results in self-supervised learning of 3D representations in several cases. Although masked autoencoders remain the best methodology for pre-training such networks in various tasks, contrastive learning, driven by Siamese networks, proves to be a competitive pre-training technique.

While Siamese networks did not outperform the best masked autoencoders on segmentation and classification task on ModelNet40, they achieved better results on ScanObjectNN classification, this is a dataset generated from scans of real objects, where better results were obtained, is more indicative of the real-world applications of these networks.

Another area where Siamese networks excel is in few-shot learning. Here, models that are not based on the Transformer structure, such as DGCNN and Point-MLP, outperform those that are. This performance with simpler models is consistent with the primary goal of reducing the need for large amounts of labeled data required by larger models.

In addition to verifying the feasibility and effectiveness of Siamese networks, it was observed that almost all pre-training improves the performance of the encoders used when comparing the pre-trained model with the model trained with labels only. This implies that it is always advisable to pre-train a model with Siamese networks before performing the desired downstream tasks. Even pre-training a model that has already been pre-trained with autoencoders can further improve the performance of the model in its tasks, as long as the main advantages of each pre-training strategy are effectively chosen.

Increasing research on Siamese networks in 3D point clouds could enable them to achieve performance comparable to the best current models. However, this area has not been as extensively researched as autoencoders. Although the performance is similar, Siamese networks offer the advantage of simplicity, as they do not require a large transformer-type decoder during pre-training and allow the use of smaller and less dense encoders than those required for autoencoders, achieving similar results in current benchmarks for assessing pre-training

quality.

Beside testing the effectiveness of Siamese networks for pre-training neural networks working on 3D point clouds, this work also tested the ability of artificially generated datasets to be used as pre-training datasets for this type of models.

On the one hand, the effectiveness of substituting the conventional pre-training dataset, ShapeNet55, with a synthetic dataset, SimpleShape, using a formula-driven supervised learning approach was demonstrated. This substitution extends the capabilities of models dealing with 3D point clouds. Although the results do not reach state-of-the-art levels in standard benchmarks, they show the advantages of a formula-driven method: a scalable dataset with an unlimited number of pre-training samples, free from copyright issues, and versatile enough to be tailored to specific pre-training tasks.

On the other hand, in common tasks for evaluating the quality of representations generated by pre-trained models, specifically in retrieval tasks, pre-training with SimpleShape even outperformed pre-training with ShapeNet55. This suggests that while the model naturally generates features that better distinguish objects of different classes in new datasets, the fine-tuning of the model does not fully capitalize on these acquired capabilities, resulting in lower final performance.

## 9.1. Contributions

This thesis presents innovative techniques to improve the processing and interpretation of 3D point cloud data using self-supervised learning techniques. Focusing on pre-training neural networks using Siamese network-based approaches and formula-driven supervised methods, the research aims to overcome the challenges associated with the need for large labeled datasets in 3D data analysis. The effectiveness of these techniques was evaluated through various methods, including linear probing, t-SNE, and transfer learning (fine-tuning), demonstrating their potential to improve tasks such as classification and segmentation in 3D point clouds.

The key findings of this study are Siamese networks the results obtained reach state-of-the-art self-supervised learning methods performance in the ScanObjectNN and ModelNet few-shot learning benchmarks. Furthermore, ShapeNet55 is not necessarily the best choice for a pre-training dataset. The encoder used in Transformer models is able to achieve good results even on simple datasets like SimpleShape5K, with 10% of the total number of samples that ShapeNet55 has. This suggests that the success of ShapeNet55 as a standard pre-training dataset is more related to the quality of the encoders used than to the ability of the dataset to pre-train models.

## 9.2. Achievement of goals

With respect to the goals of this thesis, the accomplishments achieved for each will be described below:

1. A variety of point cloud encoders have been identified that can be used for classification and segmentation tasks, including: PointNet, DGCNN, Point-MLP, Transformer, and their variations.
2. A set of 3D transformations was identified to allow these encoders to be pre-trained under the Siamese network strategy without collapsing in their pre-training. These transformations consist of linear data augmentation rotation, translation, clipping, and scaling, and then a random point group masking transformation is applied.
3. The ability of Siamese networks to improve performance on standard benchmarks has been evaluated. Highlighting the improvement of the state of the art in ScanObjectNNN classification and few-shot classification benchmarks.
4. The ability to generalize learning with the ShapeNet dataset in subsequent tasks was investigated. It was also compared to pre-training with the synthetic dataset SimpleShape.
5. An ablation study was performed in which different combinations of self-supervised learning components were tested in a structured manner. These included: different encoders, pre-training techniques, target tasks, performance from scratch and with pre-training, and different datasets to pre-train the networks.
6. The ability of SimpleShape to generate figures with high information content was evaluated. Although better results were obtained than without applying formula-driven supervised learning, in its current state SimpleShape performs worse on the standard benchmarks than ShapeNet as a pre-training dataset.
7. An implementation that can be easily integrated with other 3D deep learning frameworks was provided. It separates the different components mentioned in this thesis: dataset, pre-training, encoder, network header, transformations and target tasks. This allows any new research to make free use of the implementation.

## 9.3. Future work

The results of this research open new avenues of investigation that could pursue the same goal: advancing self-supervised learning methodology to reduce the need for labeled data in training 3D point clouds. Motivated by the excellent results of pre-training with Siamese networks, a potential area of exploration following this work is to investigate other types



of Siamese networks that are more complex than SIMSIAM and BYOL, such as SwAV and SimCLR.

In addition, other transformations can be targeted at specific downstream tasks, such as segmentation. One approach is to identify parts of objects before applying transformations, either by manual labeling or simpler methods such as dividing the space into subsections and assigning a label to each. Then, linear and masking transformations would be applied, shifting the current goal from having a similar global feature vector between two variants of a figure to having similar features for groups with the same labels in both transformed variants of the figure.

The third parameter considered in this thesis, in addition to the pre-training methodology and the applied transformations, is the encoder used. As point cloud research continues, it is expected that new encoders will emerge that are more efficient at extracting features from 3D figures. Since Siamese networks are agnostic to the encoder used, applying this methodology to any newly developed encoder has great potential, as most will benefit in their final results compared to training from scratch.

Regarding the study of formula-driven supervised learning on 3D point clouds, the recently published SimpleShape dataset has great potential for exploring its characteristics and effectiveness in generating feature-rich models. The main purpose of this dataset is to facilitate the training of symmetry detection in 3D objects, which imposes certain limitations that may not contribute positively to the pre-training presented in this work.

Therefore, it would be appropriate to explore the potential of SimpleShape for self-supervised learning by modifying the curve generation parameters and definitions. Currently, these are limited to specific shapes focused on symmetries. However, for Siamese networks, an approach focused on generating global and local features with high variability could improve pre-training results for generating representations of 3D point clouds.

Finally, just as this work focuses on investigating the effectiveness of Siamese networks for 3D point clouds, it also shows that the use of formula-driven supervised learning is effective for this task. Therefore, the creation of new synthetic datasets tailored for formula-driven training could lead to improved results in generating representations of 3D objects. In addition, combining different types of datasets could increase the diversity of the data used, thereby improving the quality of the pre-training of the network encoder.

# Bibliography

- [1] Dongare, A., Kharde, R., Kachare, A. D. & *et al.*, “Introduction to artificial neural network,” International Journal of Engineering and Innovative Technology (IJEIT), vol. 2, no. 1, pp. 189–194, 2012.
- [2] LeCun, Y., Bengio, Y. & Hinton, G., “Deep learning,” nature, vol. 521, no. 7553, pp. 436–444, 2015.
- [3] Li, H., “Deep learning for natural language processing: advantages and challenges,” National Science Review, 2017.
- [4] Irschara, A., Zach, C., Frahm, J.-M. & Bischof, H., “From structure-from-motion point clouds to fast location recognition,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2599–2606, IEEE, 2009.
- [5] Irschara, A., Zach, C., Frahm, J.-M. & Bischof, H., “From structure-from-motion point clouds to fast location recognition,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2599–2606, IEEE, 2009.
- [6] Pirasteh, S., Rashidi, P., Rastiveis, H., Huang, S., Zhu, Q., Liu, G., Li, Y., Li, J. & Seydipour, E., “Developing an algorithm for buildings extraction and determining changes from airborne lidar, and comparing with r-cnn method from drone images,” Remote Sensing, vol. 11, no. 11, p. 1272, 2019.
- [7] Caldera, S., Rassau, A. & Chai, D., “Review of deep learning methods in robotic grasp detection,” Multimodal Technologies and Interaction, vol. 2, no. 3, p. 57, 2018.
- [8] Yildirim, C., “A review of deep learning approaches to eeg-based classification of cybersickness in virtual reality,” in 2020 IEEE international conference on artificial intelligence and virtual reality (AIVR), pp. 351–357, IEEE, 2020.
- [9] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J. & *et al.*, “End to end learning for self-driving cars,” arXiv preprint arXiv:1604.07316, 2016.
- [10] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H. & *et al.*, “Shapenet: An information-rich 3d model repository,” arXiv preprint arXiv:1512.03012, 2015.
- [11] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. & Xiao, J., “3d shapenets:

- A deep representation for volumetric shapes,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1912–1920, 2015.
- [12] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. & Nießner, M., “Scannet: Richly-annotated 3d reconstructions of indoor scenes,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5828–5839, 2017.
- [13] Ericsson, L., Gouk, H., Loy, C. C. & Hospedales, T. M., “Self-supervised representation learning: Introduction, advances, and challenges,” IEEE Signal Processing Magazine, vol. 39, no. 3, pp. 42–62, 2022.
- [14] Liu, K., Han, X. & Chen, B. M., “Deep learning based automatic crack detection and segmentation for unmanned aerial vehicle inspections,” in 2019 IEEE international conference on robotics and biomimetics (ROBIO), pp. 381–387, IEEE, 2019.
- [15] Hess, G., Jaxing, J., Svensson, E., Hagerman, D., Petersson, C. & Svensson, L., “Masked autoencoder for self-supervised pre-training on lidar point clouds,” in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 350–359, 2023.
- [16] Krispel, G., Schinagl, D., Fruhwirth-Reisinger, C., Possegger, H. & Bischof, H., “Maeli–masked autoencoder for large-scale lidar point clouds,” arXiv preprint arXiv:2212.07207, 2022.
- [17] Yan, S., Yang, Z., Li, H., Guan, L., Kang, H., Hua, G. & Huang, Q., “Iae: Implicit autoencoder for point cloud self-supervised representation learning,” arXiv preprint arXiv:2201.00785, 2022.
- [18] Rios, T., van Stein, B., Menzel, S., Back, T., Sendhoff, B. & Wollstadt, P., “Feature visualization for 3d point cloud autoencoders,” in 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–9, IEEE, 2020.
- [19] Beetz, M., Ossenbeng-Engels, J., Banerjee, A. & Grau, V., “Predicting 3d cardiac deformations with point cloud autoencoders,” in International Workshop on Statistical Atlases and Computational Models of the Heart, pp. 219–228, Springer, 2021.
- [20] Zhang, Y., Lin, J., Li, R., Jia, K. & Zhang, L., “Point-dae: Denoising autoencoders for self-supervised point cloud learning,” arXiv preprint arXiv:2211.06841, 2022.
- [21] Hess, G., Jaxing, J., Svensson, E., Hagerman, D., Petersson, C. & Svensson, L., “Masked autoencoders for self-supervised learning on automotive point clouds,” arXiv preprint arXiv:2207.00531, 2022.
- [22] He, K., Chen, X., Xie, S., Li, Y., Dollár, P. & Girshick, R., “Masked autoencoders are scalable vision learners,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16000–16009, 2022.
- [23] Pang, Y., Wang, W., Tay, F. E., Liu, W., Tian, Y. & Yuan, L., “Masked autoencoders for point cloud self-supervised learning,” in European conference on computer vision,

pp. 604–621, Springer, 2022.

- [24] Chen, A., Zhang, K., Zhang, R., Wang, Z., Lu, Y., Guo, Y. & Zhang, S., “Pimae: Point cloud and image interactive masked autoencoders for 3d object detection,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5291–5301, 2023.
- [25] Jiang, J., Lu, X., Zhao, L., Dazaley, R. & Wang, M., “Masked autoencoders in 3d point cloud representation learning,” IEEE Transactions on Multimedia, 2023.
- [26] Lin, Z. Wang, Y., “Bev-mae: Bird’s eye view masked autoencoders for outdoor point cloud pre-training,” arXiv preprint arXiv:2212.05758, 2022.
- [27] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M. & *et al.*, “Bootstrap your own latent—a new approach to self-supervised learning,” Advances in neural information processing systems, vol. 33, pp. 21271–21284, 2020.
- [28] Chen, X. He, K., “Exploring simple siamese representation learning,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15750–15758, 2021.
- [29] Sipiran, I., Romanengo, C., Falcidieno, B., Biasotti, S., Arvanitis, G., Chen, C., Fotis, V., He, J., Lv, X., Moustakas, K. & *et al.*, “Shrec 2023: Detection of symmetries on 3d point clouds representing simple shapes,” in Eurographics Workshop on 3D Object Retrieval, pp. 17–237, The Eurographics Association, 2023.
- [30] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L., “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255, Ieee, 2009.
- [31] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. & *et al.*, “Imagenet large scale visual recognition challenge,” International journal of computer vision, vol. 115, pp. 211–252, 2015.
- [32] Krizhevsky, A., Hinton, G. & *et al.*, “Learning multiple layers of features from tiny images,” 2009.
- [33] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. & Zitnick, C. L., “Microsoft coco: Common objects in context,” in Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13, pp. 740–755, Springer, 2014.
- [34] Rothe, R., Timofte, R. & Van Gool, L., “Deep expectation of real and apparent age from a single image without facial landmarks,” International Journal of Computer Vision, vol. 126, no. 2–4, pp. 144–157, 2018.
- [35] Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, T. & Yeung, S.-K., “Revisiting point cloud classification: A new benchmark dataset and classification model on real-world

- data,” in Proceedings of the IEEE/CVF international conference on computer vision, pp. 1588–1597, 2019.
- [36] Geiger, A., Lenz, P., Stiller, C. & Urtasun, R., “Vision meets robotics: The kitti dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [37] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L. & Bennamoun, M., “Deep learning for 3d point clouds: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4338–4364, 2020.
- [38] Su, H., Maji, S., Kalogerakis, E. & Learned-Miller, E., “Multi-view convolutional neural networks for 3d shape recognition,” in Proceedings of the IEEE international conference on computer vision, pp. 945–953, 2015.
- [39] Yu, T., Meng, J. & Yuan, J., “Multi-view harmonized bilinear network for 3d object recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 186–194, 2018.
- [40] Wei, X., Yu, R. & Sun, J., “View-gcn: View-based graph convolutional network for 3d shape analysis,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1850–1859, 2020.
- [41] Maturana, D. & Scherer, S., “Voxnet: A 3d convolutional neural network for real-time object recognition,” in 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 922–928, IEEE, 2015.
- [42] Riegler, G., Osman Ulusoy, A. & Geiger, A., “Octnet: Learning deep 3d representations at high resolutions,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3577–3586, 2017.
- [43] Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y. & Tong, X., “O-cnn: Octree-based convolutional neural networks for 3d shape analysis,” *ACM Transactions On Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.
- [44] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M. & Geiger, A., “Convolutional occupancy networks,” in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pp. 523–540, Springer, 2020.
- [45] Qi, C. R., Su, H., Mo, K. & Guibas, L. J., “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 652–660, 2017.
- [46] Qi, C. R., Yi, L., Su, H. & Guibas, L. J., “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” *Advances in neural information processing systems*, vol. 30, 2017.
- [47] Ma, X., Qin, C., You, H., Ran, H. & Fu, Y., “Rethinking network design and lo-

- cal geometry in point cloud: A simple residual mlp framework,” arXiv preprint arXiv:2202.07123, 2022.
- [48] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M. & Solomon, J. M., “Dynamic graph cnn for learning on point clouds,” *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.
- [49] Zhang, K., Hao, M., Wang, J., de Silva, C. W. & Fu, C., “Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features,” arXiv preprint arXiv:1904.10014, 2019.
- [50] Yang, Y., Feng, C., Shen, Y. & Tian, D., “Foldingnet: Point cloud auto-encoder via deep grid deformation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 206–215, 2018.
- [51] Hassani, K.Haley, M., “Unsupervised multi-task feature learning on point clouds,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8160–8171, 2019.
- [52] Liu, J., Ni, B., Li, C., Yang, J. & Tian, Q., “Dynamic points agglomeration for hierarchical point sets learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7546–7555, 2019.
- [53] Xu, Q., Sun, X., Wu, C.-Y., Wang, P. & Neumann, U., “Grid-gcn for fast and scalable point cloud learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5661–5670, 2020.
- [54] Zhao, H., Jiang, L., Jia, J., Torr, P. H. & Koltun, V., “Point transformer,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 16259–16268, 2021.
- [55] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I., “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [56] Xie, S., Liu, S., Chen, Z. & Tu, Z., “Attentional shapecontextnet for point cloud recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4606–4615, 2018.
- [57] Liu, X., Han, Z., Liu, Y.-S. & Zwicker, M., “Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 8778–8785, 2019.
- [58] Yang, J., Zhang, Q., Ni, B., Li, L., Liu, J., Zhou, M. & Tian, Q., “Modeling point clouds with self-attention and gumbel subset sampling,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3323–3332, 2019.
- [59] Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S. & Teh, Y. W., “Set transformer:

- A framework for attention-based permutation-invariant neural networks,” in International conference on machine learning, pp. 3744–3753, PMLR, 2019.
- [60] Jaderberg, M., Simonyan, K., Zisserman, A. & *et al.*, “Spatial transformer networks,” Advances in neural information processing systems, vol. 28, 2015.
- [61] Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A. & Guibas, L., “A scalable active framework for region annotation in 3d shape collections,” ACM Transactions on Graphics (ToG), vol. 35, no. 6, pp. 1–12, 2016.
- [62] Ronneberger, O., Fischer, P. & Brox, T., “U-net: Convolutional networks for biomedical image segmentation,” in Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18, pp. 234–241, Springer, 2015.
- [63] Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J. & Tang, J., “Self-supervised learning: Generative or contrastive,” IEEE Transactions on Knowledge and Data Engineering, 2021.
- [64] Hinton, G. E. & Salakhutdinov, R. R., “Reducing the dimensionality of data with neural networks,” science, vol. 313, no. 5786, pp. 504–507, 2006.
- [65] Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D. & Sutskever, I., “Generative pretraining from pixels,” in International conference on machine learning, pp. 1691–1703, PMLR, 2020.
- [66] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T. & Efros, A. A., “Context encoders: Feature learning by inpainting,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2536–2544, 2016.
- [67] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A. & Bottou, L., “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.,” Journal of machine learning research, vol. 11, no. 12, 2010.
- [68] Zhang, R., Isola, P. & Efros, A. A., “Colorful image colorization,” in European conference on computer vision, pp. 649–666, Springer, 2016.
- [69] Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K., “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.
- [70] Xie, Z., Zhang, Z., Cao, Y., Lin, Y., Bao, J., Yao, Z., Dai, Q. & Hu, H., “SimMIM: A simple framework for masked image modeling,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9653–9663, 2022.
- [71] Liang, Y., Zhao, S., Yu, B., Zhang, J. & He, F., “MeshMAE: Masked autoencoders for 3d mesh data analysis,” in European Conference on Computer Vision, pp. 37–54,

Springer, 2022.

- [72] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E. & Shah, R., “Signature verification using a " siamese " time delay neural network,” *Advances in neural information processing systems*, vol. 6, 1993.
- [73] Le-Khac, P. H., Healy, G. & Smeaton, A. F., “Contrastive representation learning: A framework and review,” *IEEE Access*, vol. 8, pp. 193907–193934, 2020.
- [74] Wu, Z., Xiong, Y., Yu, S. X. & Lin, D., “Unsupervised feature learning via non-parametric instance discrimination,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.
- [75] Oord, A. v. d., Li, Y. & Vinyals, O., “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.
- [76] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A. & Bengio, Y., “Learning deep representations by mutual information estimation and maximization,” *arXiv preprint arXiv:1808.06670*, 2018.
- [77] Ye, M., Zhang, X., Yuen, P. C. & Chang, S.-F., “Unsupervised embedding learning via invariant and spreading instance feature,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6210–6219, 2019.
- [78] Chen, T., Kornblith, S., Norouzi, M. & Hinton, G., “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.
- [79] Chen, X., Fan, H., Girshick, R. & He, K., “Improved baselines with momentum contrastive learning,” *arXiv preprint arXiv:2003.04297*, 2020.
- [80] Henaff, O., “Data-efficient image recognition with contrastive predictive coding,” in *International conference on machine learning*, pp. 4182–4192, PMLR, 2020.
- [81] Chen, T., Kornblith, S., Norouzi, M. & Hinton, G., “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.
- [82] He, K., Fan, H., Wu, Y., Xie, S. & Girshick, R., “Momentum contrast for unsupervised visual representation learning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.
- [83] He, K., Zhang, X., Ren, S. & Sun, J., “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [84] Xiao, A., Huang, J., Guan, D., Zhang, X., Lu, S. & Shao, L., “Unsupervised point cloud representation learning with deep neural networks: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
- [85] Fei, B., Yang, W., Liu, L., Luo, T., Zhang, R., Li, Y. & He, Y., “Self-supervised learning



- for pre-training 3d point clouds: A survey,” arXiv preprint arXiv:2305.04691, 2023.
- [86] Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J. & Lu, J., “Point-bert: Pre-training 3d point cloud transformers with masked point modeling,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 19313–19322, 2022.
  - [87] Kamousi, P., Lazard, S., Maheshwari, A. & Wuhler, S., “Analysis of farthest point sampling for approximating geodesics in a graph,” *Computational Geometry*, vol. 57, pp. 1–7, 2016.
  - [88] Fan, H., Su, H. & Guibas, L. J., “A point set generation network for 3d object reconstruction from a single image,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 605–613, 2017.
  - [89] Zhang, R., Guo, Z., Gao, P., Fang, R., Zhao, B., Wang, D., Qiao, Y. & Li, H., “Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training,” *Advances in neural information processing systems*, vol. 35, pp. 27061–27074, 2022.
  - [90] Zhang, Y., Lin, J., Li, R., Jia, K. & Zhang, L., “Point-ma2e: Masked and affine transformed autoencoder for self-supervised point cloud learning,” 2023.
  - [91] Liu, H., Cai, M. & Lee, Y. J., “Masked discrimination for self-supervised learning on point clouds,” arXiv preprint arXiv:2203.11183, 2022.
  - [92] Sanghi, A., “Info3d: Representation learning on 3d objects using mutual information maximization and contrastive learning,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pp. 626–642, Springer, 2020.
  - [93] Gadelha, M., RoyChowdhury, A., Sharma, G., Kalogerakis, E., Cao, L., Learned-Miller, E., Wang, R. & Maji, S., “Label-efficient learning on point clouds using approximate convex decompositions,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pp. 473–491, Springer, 2020.
  - [94] Du, B., Gao, X., Hu, W. & Li, X., “Self-contrastive learning with hard negative sampling for self-supervised point cloud learning,” in Proceedings of the 29th ACM International Conference on Multimedia, pp. 3133–3142, 2021.
  - [95] Wang, P.-S., Yang, Y.-Q., Zou, Q.-F., Wu, Z., Liu, Y. & Tong, X., “Unsupervised 3d learning for shape analysis via multiresolution instance discrimination,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 2773–2781, 2021.
  - [96] Xie, S., Gu, J., Guo, D., Qi, C. R., Guibas, L. & Litany, O., “Pointcontrast: Unsupervised pre-training for 3d point cloud understanding,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pp. 574–591, Springer, 2020.
  - [97] Zhang, Z., Girdhar, R., Joulin, A. & Misra, I., “Self-supervised pretraining of 3d fea-

- tures on any point-cloud,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10252–10263, 2021.
- [98] Huang, S., Xie, Y., Zhu, S.-C. & Zhu, Y., “Spatio-temporal self-supervised representation learning for 3d point clouds,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6535–6545, 2021.
- [99] Mei, G., Huang, X., Liu, J., Zhang, J. & Wu, Q., “Unsupervised point cloud pre-training via contrasting and clustering,” in 2022 IEEE International Conference on Image Processing (ICIP), pp. 66–70, IEEE, 2022.
- [100] Wu, Y., Zhang, T., Ke, W., Süssstrunk, S. & Salzmann, M., “Spatiotemporal self-supervised learning for point clouds in the wild,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5251–5260, 2023.
- [101] Hatem, A., Qian, Y. & Wang, Y., “Point-tta: Test-time adaptation for point cloud registration using multitask meta-auxiliary learning,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 16494–16504, 2023.
- [102] Zhang, Z., Bai, M. & Li, E. L., “Self-supervised pretraining for large-scale point clouds,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 37806–37821, 2022.
- [103] Kataoka, H., Okayasu, K., Matsumoto, A., Yamagata, E., Yamada, R., Inoue, N., Nakamura, A. & Satoh, Y., “Pre-training without natural images,” in Proceedings of the Asian Conference on Computer Vision, 2020.
- [104] Hirokatsu, K., Asato, M., Eisuke, Y., Ryosuke, Y., Nakamasa, I., Nakamura, A. & Yutaka, S., “Pre-training without natural images,” *International Journal of Computer Vision*, vol. 130, no. 4, pp. 990–1007, 2022.
- [105] Anderson, C. Farrell, R., “Improving fractal pre-training,” in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 1300–1309, 2022.
- [106] Kataoka, H., Hayamizu, R., Yamada, R., Nakashima, K., Takashima, S., Zhang, X., Martinez-Noriega, E. J., Inoue, N. & Yokota, R., “Replacing labeled real-image datasets with auto-generated contours,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21232–21241, 2022.
- [107] Nakashima, K., Kataoka, H., Matsumoto, A., Iwata, K., Inoue, N. & Satoh, Y., “Can vision transformers learn without natural images?,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 1990–1998, 2022.
- [108] Yamada, R., Kataoka, H., Chiba, N., Domae, Y. & Ogata, T., “Point cloud pre-training with natural 3d structures,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21283–21293, 2022.
- [109] Said, K. A. M. Jambek, A. B., “Analysis of image processing using morphological erosion and dilation,” in *Journal of Physics: Conference Series*, vol. 2071, p. 012033, IOP

Publishing, 2021.

- [110] Miller, G. A., “Wordnet: A lexical database for,” *English. Commun. ACM*, vol. 38, p. 11, 1995.
- [111] Yu, X., Rao, Y., Wang, Z., Liu, Z., Lu, J. & Zhou, J., “Pointr: Diverse point cloud completion with geometry-aware transformers,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12498–12507, 2021.
- [112] Xiao, J., Hays, J., Ehinger, K. A., Oliva, A. & Torralba, A., “Sun database: Large-scale scene recognition from abbey to zoo,” in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492, IEEE, 2010.
- [113] Shilane, P., Min, P., Kazhdan, M. & Funkhouser, T., “The princeton shape benchmark,” in *Proceedings Shape Modeling Applications, 2004.*, pp. 167–178, IEEE, 2004.
- [114] Hua, B.-S., Pham, Q.-H., Nguyen, D. T., Tran, M.-K., Yu, L.-F. & Yeung, S.-K., “Scenenn: A scene meshes dataset with annotations,” in *2016 fourth international conference on 3D vision (3DV)*, pp. 92–101, Ieee, 2016.
- [115] Sharma, C.Kaul, M., “Self-supervised few-shot learning on point clouds,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 7212–7221, 2020.
- [116] Van der Maaten, L.Hinton, G., “Visualizing data using t-sne.,” *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [117] McInnes, L., Healy, J. & Melville, J., “Umap: Uniform manifold approximation and projection for dimension reduction,” *arXiv preprint arXiv:1802.03426*, 2018.
- [118] Loshchilov, I.Hutter, F., “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.
- [119] Loshchilov, I.Hutter, F., “Sgdr: Stochastic gradient descent with warm restarts,” *arXiv preprint arXiv:1608.03983*, 2016.
- [120] Wang, D.Yang, Z.-X., “Self-supervised point cloud understanding via mask transformer and contrastive learning,” *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 184–191, 2022.
- [121] Zamorski, M., Zięba, M., Klukowski, P., Nowak, R., Kurach, K., Stokowiec, W. & Trzciniński, T., “Adversarial autoencoders for compact representations of 3d point clouds,” *Computer Vision and Image Understanding*, vol. 193, p. 102921, 2020.