Representation learning on point cloud data with deep neural networks

PhD dissertation

Maciej Zamorski

Wrocław, 2022

Contents

Acknowledgements

This work could not be possible without many people's academic, psychological, and emotional support.

I am incredibly grateful to my dear wife Asia for her constant love and support and for lending me her strength when I was lacking.

I would like to express my deepest appreciation to my supervisors and research mentors, prof. Jerzy Świątek and dr hab. Maciej Zięba, for their invaluable insight into the field of machine learning, unwavering support, constructive criticism and profound belief in my abilities.

I would also like to extend my gratitude to other scientific mentors at various stages of my research adventure for challenging my perspective and providing me with multiple opportunities for my scientific growth: dr Adam Gonczarek, dr Piotr Klukowski and dr hab. Tomasz Trzciński.

Additionally I would like to thank the co-authors I had a pleasure to work with, for pushing the boundaries of scientific discovery together: mgr Michał Augoff, dr hab. Jan Chorowski, mgr Kacper Kania, Konrad Karanowski, dr Karol Kurach, dr Rafał Nowak, dr Przemysław Spurek, mgr Wojciech Stokowiec, mgr Michał Stypułkowski, dr Michał Walczak, mgr Sebastian Winczowski, mgr Adrian Zdobylak.

Moreover I offer my gratitude for all more and less scientific discussions to inż. Piotr Gródek, dr Tomasz Konopczyński and dr Michał Koperski.

I am grateful to my family of the dearest friends for supplying me with hope and nodding politely when I have talked about my research: Monika Dąbkowska, Aleksandra Krzyżowska, Tina Krzyżowska, Mateusz Kujawa, Paweł Kuźba, Marcelina

Łoś, Michał Stypułkowski, Michał Werner, Damian Wybraniec, Adrian Zdobylak, Sabina Zimon-Werner and Tomasz Żdanuk.

Last but not least, I also wish to thank my family from both sides. In particular, I dedicate this work to the memory of my grandmother Helena, who taught me the value of hard work and good education, and was (and still is) the constant source of inspiration for life-long learning.

Part of this work was supported by the National Centre of Science (Poland) Grant No. 2020/37/B/ST6/03463.

Abstract

Machine learning with deep neural networks is one of the leading approaches to computer vision, natural language processing, automation and robotics, often achieving far superior performance compared to humans. Therefore, deep learning has already established a dominating role in artificial intelligence research, and one can expect it to be applied in other areas where human results are subpar due to the amount of data or complexity of the tasks.

This thesis presents several deep learning approaches focused on the area of 3-dimensional computer vision. While deep neural networks are widely used for image and video processing, their application to spectral data, with its non-regular forms like point clouds and meshes, is still relatively new and under-explored. This work provides a series of advancements to the representation learning field of machine learning, particularly in application to point cloud data.

The thesis has seven chapters. The first chapter describes research motivation and lists original contributions. Chapter 2 provides an overview of the fundamental concepts related to the domains of machine learning and computer vision, as well as the relevant works from the literature are presented, with attention to previous research related to this thesis contribution. Chapters 3 to 6 present the research conducted in the area of representation learning along with the empirical evaluation of the following methods: (a) Adversarial autoencoder, Hypernetwork-based and normalizing flow approaches to data density estimation of point clouds and meshes (chapter 3), (b) Autoencoder-like methods for continual learning (chapter 4), (c) Autoencoder-based models for point cloud completion (chapter 5), (d) An analysis of proposed

techniques for feature extraction from point cloud shapes (chapter 6). The last chapter offers concluding remarks with possible directions for future work.

Streszczenie

Uczenie maszynowe z wykorzystaniem głębokich sieci neuronowych jest jednym z wiodących podejść do problemów wizji komputerowej, automatycznego przetwarzania języka naturalnego, automatyki i robotyki, wykazując się skutecznością działania przewyższającą ludzkie możliwości. Z tego powodu, uczenie głębokie jest szeroko wykorzystywane w badaniach dotyczących rozwoju sztucznej inteligencji oraz w zastosowaniach praktycznych, gdzie manualnie uzyskiwane wyniki są niewystarczające z powodu ilości danych, bądź złożoności zadań.

Niniejsza rozprawa przedstawia szereg opracowanych podejść, opartych o modele uczenia głębokiego, skupione na automatycznym przetwarzaniu danych trójwymiarowych, reprezentowanych jako chmury punktów. Podczas gdy głębokie sieci neuronowe są szeroko stosowane do przetwarzania danych obrazowych i filmowych, ich zastosowanie do danych przestrzennych, (zwłaszcza o nieregularnych postaciach, jak chmury punktów czy siatki wielokątów, w odróżnieniu od tablic wokseli) jest wciąż relatywnie nową i rozwijającą się dziedziną. Ta praca przedstawia serię badań stanowiących wkład do rozwoju obszaru uczenia reprezentacji na potrzeby uczenia maszynowego, ze szczególnym skupieniem na problemach uczenia reprezentacji chmur punktów.

Rozprawa składa się z siedmiu rozdziałów. Rozdział pierwszy przedstawia motywację do podjętych badań oraz streszczają oryginalny wkład autora w rozwój dziedziny komputerowego widzenia trójwymiarowego. Rozdział drugi zawiera opis podstawowych zagadnień związanych z uczeniem maszynowym i wizją komputerową, a ponadto prezentuje najistotniejsze postępy w dziedzinie uczenia maszynowego,

ze szczególną uwagą poświęconą osiągnięciom w literaturze mieszczącej się w zakresie badawczym tej pracy. Rozdziały 3 do chapter 6 opisują oryginalny wkład autora w rozwój dziedziny uczenia maszynowego, przeprowadzone badania w zakresie uczenia reprezentacji oraz eksperymentalną ewaluację następujących podejść: (a) Metod opartych o architekturę autokoderów przeciwstawnych (ang. *adversarial autoencoders*), hipersieci (ang. *hypernetworks*) i przepływów normalizacyjnych (ang. *normalising flows*) dla celów estymacji gęstości rozkładu danych, reprezentowanych przez chmury punktów i siatki wielokątów (ang. *polygon meshes*) (rozdział 3), (b) Metod opartych o architekturę autokoderów dla problemu uczenia ciągłego (ang. *continual learning*) modeli klasyfikujących chmury punktów (rozdział 4), (c) Metod opartych o architekturę autokoderów dla problemu uzupełniania kształtu danych reprezentowanych jako chmury punktów (rozdział 5), (d) Metod łączenia reprezentacji indywidualnych punktów w celu uzyskania reprezentacji chmury punktów dla problemu klasyfikacji (rozdział 6). Ostatni rozdział wskazuje wnioski końcowe z odniesieniem do proponowanych kierunków dalszych prac badawczych.

Nomenclature

Roman symbols

a	Scalar
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix
$\mathbf{I}$	Identity matrix
A	Function
I	Indicator function
L	Loss function
P	Probability
X, Y, Z	Random variables
$\mathcal{A}$	Set / family of sets
$\mathcal{D}$	Dataset
$\mathcal{N}$	Normal distribution
$\mathcal{P}$	Point cloud
$\mathcal{U}$	Uniform distribution

$x, \mathbf{x}, \mathbf{X}, \mathcal{X}$	Input sample for a model
$y, \mathbf{y}, \mathbf{Y}, \mathcal{Y}$	Output sample for a model
$z, \mathbf{z}, \mathbf{Z}, \mathcal{Z}$	Samples from a prior distribution
$\tilde{x}, \tilde{\mathbf{x}}, \tilde{\mathbf{X}}, \tilde{\mathcal{X}}$	Model prediction of the input sample
$\tilde{y}, \tilde{\mathbf{y}}, \tilde{\mathbf{Y}}, \tilde{\mathcal{Y}}$	Model prediction of the output sample
$\tilde{z}, \tilde{\mathbf{z}}, \tilde{\mathbf{Z}}, \tilde{\mathcal{Z}}$	Samples from a posterior distribution

Greek symbols

ε	Random noise
η	Learning rate
λ	Regularisation hyperparameter
$\nabla_\theta L$	Gradient of the function L w.r.t. parameters θ
θ and φ	Parameters of a function
Ψ	Bijective (invertible) function

Other symbols

$\mathbb{E}_X$	Expected value of a random variable X
$\mathbb{H}$	Entropy
$\mathbb{N}$	The natural numbers
$\mathbb{R}$	The real numbers
$\mathbb{X}$	Data space
$\mathbb{Y}$	Class label space
$\mathbb{Z}$	Representation (latent) space
$\lvert \cdot \rvert$	Absolute value / length of a vector / cardinality of a set
$\ell^2(\cdot)$	L^2 vector norm

Superscripts & subscripts

$\mathbf{a}_i$	Variable
$\mathbf{a}^{(i)}$	i-th element of vector $\mathbf{a}$
$\mathbf{a}^{(i:j)}$	Subvector of $\mathbf{a}$ from i-th to j-th place
$\mathbf{a}^{(\mathbf{i})}$	Subvector of $\mathbf{a}$ from the indices specified by vector $\mathbf{i}$
$\mathbf{A}^{(i,:)}, \mathbf{A}^{(:,j)}$	i-th row / j-th column of matrix $\mathbf{A}$
$\mathbf{A}^{(i,j)}$	Specific element of matrix $\mathbf{A}$
$\mathbf{a}^T, \mathbf{A}^T$	Transposition of a vector or matrix
$\vec{x}, \vec{y}, \vec{z}$	Axes in the 3-dimensional Euclidean space

$\overrightarrow{xy}, \overrightarrow{xz}, \overrightarrow{yz}$	Planes in the 3-dimensional Euclidean space
F_θ	Function F parameterised with θ
$\mathbb{R}^d$	Space $\mathbb{R}$ of d dimensions

Acronyms & Abbreviations

1-NNA	1-Nearest neighbour accuracy
3dAAE	3-dimensional adversarial autoencoder
- 3dAAE-B	” - binary distributed
- 3dAAE-C	” - categorically distributed
- 3dAAE-N	” - normally distributed
AE	Autoencoder
- AAE	Adversarial Autoencoder
- VAE	Variational Autoencoder
CD	Chamfer distance
CE	Cross-entropy
CF	Catastrophic forgetting
CIF	Conditional invertible flow
CL	Continual learning
DL	Deep Learning
e.g.	Exempli gratia (for example)
EMD	Earth mover’s distance
FC	Fully-connected
GAN	Generative adversarial network
GMM	Gaussian mixture model
i.e.	Id est (that is to say)
IL	Incremental learning
JS	Jensen-Shannon
- JSD	Jensen-Shannon divergence
KL	Kullback-Leibler

- KLD	Kullback-Leibler divergence
k-NN	k-nearest neighbours
mAP	Mean average precision
MLP	Multi-layer perceptron
MMD	Minimum matching distance
NLL	Negative log-likelihood
NN	Neural network
- CNN	Convolutional neural network
- DNN	Deep neural network
PCA	Principal component analysis
RBF	Radial basis function
RCR	Random compression rehearsal
ReLU	Rectified linear unit
RL	Representation learning
r.v.	Random variable
SVM	Support vector machine
t-SNE	t-Stochastic neighbour embedding
w.r.t.	With respect to

1 Introduction

1.1 Research motivation

Rapid advancements in information technology provide the world with a rapid influx of digital data. The unprecedented growth of computer systems, everyday access to high-speed wireless internet, and ubiquitous media capture devices make up the continuous stream of information that forces the necessity of developing automation systems implementing machine learning and artificial intelligence methods for efficient data processing.

One of the most important methods are neural networks, that are used in many of the core data processing applications in the domain of computer vision, e.g., object recognition and localisation, image segmentation and retrieval, or key-points prediction for human pose and emotion detection (Duch et al. 2000; Schmidhuber 2015; Tadeusiewicz and Szaleniec 2015; Macukow 2016). However, while deep learning on image data is already a well-established (albeit far from solved) domain, machine learning methods applied to 3-dimensional data are still relatively new and under-explored, especially in the context of irregular data types such as point clouds or meshes (Y. Li et al. 2020; Singh et al. 2020). Since many industrial automation systems (and even consumer-focused ones, with a present-day availability of cheap LIDAR sensors) rely on depth estimation, the progress in processing spatial data is crucial (Khan et al. 2020).

This thesis explores applications of deep neural networks to representation learning on point cloud data. Efficient and accurate extraction of data representation is a

cornerstone of any machine learning method, as any downstream tasks (e.g., object classification or localisation) can only be as good as the provided input. Therefore, enhancements to this crucial first step of the data processing pipeline alleviate one of the critical bottlenecks of any deep learning model.

1.2 Thesis goal and contributions

As a whole, the following dissertation is purposed to develop deep learning models focused on advancing the representation learning capabilities to automate and simplify 3-dimensional point cloud processing. This objective is realised by answering the following research questions:

1.2.1 How to appropriately measure the quality of representations?

The scientific method describes a rigorous approach to empirical experimentation. Therefore the appropriate set of metrics on which the representation quality will be measured must be defined to assess whether the proposed methods are beneficial to solving a given problem.

Representations are abstract descriptions of features that characterise the data. As multi-dimensional embeddings, their quality is impossible to assess without the ground truth descriptions, that is seldom available. Therefore, to answer this research question, the representations were measured based on the performance of the simple models on the downstream tasks. Those models, such as linear support vector machine or k-neighbours classifier, received extracted embeddings as an input to minimise the impact of the model on the final results. The type of metrics used for evaluation depended on the nature of the downstream task. They included accuracy for classification, mean average precision for information retrieval, Kullback-Leibler and Jensen-Shannon divergences for probability distribution matching, as well as fidelity, coverage and the nearest neighbour accuracy for data generation task.

1.2.2 How to apply generative modelling to point cloud representation learning?

The point cloud is a set of real-valued points in a given 3-dimensional space. The unordered nature of sets causes the cloud consisting of k points to have $k!$ equivalent definitions, providing a challenge from the modelling standpoint. One of the most popular generative methods applied to point clouds is Generative Adversarial Networks (GANs) (Achlioptas et al. 2018; C.-L. Li et al. 2019). However, the GAN implementations used in point cloud literature approximate data distribution and generate point clouds from the random representation. As such, they do not offer a possibility to extract and shape the latent representation so that it can be used for downstream tasks. Therefore, applying other types of generative models that allow learning of order-invariant representations while also modelling the latent space to a given prior distribution may be beneficial to the overall quality of downstream models.

Representation learning for point cloud data has already been studied in (Achlioptas et al. 2018) through the use of the PointNet (Qi et al. 2017a)-based autoencoder. However, the application of generative modelling was restricted to the adversarial training of Wasserstein GAN (Arjovsky et al. 2017; Gulrajani et al. 2017) for approximation of the autoencoder's latent representation. To address this problem I propose *3-dimensional Adversarial Autoencoder (3dAAE)* (Zamorski et al. 2020a). This method is a novel, single-stage, end-to-end generative approach to representation learning and point cloud data generation. An adversarial autoencoder architecture is used in the proposed approach, with a feature encoder implemented as the PointNet model. Generative modelling of the data distribution allows for simultaneous data generation, feature extraction, clustering, and object interpolation. These goals are attained using only unsupervised training with reconstruction loss and prior latent distribution regularisation.

An in-depth description of the proposed solution is provided in chapter 3 with particular emphasis on section 3.3.1.

1.2.3 How to obtain compact binary representations for 3-dimensional point clouds?

Compact binary representations are essential to many real-time applications, such as simultaneous localisation and mapping or autonomous driving. Calculating the distance between two binary vectors, also known as the *Hamming distance* (Hamming 1950), requires just two operations, i.e. `popcount`[1]-ing the result of their `xor`, offering straightforward data processing and efficient information retrieval. Moreover, assuming comparable representation length, binary representations require significantly less storage than real-valued counterparts. However, obtaining accurate and compact binary representations is non-trivial due to the size restrictions of the feature space. The question entails constructing a novel solution capable of learning and extracting such embeddings.

The previous contribution described the development of an adversarial autoencoder-based solution to the problem of representation learning on point clouds. Due to being AAE-based, one of the advantages of the 3dAAE is the ability to regularise the latent space to the arbitrary distribution (as long as it is possible to sample from that distribution). The original 3dAAE model showed promising results when trained with normal and a mixture of Gaussian priors.

Thus, the natural direction for obtaining even higher rate data compression and simpler representations for point clouds required the regularisation of the feature space with binary priors. After the experimentation with using strictly-binary (Bernoulli) and easy-to-binarise (Beta with coefficients $\alpha, \beta < 1$) distributions, the proposed solution *3dAAE-Beta* utilises the Beta($\alpha = 0.01, \beta = 0.01$) prior distributions with subsequent threshold at 0.5, to ensure the decoder's ability to generate data from the binary embeddings. The final 3dAAE-Beta model can represent 3D point clouds in compact binary space (up to 100 bits) and achieves competitive results compared to models using wide and continuous representations. The model's

[1]The CPU instruction counting the number of input bits set to 1, also known as the *Hamming weight*.

representation and generation capabilities were presented by producing samples from latent interpolation and latent algebra of feature vectors.

An in-depth description of the proposed solution is provided in chapter 3 with particular emphasis on section 3.3.2.

1.2.4 How to perform unsupervised clustering of 3-dimensional point clouds?

Clustering is one of the oldest and most extensively researched tasks in the data exploration domain. Its objective is to find a structure in the set of unannotated data by discovering the groups (clusters) of similar samples, which is crucial as, due to the amount of 3-d sensors, it is possible to annotate only a tiny portion of the captured data manually. However, point clouds defined as order-less sets are a challenge for classic approaches, such as k-means (Forgy 1965; Lloyd 1982) or k-medoids (Kaufman and Rousseeuw 1990). Therefore, creating an automatic solution for organising and learning from the unlabelled point cloud data is necessary, which may be further used to improve existing supervised approaches with semi-supervised learning.

The *3dAAE-Beta* contribution described above presented the possibility of obtaining compact, binary embeddings of point cloud data. This leads to a natural follow-up question: how much can the latent space be reduced while maintaining the ability to infer some underlying knowledge structure of the data? More specifically, is it possible to create feature embeddings represented as one-hot vectors, i.e. regularise latent space with the categorical distribution?

To answer this research question, the new 3dAAE-based model, called *3dAAE-C (Categorical)* was introduced. Reducing the whole feature space to just a handful of possible values would hinder the reconstruction and generation capabilities, so the 3dAAE-C extracts two latent representations for each shape. Each of those representations is regularised with a separate discriminating module, which allows them to have different dimensionality and prior distribution regularisation, with one

of them being very compact, categorical distributions. By adding this categorical prior distribution, the 3dAAE-C learns an unsupervised clustering even when trained on the data belonging to a single class. Among the detected clusters, it can be noted that the subgroups of chairs contain characteristic features and shapes (e.g., the number and type of legs, the height etc.) that are unique to this particular group.

An in-depth description of the proposed solution is provided in chapter 3 with particular emphasis on section 3.3.3.

1.2.5 How to effectively apply generative modelling to representation learning on arbitrarily-sized point clouds?

Common generative approaches to representation learning utilise an architecture inspired by Variational (Kingma and Welling 2014) or Adversarial (Makhzani et al. 2016) Autoencoders. Training these models is done by learning a reconstruction of an input passed through encoding and decoding modules. For straightforward comparison of input and output, the decoder is often implemented as fully-connected or convolution-transposition layers and produces a fixed number of values, i.e. pixels or points. While such an approach is adequate for image data represented as a pixel grid, point clouds are inherently an approximation to a continuous surface. Therefore, one would want to sample much more points than the model was trained on for a better approximation of the object's true shape. A solution supporting the arbitrarily-sized sampling of point clouds may be provided by generative estimating of the underlying shapes' density.

The 3dAAE and its extensions successfully learned meaningful 3-dimensional shapes' representations for various downstream tasks. However, their main disadvantage was employing the fixed-size, fully-connected neural network as a generative module. This approach created the model capable of producing point clouds consisting of a constant number of points.

I answered the question of performing generative modelling on variable-sized point clouds by taking part in the development of *HyperCloud* and *Conditional Invertible*

Flow (CIF) models.

HyperCloud extends 3dAAE with two neural networks, called trainer and target network. Instead of modelling the shapes directly, the trainer network, given the feature representation as an input, learns to generate the network parameters for the target, which transforms points in 3-dimensional space. That way, the model is able to generate an arbitrary number of points and vertices.

On the other hand, CIF learns invertible transformations of points from latent to point cloud data space. Employing a reversible architecture enables optimising the model's weights with log-likelihood training. Moreover, using an additional shape embedding, CIF allows for the conditional generation of shapes.

An in-depth description of the proposed solution is provided in chapter 3 with particular emphasis on sections 3.3.4 and 3.3.5.

1.2.6 How to learn point cloud representations suitable for catastrophic forgetting mitigation?

The catastrophic forgetting phenomenon, i.e. the rapid degradation of the neural network performance on the previous data after being re-trained on the new dataset, is an active field of research for the image, video and text domains while still being a relatively under-explored area in the context of the point cloud data. This poses the question: do approaches explicitly prepared for the image data naturally extend to the 3-dimensional shapes, or does this problem require a point-cloud specific solution? Assuming the latter, what approach produces the most valuable representations for 3-d point clouds in the context of continual learning?

To answer this research question, it was necessary to adapt existing approaches in order to assess the quality of their performance on 3-dimensional data adequately. Implementation of several existing methods for 2-dimensional input (images), i.e. Learning Without Forgetting (Zhizhong Li and Hoiem 2017), Elastic Weight Consolidation (Kirkpatrick et al. 2017) and iCaRL (Rebuffi et al. 2017), resulted in unsatisfactory performance on point cloud data, measured as the classification ac-

curacy after subsequent re-trainings.

Therefore, the new approach, called *Random Compressed Rehearsal (RCR)* has been proposed. The RCR uses basic autoencoder-like architecture and shares the latent representation of the input point cloud with the classification component. During the training, the reconstruction loss serves as an additional regularisation term. Additionally, by introducing a new rehearsal technique to compress point clouds from previous tasks, there is a need to store just a small number of points from the past training datasets to mitigate the catastrophic forgetting effectively. Moreover, this approach is architecturally simple and does not require training an additional generative model, keeping a snapshot for the inference of soft labels, or introducing several additional layers to the classifier.

An in-depth description of the proposed solution is provided in chapter 4.

1.2.7 How to learn meaningful representations for the point cloud completion task?

Data restoration, an applied case of a matrix completion task, is a vital machine learning issue that has already been successfully applied to the image domain in the forms of up-scaling the resolution (Ledig et al. 2017), colourising the black and white pictures (Cheng et al. 2015) or generating the missing data (Zheng et al. 2019). Contrary to the image domain, the 3-dimensional restoration (completion) problem often requires generating an entirely new yet coherent view of input data (e.g., creating a full 360-degree point cloud from the RGB-D data). This research topic consists of creating a method that will be able to translate spatial features of the input data to generate the missing portion of the given shape.

The problem of learning representations suitable for point cloud completion tasks was attacked by comparing several modifications to 3dAAE in terms of the model's architecture, encoding backbone, and the prior distribution regularisation type.

The family of variational, adversarial, and regular (bottleneck) autoencoders was considered in terms of architecture. The models, trained by using a reconstruction

objective only, learn to generalise to samples containing too little information for perfect completion, showing a lack of learned assumptions about data belonging to a specific class or shape distribution. On the other hand, generative autoencoders showed an ability to infer the general structure of a previously-unseen shape despite seeing just the portion of the input (e.g. the chair with no legs).

The double encoding approach was applied to further increase the encoder's capability to produce holistic representation. Therefore, after the first pass through the encoder, the returned latent vector is concatenated to each 3-dimensional point vector to obtain a shape's features. The resulting multi-dimensional data structure is then passed through the additional encoding module. This method resulted in significantly improved qualitative results of shape completions.

An in-depth description of the proposed solution is provided in section 5.3.

1.2.8 What is the influence of the feature aggregation function on the overall representation quality?

PointNet (Qi et al. 2017a) performs the feature extraction of the point cloud shape by processing each point individually and aggregating the point-wise features with the permutation-invariant `max` function. However, while effective and easy to implement, such an approach only considers the extreme value of each point-wise feature distribution. Calculating global feature value based on the whole distribution may increase the quality of extracted representations.

Several proposed representation extraction strategies were considered to answer the question about the effect of the feature aggregation function. Instead of the default `max` function described in the PointNet architecture, this research examines extension to k-max values as well as single-statistic replacements (such as median, mean and sum) and combinations of these functions. Evaluations were performed quantitatively based on achieved classification accuracy, feature vector diversity and qualitatively by key set visualisations. Conducted experiments show that using combined statistics aggregation results in a performance increase in classification

accuracy regardless of the feature vectors length. Moreover, the statistic that takes the full feature's value distribution into account results in a more holistic shape representation.

An in-depth description of the proposed solution is provided in chapter 6.

1.3 Plan of this work

This dissertation is structured as follows:

- **Chapter 1** describes the goals, purpose and contributions of this work.
- **Chapter 2** provides preliminary information and gives an overview of the representation learning domain, and reviews the relevant literature.
- **Chapter 3** introduces the experimental setup.
- **Chapter 4** describes proposed methodologies for generative modelling of point clouds.
- **Chapter 5** describes research in the area of continual learning of models trained on point clouds.
- **Chapter 6** presents the application of introduced methods to point cloud completion tasks.
- **Chapter 7** contains the experimental work focused on the classification task of 3-dimensional point cloud data.
- **Chapter 8** offers a summary and points to possible directions for future work.

2 Problem background

The first part of the chapter offers the necessary background on the theoretical aspects of training machine learning models and describes the types of machine learning tasks. The following section focuses on defining the representation learning task. The chapter concludes with discussions about the most important formats of storing two and three-dimensional data and relevant literature regarding learning representations of those formats.

2.1 Machine learning

The approach to training machine learning models depends on their purpose and the availability of the data. This section describes the essential learning methods (such as supervised and unsupervised learning) and statistical approaches (i.e. discriminative and generative modelling).

2.1.1 Learning methods

Due to the rapid technological advancements, there is an enormous amount of data available. However, the vast majority of the data is provided without any description of its content, e.g. a type of object in the picture. This type of description is called a *label* of a data sample. Depending on the availability of those labels, machine learning can be divided into two groups: *supervised learning* if they are present and *unsupervised learning* otherwise.

Supervised learning assumes the existence of the dataset that contains data

samples along with corresponding data labels. We can define such dataset of n sample-label pairs as

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}, \tag{2.1}$$

where $\mathbf{x}$ denotes the vector representing data sample and y its target value.

One of the fundamental and most well-known tasks for supervised machine learning models is *classification*. Assuming the dataset as described in equation (2.1), the goal is to find a function F^* such that

$$F^* = \min_{F:\mathbb{X}\to\mathbb{Y}} \sum_{\mathbf{x},y\in\mathcal{D}} \text{dist}(F(\mathbf{x}), y), \tag{2.2}$$

where F is any function from data space $\mathbb{X}$ into class space $\mathbb{Y}$ and $\text{dist}(\cdot)$ is a comparison (distance) function. In general, the term *classification* is reserved for situations when the codomain of the function F is a set of a predetermined number of classes, i.e. $\mathbb{Y} \subset \mathbb{N}$. When targeting a real-valued codomain, i.e. $\mathbb{Y} \subseteq \mathbb{R}$, the task is called *regression*. An in-between case, named *ordinal regression* or *ordinal classification* exists when predicting an ordinal variable, i.e. when there exists an ordering between target variable values (in standard classification, the classes are assumed to be independent), e.g. when predicting the score taking values 1 – bad, 2 – neutral to 3 – good.

Unsupervised learning often focuses on finding an underlying structure of the dataset $\mathcal{D}$, which we can define as:

$$\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, \tag{2.3}$$

where $\mathbf{x}$ denotes the vector representing a single data sample and n is the number of samples.

Typical instances of unsupervised learning tasks are *clustering* – discovering groups of similar examples within the data and *density estimation* – determining the distribution of data based on the given discrete samples (Murphy 2012).

Another vital purpose of unsupervised learning is projecting the high-dimensional space into a lower-dimensional space without losing separability and information in

the process, called *dimensionality reduction* or *representation learning*. The ability to obtain meaningful features of the data that reduce the complexity of the data samples significantly reduces the curse of dimensionality effect, as described in section 2.2. Moreover, reducing the data to 2 or 3 dimensions allows for straightforward visualisation of the data space (Wold et al. 1987; Van der Maaten and Hinton 2008). As the main objective of this thesis, an in-depth description of this task is offered in section 2.2.

In addition to the approaches mentioned above, we can also specify the *semi-supervised learning* (Van Engelen and Hoos 2020) in which only a portion of a dataset is labelled and *self-supervised learning* (Jing and Tian 2020), where the labels are generated (usually on-the-fly during the training process) based on the data samples.

Another area of machine learning is *reinforcement learning* (Sutton and Barto 2018). Instead of considering the datasets of input and (if they exist) corresponding class values, reinforcement learning focuses on mapping situations to the sequence of possible decisions. Rather than being trained on the optimal set of actions, the machine learning agent tries to develop an action sequence on its own, and the only feedback the model receives is through the reward function. The reward can be an arbitrary objective describing the state of the situation, e.g. a number of points scored in the given moment of the game.

While not a distinctive area in itself, *continual learning* (Parisi et al. 2019) is an important extension of the learning methods presented above, which assume a single optimisation process on the fixed training dataset. However, when conducting multiple optimisation phases (e.g., re-training the model on new data), the performance of the model on the original dataset tends to deteriorate. The mitigation of this phenomenon, called *catastrophic forgetting* is a critical research area, intending to not only mitigate forgetting effects but also to use new data samples to enhance the model's quality on the past data (Ven and Tolias 2019).

2.1.2 Statistical approaches

From probabilistic perspective, machine learning models can be divided into *generative* and *discriminative*. This subsection describes those approaches in supervised and unsupervised scenarios.

2.1.2.1 Supervised

Let us assume X and Y to be random variables taking values from the data space $\mathbb{X}$ and the class space $\mathbb{Y}$ respectively.

In a supervised learning scenario, mapping class labels to data samples relies on estimating the joint probability $P(X, Y)$, which can be calculated using the product rule as in equation (2.4).

$$P(X, Y) = P(Y|X)P(X) = P(X|Y)P(Y) \tag{2.4}$$

By transforming equation (2.4) we can arrive at the equation (2.5), called *Bayes' theorem*.

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{P(X|Y)P(Y)}{\sum_{y\in\mathbb{Y}} P(X, Y = y)}. \tag{2.5}$$

The classification using equation (2.5) is then performed by selecting the class $\tilde{y} \in \mathbb{Y}$ with the highest probability given the data sample $\mathbf{x}$

$$\tilde{y} = \arg\max_{y\in\mathbb{Y}} P(Y = y|X = \mathbf{x}) = \arg\max_{y\in\mathbb{Y}} P(y|\mathbf{x}) \tag{2.6}$$

Note that in the equation (2.5) above, $P(X)$ does not depend on the class label and, for the classifier definition purpose, can be discarded without affecting the final result. Therefore equation (2.6) is equivalent to

$$\tilde{y} = \arg\max_{y\in\mathbb{Y}} P(X = \mathbf{x}|Y = y)P(Y = y) = \arg\max_{y\in\mathbb{Y}} P(\mathbf{x}|y)P(y). \tag{2.7}$$

The approach of modelling $P(y|\mathbf{x})$ directly is called discriminative approach, as given by equation (2.6). On the other hand generative approach models both $P(\mathbf{x}|y)$ and $P(y)$, as in the equation (2.7).

2.1.2.2 Unsupervised

Contrary to the supervised scenario, unsupervised learning assumes the case when the data annotations are not available. Therefore it is only possible to model data distribution, which will be presented as a random variable X taking values from the data space $\mathbb{X}$.

Discriminative modelling The discriminative approach to supervised learning models the conditional probability $P(Y|X)$ directly. Since the actual class space $\mathbb{Y}$ is unknown in the unsupervised setting, the standard approach is to create artificial (also known as *surrogate*) classes, which are generated based on data samples themselves. The surrogate classes generation technique depends on the structure of unlabelled data. One possibility, as presented in (Dosovitskiy et al. 2014), assigns a distinct class label for every image in the dataset. Then, additional data for each class is generated by performing data augmentation on the exemplar sample. The further procedure follows the discriminative approach of the supervised scenario.

Another strategy consists of defining the proxy objective that substitutes the main goal with the one that can be defined using only the data at hand. Examples of such auxiliary objectives are smoothness and photometric consistency in the optical flow (Jonschkowski et al. 2020) and structural similarity in the monocular depth estimation (Yinglong Feng et al. 2019).

Generative modelling While discriminative models that learn in an unsupervised way exist, the generative approach is the prevailing methodology for training on unlabelled datasets (Murphy 2022). This section will describe the deep generative approaches to data distribution $P(X)$ modelling.

Various machine learning methodologies for generative unsupervised learning have been proposed in the literature. In (Tomczak 2021) the general taxonomy of such models, depending on the approach to modelling $P(X)$, are as follows (see figure 2.1):

- **Autoregressive models** base the training process on a series of conditional

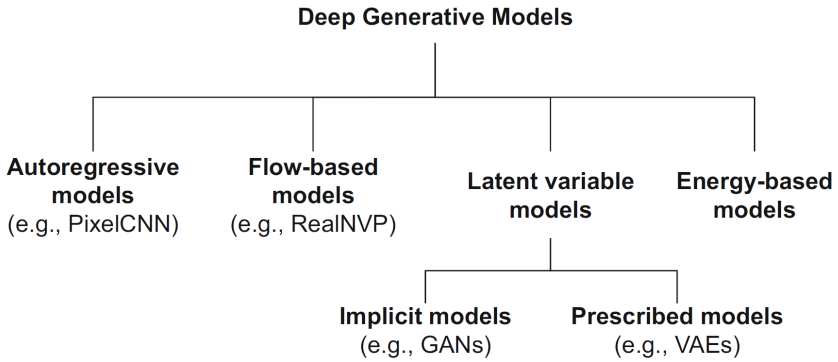

Figure 2.1: A hierarchy of deep generative models. Credit: (Tomczak 2021).

probabilities in which each consecutive piece (e.g. pixel in the image) of the data sample is conditioned on all of the preceding pieces. A probability distribution over data sample $\mathbf{x} \in \mathbb{X}$ can be defined as:

$$P(\mathbf{x}) = P(\mathbf{x}^{(1)}) \prod_{i=2}^{d} P(\mathbf{x}^{(i)} | \mathbf{x}^{(1:i-1)}) \tag{2.8}$$

- **Flow-based models** apply the change of variables formula, given as:

$$P(\mathbf{x}) = \Pi(\mathbf{z}) \left| \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right| = \Pi(\Psi^{-1}(\mathbf{x})) \left| \frac{\partial \Psi^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right|, \tag{2.9}$$

where Π – known probability distribution over $\mathbf{z}$, $\Psi : \mathbb{Z} \to \mathbb{X}$ – bijection.

Modelling the probability of the data is achieved by utilising a series of invertible transformations from data space to the latent space, described with given prior probability distribution, such as $\mathcal{N}(0, \mathbf{I})$. The flow-based model can be therefore written down as:

$$P(\mathbf{x}) = P(\mathbf{z} = \Psi(\mathbf{x})) \left\| \mathbf{J}_{\Psi(\mathbf{x})} \right\|, \tag{2.10}$$

where $\Psi = (\Psi_1 \circ \Psi_2 \circ \cdots \circ \Psi_k)$ is a composition of bijections and $\left\| \mathbf{J}_{\Psi(\mathbf{x})} \right\|$ is the absolute value of the determinant of the Jacobian of the transformation Ψ.

- **Latent variable models** aim to condition the distribution of the random variable X taking values from the higher-dimensional data space $\mathbb{X}$ on the

distribution of r.v. Z taking values from the lower-dimensional *latent space* $\mathbb{Z}$. The latent variable corresponds to underlying factors governing the distribution of the observable data. Their relationship can be summed up as follows:

$$Z \sim P_Z(Z) \tag{2.11}$$

$$X \sim P_X(X|Z) \tag{2.12}$$

- **Energy-based models** represent the data distribution as the Boltzmann distribution:

$$P(\mathbf{x}) = \frac{\exp(-E(\mathbf{x}))}{\sum_{\mathbf{x}\in\mathbb{X}} \exp(-E(\mathbf{x}))}, \tag{2.13}$$

where $E(\mathbf{x})$ – the energy function. The Energy-based model works by approximating the denominator of the equation above for and is known as the *Boltzmann machine* (Ackley et al. 1985; Hinton, Sejnowski, et al. 1986) when using an energy function $E(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{x}$ or as *restricted Boltzmann machine* (Hinton 2012) after an inclusion of latent variable to the energy function as $E(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{z}$.

2.2 Representation learning – Problem formulation

This section will describe the main objectives of representation learning. The subsequent section describes the essential feature extraction and dimensionality reduction methods. After that, the description of representation learning will focus on methodologies applied to 3-dimensional data.

Representation learning is often equated with dimensionality reduction, i.e. embedding the original, high-dimensional data into a lower-dimensional subspace with an aim to preserve its structure. While dimensionality reduction is undoubtedly an important outcome, it is only one of the effects of obtaining good data representations.

In (Bengio et al. 2013) authors define representation learning as *learning representations of the data that make it easier to extract useful information when building classifiers or other predictors.*

Depending on the context *useful* can mean different things, and the precise definition of the usefulness is most of the time challenging to formulate. In general, representation learning is performed to extract embeddings composed of salient features representing data, which will be further used for a given downstream task, e.g. classification or regression. Obtaining good representation can significantly increase the quality and simplicity of the machine learning model applied to this task.

This section will describe the desirable properties of good representations.

Disentanglement of underlying factors Almost all real-world data is based on a set of hidden, underlying variables. While they are often impossible to explicitly obtain, their existence is the core motivation for representation learning research. It is hypothesised that *an ideal representation is one in which the features within the representation correspond to the underlying causes of the observed data, with separate features or directions in feature space corresponding to different causes, so that the representation disentangles the causes from one another* (Goodfellow et al. 2016).

Therefore, one can define the *disentanglement* as the property of the representation where each feature correlates with a single latent factor describing the data. According to (Lake et al. 2017) learning of the insufficiently disentangled representations is the reason for the poor performance of state-of-the-art machine learning models on certain tasks, such as transfer of ideas in the reinforcement learning scenarios.

However, measuring the disentanglement of the representation is non-trivial. Assuming the possibility of generating data in original dimensionality based on the embedding, one could manually change the value of a single latent variable and observe the results. Nevertheless, this approach can be tedious and time-consuming,

even for very low-dimensional embeddings. Recently, the disentanglement metrics are starting to be introduced (Higgins et al. 2016; Zaidi et al. 2020), but require knowledge about the actual latent factors and therefore work only on artificially generated data.

Abstraction and transfer of features Another coveted characteristic of good representations is the ability to contain semantically meaningful features at different levels of abstraction. Constructing representations with deep architectures (such as deep neural networks) promotes the reuse of features learned on the previous layers and allows for learning higher-level data properties. Such stacking produces hierarchically organised representations, reducing the number of parameters needed to model complex data.

Moreover, learning abstract yet meaningful features allows for the straightforward transfer of obtained knowledge within the task domain (Pan and Q. Yang 2009). By first training a deep neural network on a richer but unrelated dataset, the model will learn a range of primary and intermediate features that will remain useful when fine-tuning the model to work on a much smaller dataset of interest. Training the same architecture directly on the small dataset may result in overfitting the data and, therefore, unsatisfactory performance on the actual task.

Increasing information density All the described data types are highly-dimensional representations of reality. These large representations, although faithful, pose a significant challenge from the machine learning standpoint that will be introduced on an example.

Let us assume an image of the relatively small size of 6×6 pixels. Each pixel is usually defined by three integers in the 0-255 range, corresponding to red, green and blue channel intensities. Therefore, a single pixel can represent up to $256^3 \approx 1.7 \times 10^7$ possible colours. Thus there exist over $(1.7 \times 10^7)^{36} \approx 1.2 \times 10^{260}$ unique 6×6 images. Such a vast space of possible samples poses a challenge from a machine learning standpoint and is known as a *curse of dimensionality* (Bishop 2006). When

dealing with high-dimensional data, the number of training samples we can gather will usually be insignificant in comparison to the size of the entire data space. This phenomenon reveals the necessity of learning concise and descriptive data representations to avoid the risk of overfitting the model to available training data and significantly increase the signal-to-noise ratio in the input values to the machine learning model.

2.3 Representation learning for images

Pictures have long been the most popular visual medium in the world. Since *an image is worth a thousand words*, the graphics are widely used as they offer a complementary way to communicate information. Today there are more and more devices able to capture and store image data, such as smartphones, cameras, drones and even smartwatches, so it does not come as a surprise that images are ubiquitous as a digital representation of data.

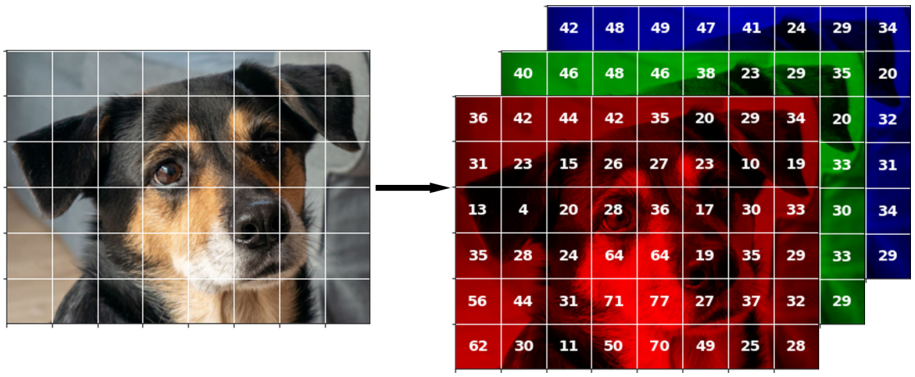

Figure 2.2: Example of an image represented as three 2-dimensional arrays corresponding to red, green and blue channels. Number values represent channel intensities for a given region.

In the digital world, graphical data may be stored using a variety of formats (e.g. raster or vector), codecs (BMP, JPEG, PNG), colour representations (e.g. RGB, P3) and colour depths (e.g. 8-bit, 10-bit). For the purpose of this work, an image $\mathbf{M}$ will be defined as a matrix of width w, height h and channels c, i.e. $\mathbf{M} \in \mathbb{R}^{w \times h \times c}$.

Depending on the tonality, there would be 1 channel for the greyscale, 3 channels for the colour and when there is a necessity to account for transparency, up to 4 channels (see figure 2.2). Typically, each channel will take a value between 0 and $2^d - 1$, with the usual values for d are $d = 8$ for standard images and $d = 10$ for high dynamic range (HDR) data (Mantiuk et al. 2015).

This chapter reviews the most important works related to representation learning on image data.

2.3.1 Principal Components Analysis

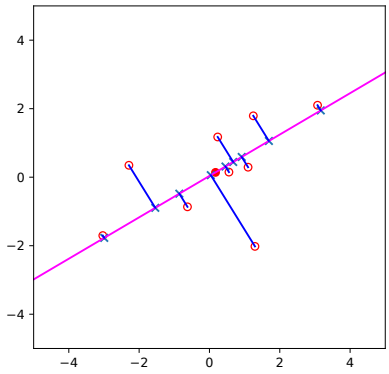

Figure 2.3: An illustrative example of PCA application for data projection $\mathbb{R}^2 \rightarrow \mathbb{R}$, where circles – original data, crosses –reconstructions, red line – selected eigenvector. Credit: (Murphy 2022).

One of the fundamental approaches to obtaining representations is Principal Component Analysis (PCA) algorithm. While not dedicated to images specifically, PCA is oftentimes used in preprocessing visual data for dimensionality reduction and feature extraction purposes due to its easy-to-understand geometric interpretation and inexpensive computational cost. As described in (Murphy 2022), PCA can be described as calculating an linear, orthonormal projection $\mathbf{W} \in \mathbb{R}^{h \times l}$ of high-dimensional data $\mathbf{x} \in \mathbb{R}^h$ to lower-dimensional embeddings $\mathbf{z} \in \mathbb{R}^l$ that minimise the average reconstruction error objective E:

$$E(\mathbf{W}, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{W}\mathbf{z}_i\|^2, \tag{2.14}$$

where $\|\cdot\|$ – ℓ^2-norm.

Assuming whitened data (i.e., data with the empirical mean subtracted and then divided by the empirical standard deviation), the optimal solution is then obtained, by selecting eigenvectors with largest eigenvalues of the following empirical covariance matrix:

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T = \frac{1}{n} \mathbf{X}_c^T \mathbf{X}_c, \tag{2.15}$$

where $\bar{\mathbf{x}}$ is an average of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ and the matrix $\mathbf{X}_c$ is a centred version of the $\mathbf{X} \in \mathbb{R}^{n \times h}$ data matrix.

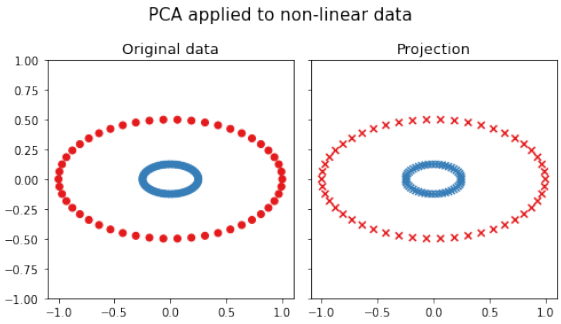

Figure 2.4: Application of PCA to non-linear dataset. Due to linear projection, PCA fails to cluster two classes of data.

As PCA is restricted to linear projections from higher- to lower-dimensional spaces, the simplicity it offers comes at the cost of the inability to model more complex dependencies (as presented in the figure 2.4). Example modification to improve PCA flexibility consists of KernelPCA, which utilises applying different functions for calculating non-linear distances between objects (called *kernels*). Other approaches include implementing the task from the perspective of Probabilistic PCA (Roweis 1997; Tipping and Bishop 1999) or even by considering the more general probabilistic method of factor analysis (Ghahramani, Hinton, et al. 1996) or t-SNE (Van der Maaten and Hinton 2008) (see figure 2.5).

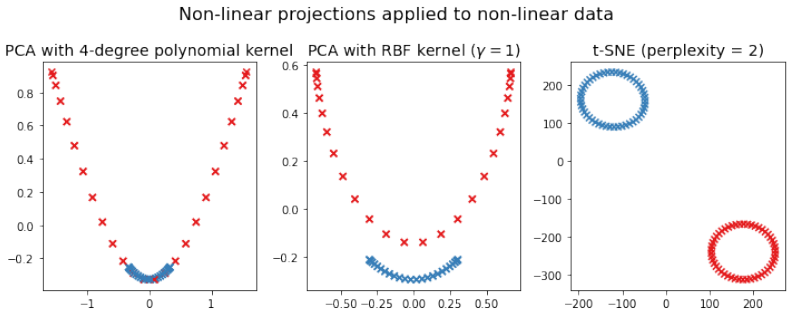

Figure 2.5: Application of non-linear projections to a non-linear dataset. Compared to the linear PCA, they more or less correctly cluster the original two circles dataset.

.

2.3.2 t-Stochastic Neighbour Embedding

In contrast to the algebraic approach of PCA, *t-Stochastic Neighbour Embedding (t-SNE)* (Van der Maaten and Hinton 2008) algorithm performs the data dimensionality reduction by employing an optimisation framework. The t-SNE performs learning the data embeddings by minimising the Kullback-Leibler divergence (see section 2.6.3) between the joint probabilities P in the high-dimensional space (equation (2.16)) and the joint probabilities Q (equation (2.18)) in the low-dimensional space. The modelling of the pairwise distances in the high-dimensional space is given as a two-way mean of conditional Gaussian probabilities, which modifies an original one-way approach in order to combat the original SNE (Hinton and Roweis 2002) problem of ignoring outliers:

$$P(i,j) = \frac{P(j|i) + P(i|j)}{2n}, \tag{2.16}$$

where n is a number of samples, i, j are the indices of data samples in the high-dimensional space and $P(\cdot|\cdot)$ is defined as:

$$P(i|j) = \begin{cases} \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right)\left(\sum_{k\neq i}\exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_k\|^2}{2\sigma_i^2}\right)\right)^{-1} & \text{if } i \neq j \\ 0 & \text{otherwise,} \end{cases} \tag{2.17}$$

where $\mathbf{x}_i, \mathbf{x}_j$ denote higher-dimensional data representations and σ_i^2 is the variance of the Gaussian that is centred on representation $\mathbf{x}_i$.

Moreover, as a way to solve the crowding problem (placing many samples that are equidistant in the higher-dimensional space in the proximity to each other in the lower-dimensional space), t-SNE introduces modelling the dissimilarity in the lower-dimensional space using the Student's t-distribution with 1 degree of freedom (i.e. the Cauchy distribution):

$$Q(i,j) = \begin{cases} \dfrac{(1+\|\mathbf{h}_i - \mathbf{h}_j\|^2)^{-1}}{\sum_{k \neq l}(1+\|\mathbf{h}_k - \mathbf{h}_l\|^2)^{-1}} & \text{if } i \neq j \\ 0 & \text{otherwise,} \end{cases} \tag{2.18}$$

where $\mathbf{h}_i, \mathbf{h}_j$ denote lower-dimensional data representations.

However, the disadvantage of t-SNE is the necessity for a manual selection of the *perplexity* value that determines the binary-searched value of variance σ_i around the higher-dimensional samples (equation (2.17)). The perplexity of probability distribution $P(i)$ induced by variation σ_i is defined as

$$\text{Perplexity}(P_i) = 2^{\mathbb{H}(P(i))}, \tag{2.19}$$

where $\mathbb{H}$ is an entropy measured in bits and defined as:

$$\mathbb{H}(P(i)) = -\sum_j P(j|i) \log_2 P(j|i). \tag{2.20}$$

While the perplexity can be described as a smoothing measure of the size of the neighbourhood, and the literature specifies that the SNE and t-SNE approaches are robust for the values in the range from 5 to 50, new research (Wattenberg et al. 2016) indicates that various values of perplexity can have a significant impact on the outcome of the clustering (and, in the extreme cases, making a difference between obtaining low-dimensional clusters and not). Furthermore, in contrast to PCA, the size and shapes of clusters, as well as distances between them, do not correlate with the original data (Van der Maaten and Hinton 2008).

2.3.3 Contrastive learning

Another approach to perform representation learning is *contrastive* or *metric learning*. Instead of attempting to map the closeness of the objects in high- and low-dimensional space, contrastive learning depends on additional labels provided alongside the original data samples that specify whether two objects belong to the same class. Contrastive learning works by learning non-linear transformations that keep similar (in terms of the class label) samples close to each other and dissimilar ones far apart in the latent space.

One of the earliest approaches to metric learning is contrastive loss (Chopra et al. 2005). For samples $\mathbf{x}_i, \mathbf{x}_j$ belonging to the same class, the mapping is optimised to minimise the distance between them in the feature space. The reverse criterion is applied for the samples of different classes, which can be written as:

$$L_c(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \min\limits_{\theta} \|F_\theta(\mathbf{x}_i) - F_\theta(\mathbf{x}_j)\|_2^2 & \text{if } y_i = y_j \\ \max\limits_{\theta} \|F_\theta(\mathbf{x}_i) - F_\theta(\mathbf{x}_j)\|_2^2 & \text{otherwise,} \end{cases} \tag{2.21}$$

where F_θ is the mapping function from higher- to lower-dimensional space parameterised with weights θ and usually represented as neural network, and $\|\cdot\|_2^2$ – squared ℓ^2 norm.

An extension of contrastive loss, called triplet loss, was presented in (Schroff et al. 2015). Instead of alternating between similar and different classes, triplet loss processes three samples at once – an anchor $\mathbf{x}$, a positive sample $\mathbf{x}^+$ (belonging to the same class as $\mathbf{x}$) and a negative sample $\mathbf{x}^-$ (belonging to a different class). The training objective $\mathcal{L}_t$ for this approach can be defined as

$$L_t(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathbb{X}} \max\left(0, \|F(\mathbf{x}) - F(\mathbf{x}^+)\|_2^2 - \|F(\mathbf{x}) - F(\mathbf{x}^-)\|_2^2 + \lambda\right), \tag{2.22}$$

where $\lambda \in [0, \infty)$ – margin hyperparameter defining minimum difference in the distance between similar and dissimilar samples and the rest as in equation (2.21).

Recently, contrastive learning has seen a rapid increase in popularity. In supervised setting, many extensions to the methods presented above have been introduced, such as Lifted Structured Loss (Oh Song et al. 2016), N-pair Loss (Sohn

2016), NCE (Gutmann and Hyvärinen 2010), InfoNCE (Van den Oord et al. 2018), Soft-Nearest Neighbours Loss (Frosst et al. 2019) or Quadruplet Loss (W. Chen et al. 2017). Metric learning has also been widely used for powerful self-supervised models, where samples belonging to the same class are obtained by performing various preprocessing operations on crops of the sample, as is the case in SimCLR (T. Chen et al. 2020), Barlow Twins (Zbontar et al. 2021) and BYOL (Grill et al. 2020).

2.3.4 Autoencoders

One of the first neural-network-based approaches to representation learning has been the *autoencoder* (Rumelhart et al. 1985; Hinton and Salakhutdinov 2006). In its basic form, the autoencoder consists of two parameterised functions – encoder E producing the latent representation and decoder D aiming at reconstructing the encoder's input based on those features. The encoder and decoder parameters are obtained during the optimisation (learning) process by minimising the reconstruction error function L on training samples $\mathbf{x} \in \mathbb{R}^n$:

$$L(\mathbf{x}) = \text{dist}(\mathbf{x}, D(E(\mathbf{x}))), \tag{2.23}$$

where $\text{dist}(\cdot, \cdot)$ is a distance function. It can be noted that when the encoder and decoder are modelled as one-layer linear layers and operate on representation lengths $z < n$, the task becomes equivalent to obtaining a Probabilistic PCA representation of size z. However, most contemporary autoencoders implement encoders and decoders as multi-layer neural networks with non-linear activations.

An exception to the typical case of bottleneck (or regularised) autoencoders (i.e. the case when the representation size z is smaller than data dimensionality d) is a concept of *sparse autoencoder* (Bengio et al. 2006). In this case bigger representations ($z > d$) are offset by penalising the output $\mathbf{z}$ of the representation layer with Student's-t penalty (Olshausen and Field 1996):

$$\text{penalty}(\mathbf{z}) = \sum_{i=0}^{|\mathbf{z}|} \log\left(1 + \left(\mathbf{z}^{(i)}\right)^2\right). \tag{2.24}$$

The representation learning using autoencoders has been described so far from a non-probabilistic perspective. The probabilistic approach was introduced in perhaps one of the most seminal works in the generative representation learning field – Variational Autoencoder (VAE) (Kingma and Welling 2014). Although VAE is a probabilistic derivation of the variational Bayesian method, its architecture resembles the autoencoder framework and is often grouped alongside the abovementioned approaches.

The objective of training the VAE is to model the true data distribution by approximating it with the parameterised joint distribution P_θ[1], which can be written as:

$$P_\theta(X) = \int_Z P_\theta(X, Z) dZ = \int_Z \underbrace{P_\theta(X|Z)}_{\text{cond. likelihood}} \underbrace{P_\theta(Z)}_{\text{prior}} dZ, \tag{2.25}$$

where Z is a random variable taking values from the latent representation space $\mathbb{Z}$ and X is the random variable taking values from the data space $\mathbb{X}$.

Assuming that the model is already trained (i.e. the optimal parameters θ^* have been found), generating the data from such model is performed in two steps. First the value $\tilde{\mathbf{z}}$ is drawn from the prior distribution $P_{\theta^*}(Z)$. Then, $\tilde{\mathbf{x}}$ is sampled from a distribution $P_{\theta^*}(X|Z = \tilde{\mathbf{z}})$.

The optimal set of parameters θ^* is the set that maximises the (log-)likelihood of real data, which can be written as

$$\theta^* = \arg\max_\theta \sum_{\mathbf{x} \in \mathcal{D}} \log P_\theta(\mathbf{x}) = \arg\max_\theta \sum_{\mathbf{x} \in \mathcal{D}} \log \int_Z P_\theta(X = \mathbf{x}|Z) P_\theta(Z) dZ. \tag{2.26}$$

However, determining θ^* as presented in equation (2.26) is infeasible due to the necessity of checking all the possible values of Z. Therefore, there is a need to obtain the posterior distribution $P_\theta(Z|X)$ that specifies the representation given the data. Since posterior distribution $P_\theta(Z|X)$ is intractable, it is approximated using a variational posterior $Q_\phi(Z|X)$.

[1]For the purpose of VAE definition, the P_θ will denote the probability density P represented with function (e.g. neural network) parameterised with θ. Parameterisation with θ has been omitted for brevity in the previous chapters without introducing ambiguity.

Compared to the regularised autoencoder approaches, the distributions $P_\theta(X|Z)$ and $Q_\phi(Z|X)$ fulfil roles similar to the decoder and encoder functions, respectively. However, contrary to regular autoencoders, training of the VAE is done by matching the approximated posterior distribution to the original one. It can be derived as:

$$\begin{aligned}
D_{\mathrm{KL}}\left(Q_\phi(Z|X)\|P_\theta(Z|X)\right) &= \int Q_\phi(Z|X)\log\frac{Q_\phi(Z|X)}{P_\theta(Z|X)}dZ \\
&= \int Q_\phi(Z|X)\log\frac{Q_\phi(Z|X)P_\theta(X)}{P_\theta(Z,X)}dZ \\
&= \int Q_\phi(Z|X)\left(\log P_\theta(X) + \log\frac{Q_\phi(Z|X)}{P_\theta(Z,X)}\right)dZ \\
&= \log P_\theta(X) + \int Q_\phi(Z|X)\log\frac{Q_\phi(Z|X)}{P_\theta(Z,X)}dZ \\
&= \log P_\theta(X) + \int Q_\phi(Z|X)\log\frac{Q_\phi(Z|X)}{P_\theta(X|Z)P_\theta(Z)}dZ \\
&= \log P_\theta(X) + \mathbb{E}_{Z\sim Q_\phi(Z|X)}\left[\log\frac{Q_\phi(Z|X)}{P_\theta(Z)} - \log P_\theta(X|Z)\right] \\
&= \log P_\theta(X) + D_{\mathrm{KL}}\left(Q_\phi(Z|X)\|P_\theta(Z)\right) - \mathbb{E}_{Z\sim Q_\phi(Z|X)}\log P_\theta(X|Z),
\end{aligned} \tag{2.27}$$

where D_{KL} is a Kullback-Leibler divergence, described in section 2.6.3.

By rearranging the terms of the final equality in equation (2.27) the model objective, consisting of likelihood maximisation $P_\theta(X)$ and matching the real posterior to the approximate, variational posterior $D_{\mathrm{KL}}\left(Q_\phi(Z|X)\|P_\theta(Z|X)\right)$ can be written down as:

$$\log P_\theta(X) - D_{\mathrm{KL}}\left(Q_\phi(Z|X)\|P_\theta(Z|X)\right) = \mathbb{E}_{Z\sim Q_\phi(Z|X)}\log P_\theta(X|Z) - D_{\mathrm{KL}}\left(Q_\phi(Z|X)\|P_\theta(Z)\right) \tag{2.28}$$

However, as mentioned before, the term $D_{\mathrm{KL}}\left(Q_\phi(Z|X)\|P_\theta(Z|X)\right)$ is intractable. Therefore, this term is dropped from the VAE objective function in practice. Knowing that $D_{\mathrm{KL}}\left(Q_\phi(Z|X)\|P_\theta(Z|X)\right) \geqslant 0$, by removing this term, the new joint objective is a lower bound of the true log-likelihood function and is called *evidence lower*

bound (ELBO). It is defined as:

$$\begin{aligned} L_{VAE} &= -\log P_\theta(X) + D_{\mathrm{KL}}\left(Q_\phi(Z|X)|P_\theta(Z|X)\right) \\ &= \mathbb{E}_{Z\sim Q_\phi(Z|X)}\left[\underbrace{-\log P_\theta(X|Z)}_{\text{reconstruction}} + \underbrace{D_{\mathrm{KL}}\left(Q_\phi(Z|X)\|P_\theta(Z)\right)}_{\text{regularization}}\right]. \end{aligned} \tag{2.29}$$

It can be observed that the VAE training consists of not only optimising the model for input reconstruction, but also by matching variational posterior $Q_\phi(Z|X)$ to prior distribution $P_\theta(Z)$ (usually standard normal distribution $\mathcal{N}(0, \mathbf{I})$).

An interesting modification to VAE optimisation objective is presented in *β-VAE* (Higgins et al. 2016). By adding the scaling hyperparameter $\beta > 1$ to the regularisation part of the VAE objective, the β-VAE puts a stronger constraint on the latent representation, limiting its capacity. This model property can be used to learn disentangled representations, i.e. representations where each feature is tied with only a single underlying latent variable of the data.

The VAE and β-VAE utilise the KL divergence to regularise the latent representation distribution. However, this approach restricts the selection of prior distribution to only ones for which KL divergence exists and is traceable. The *Adversarial Autoencoder (AAE)* (Makhzani et al. 2016) framework sidesteps this limitation by imposing the regularisation on the latent distribution with an additional discriminator function. Trained in an adversarial fashion (Goodfellow et al. 2014), the discriminator aims to differentiate between representations obtained from the encoder and those sampled from the prior distribution. In contrast, an encoder tries to produce output that would fool the discriminator. Therefore for a given prior distribution, adversarial training assumes only an ability to sample without the need for divergence calculations.

2.4 Representation learning for 3-dimensional data

This chapter reviews the most important developments related to deep learning on 3-dimensional data. First, the approaches to obtain depth from the set of 2-

dimensional images are introduced. Next, each section describes a specific type of spatial data (i.e. voxels, point clouds and meshes, as presented in figure 2.6) along with relevant literature. The section after that gives an overview of the most popular 3-dimensional datasets. The chapter is concluded with metrics used for methods evaluation on the most prominent problems researched for applications to 3-d data.

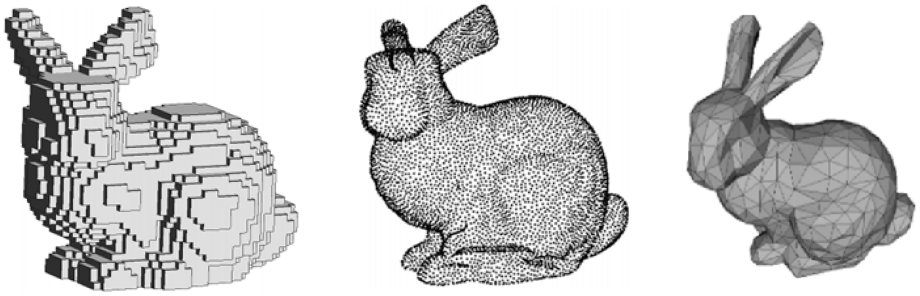

Figure 2.6: The example of different types of 3-dimensional data. From the left: voxel-based shape, point cloud and a mesh. Adapted from (Hoang et al. 2019).

2.4.1 Multi-view-based approaches

The ability to capture 2-dimensional data came long before the first devices capable of capturing three dimensions were created and, to this day, far outweigh them in terms of popularity and ubiquity. Therefore, one of the fundamental challenges of computer vision is reconstructing spatial information based on planar representations.

Several approaches to multi-view inference of 3-dimensional shapes have been proposed, for the classification and recognition (Hang Su et al. 2015; Qi et al. 2016; Kanezaki et al. 2018), generation (Arsalan Soltani et al. 2017) and retrieval (Ma et al. 2018; J. Jiang et al. 2019).

Recently, there can be observed the growing popularity of approaches performing single-view inference, e.g. for the purpose of monocular depth estimation (Godard et al. 2017; Qi et al. 2018; Godard et al. 2019; Ranftl et al. 2021; Safadoust and

Güney 2021), 3-dimensional reconstruction (Hoiem et al. 2005; Fouhey et al. 2013; Choy et al. 2016; Gkioxari et al. 2019).

2.4.2 Voxel-based approaches

Images present the 3-dimensional world in a 2-dimensional way, therefore losing information in the process. Thus, an appropriate data type must be used if one wants to capture the view without losing the scene's depth. The natural generalisation of a 2-dimensional pixel represented by a square to 3-dimensional space is a cube called a *voxel*. Same as in the image case, voxel-based data must have a fixed width w, height h, but with an additional dimension for depth d. Each space in the voxel grid may contain information about the colour value c. Therefore, for this work, we will define voxel-based data as a tensor $\mathbf{V} \in \mathbb{N}^{w \times h \times d \times c}$.

Most often, the voxels are stored as a single scalar (i.e. $c = 1$ in the case of a binary or monochromatic voxels) or as a vector containing colour information, as is the case with regular images (i.e. $c \in \{3, 4\}$).

Voxel-based data can offer a realistic representation of 3-dimensional space. However, to do so, they require very high resolution, making them impractical in most common uses, where spatial data would be used. However, voxels are still applied in areas where high precision is a crucial requirement, such as computed tomography scanning (Weese et al. 1997), magnetic resonance imaging (Hutton et al. 2008), medical ultrasound (Aydin et al. 2020). On the other hand, low-resolution voxels are often used as an intermediate representation when performing a rendering of a 3-dimensional scene, making it useful for a fast and approximate preview of the final result without the need for expensive computations (Levoy 1990; Kanzler et al. 2018; Yan et al. 2018)).

As the natural extension of pixels, voxels were among the first 3-dimensional data types to be extensively studied for potential applications using deep learning. Generalising the convolution operator known from 2-d convolutional neural networks (CNNs) to 3 dimensions introduced several concepts utilised for image data to voxel-

based data (Maturana and Scherer 2015; Brock et al. 2016; J. Wu et al. 2016; Zhou and Tuzel 2018).

Moreover, adding the third dimension to the data introduced new classes of problems, such as object orientation (Sedaghat et al. 2017), 3-d shape generation (Z. Chen and H. Zhang 2019), 3-dimensional medical imaging (Suzuki 2017; M. Kim et al. 2019; S.-C. Huang et al. 2020) and 3-d object recognition (Xiang et al. 2015).

However, since voxel-based data lies on a structured grid, it may be insufficient to capture highly detailed, unevenly structured data. Therefore it is most often used in areas where the high recording fidelity is not required (e.g. approximations of the results of computationally-complex operations (Levoy 1990; Kanzler et al. 2018; Yan et al. 2018)) or with access to high precision scanners (e.g. computed tomography scanners (Çiçek et al. 2016; Siddique et al. 2021))

Recently, methods applied to voxels are being enhanced with the point cloud representations to take advantage of both structured and unstructured data (Cao et al. 2019; Liu et al. 2019).

2.4.3 Point cloud-based approaches

Another type of 3-d data representation and the primary point of interest of this dissertation is *point cloud*. Point cloud is a data type represented by a set of points in a given multi-dimensional space. The d-dimensional point cloud $\mathcal{P}$ is defined as a set of n real-valued points, such as

$$\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}, \tag{2.30}$$

where each point $\mathbf{p} \in \mathcal{P}$ is represented by a column vector of size d, that we can write as

$$\mathbf{p} = [p_1\ p_2\ \ldots\ p_d]^T \tag{2.31}$$

Compared to voxels, point clouds excel when capturing the objects, surfaces or large areas of irregular granularity or density, e.g. when collecting the 3-dimensional view of environmental surroundings or performing nuclear magnetic resonance (Guo

et al. 2020). Therefore, it is one of the most common ways to store and represent 3D data gathered by the scanning devices such as LIDAR or multi-camera setups. Point clouds are the most commonly used for the purpose of simultaneous localisation and mapping (SLAM) (P. Kim et al. 2018; Palomer et al. 2019; Singandhupe and La 2019), the self-driving technology (Y. Li et al. 2020), and environment or object capture (Tang et al. 2010), where 3-dimensional, long-distance scanners, such as LIDARs calculate ranges using lasers. They are also used in medical and biological imaging, such as nuclear magnetic resonance (NMR) (Fitzpatrick et al. 2006).

One of the first models able to work on point sets was the PointNet (Qi et al. 2017a), by introducing a feature representation backbone that is able to encode the latent factors of point clouds regardless of the order of the input. The encoder treats each point of the point cloud independently and is utilising a permutation invariant function as a final activation layer, namely a max(·) function (detailed description of the architecture in figure 2.7).

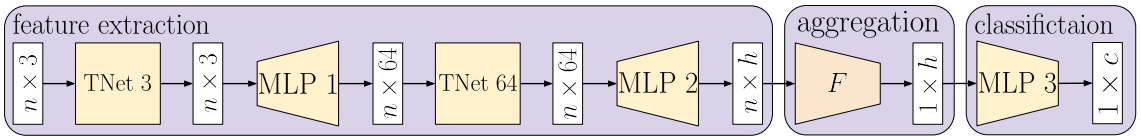

Figure 2.7: A PointNet architecture consists of three parts: (a) feature extraction encoding separately each point of a given point cloud to a feature vector of length h, (b) aggregation combining features feature-wise across all points, and (c) classification part learning a mapping from aggregated features to class labels. The `TNet K` refers to learnable affine transforms in k dimensions introduced in (Qi et al. 2017a).

PointNet (Qi et al. 2017a) achieved state-of-the-art results for point cloud classification and instance as well as semantic segmentation. However, the main PointNet drawback is treating each point without its context. In PointNet++ (Qi et al. 2017b) the authors extend the PointNet's idea to generating hierarchical point cloud features. They do so by alternating between sampling & grouping layers and PointNet processing modules to capture local context at different scales.

PointNet utilised 1-dimensional convolutional operators as an efficient way to

share network parameters between each point in the cloud. A different approach, presented in PointConv (W. Wu et al. 2019) proposes an extension of traditional convolution and deconvolution used in Convolutional Neural Networks for images into the point clouds. The proposed adaptation works by performing a Monte Carlo approximation of the 3-d continuous convolution operator. A Multi-Layer Perceptron is then used to approximate a weight function for each convolutional filter.

One of the first works to utilise the aforementioned representation learning for point clouds was (Achlioptas et al. 2018). In this work, the authors present a generative approach that works on raw point clouds to reconstruct input and generate new shapes. The authors utilised a two-step solution. First, the bottleneck autoencoder is trained to obtain latent representations for point cloud shapes. Next, a generative model (Wasserstein (Arjovsky et al. 2017) GAN with Gradient Penalty (Gulrajani et al. 2017) or Gaussian Mixture Model (Murphy 2012)) is trained to approximate the feature space mentioned above. Sampled feature vectors are then passed to the decoding module of the autoencoder, producing a point cloud.

The other approach to the generation and matching of 3-dimensional shapes is presented in (Deprelle et al. 2019). In this work, the authors introduce the idea of storing and representing shapes as learnable elementary primitives. The obtained primitives are then used to generate and visualise the underlying features of the data.

The domain adaptation framework utilised in PointDAN (Qin et al. 2019) introduces applying rich spatial and geometric information contained in point clouds to the representation learning and cross-domain matching. PointDAN works by aligning features of point clouds from different datasets on a local and global level. The Self-Adaptive module performs local alignment with an adjusted receptive field. On the other hand, the authors employ an adversarial training framework for global alignment.

Recently, another class of generative models called *normalising flows* (Rezende and Mohamed 2015; Papamakarios et al. 2021) that is already frequently in application

to image data (Dinh et al. 2015; Kingma et al. 2016; Dinh et al. 2017; Papamakarios et al. 2017; Kingma and Dhariwal 2018; Grathwohl et al. 2019) is gaining popularity for the purposes of representation learning and generation of 3-dimensional shapes. In the point cloud domain, one of the first models that utilised normalising flows was PointFlow (G. Yang et al. 2019). In this work, the authors present a probabilistic framework to generate 3D point clouds by training the hierarchical distribution of features. It is done by utilising Continuous Normalising Flows (R. T. Q. Chen et al. 2018) to model the shape and the class distribution separately.

In (Pumarola et al. 2020) the authors introduce C-flow – conditional training for normalising flows. By conditioning the shape generation process on condition embeddings, instead of sampling from the flow as in (G. Yang et al. 2019), C-flow allows modelling of a distribution spanning multiple shapes. Moreover, using the discrete normalising flow Glow (Kingma and Dhariwal 2018) instead of continuous flows allows for much faster inference time by creating an entire shape at once, instead of sequential inference point by point.

2.4.4 Mesh-based approaches

Voxels possessed one important property – they were able to model the volume of the given object, unlike point clouds that were only able to represent the sparse set of points on its surface. The possibility of modelling the continuous shape is a critical feature required for many graphical and engineering applications. Therefore, the polygon meshes are used where there is a need for a homogeneous shape.

We can describe the polygon mesh $\mathcal{M}$ as a tuple

$$\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F}), \tag{2.32}$$

consisting of vertices $\mathcal{V}$ – a point cloud, edges $\mathcal{E}$ – a set of tuples of points from a point cloud and faces $\mathcal{F}$ – the set of closed sets of edges.

While meshes offer superior 3-dimensional volume representation compared to point clouds, they are often difficult to process from the machine learning standpoint.

Providing the vertices with additional information about edges and faces increases the data complexity, requiring more extensive and more advanced deep learning models (Gkioxari et al. 2019).

Training deep learning models on mesh data is a relatively new but dynamically growing branch of 3-dimensional machine learning. One of the first works to introduce deep learning on the meshes was MeshNet (Yutong Feng et al. 2019). This paper introduces new design blocks that capture and aggregate features of polygon faces based on their statistics, e.g. centre, corners, normal vector and their neighbourhood.

Mesh R-CNN (Gkioxari et al. 2019) is an extension of previous work applied to images, i.e. R-CNN (Girshick et al. 2014), Fast R-CNN (Girshick 2015), Faster R-CNN (H. Jiang and Learned-Miller 2017), Mask R-CNN (He et al. 2017). While not taking a mesh as an input to the network, it allows the creation of watertight meshes from a single 2-dimensional image using voxel-based intermediate representation. Similar work, presented in AtlasNet (Groueix et al. 2018) uses a set of parametric surface elements to directly infer a surface representation of the shape from a still image or a point cloud. The final mesh combines patch elements predicted by a neural network based on sampled points from the 2- or 3-dimensional input.

The other interesting direction of research is focused on learning *signed distance functions* (*SDFs*). In this approach, the goal is to model the shape as a decision boundary, represented as a function that returns the positive, negative or zero value depending on whether the given point lies outside, inside or precisely on the shape's surface. This method, implemented with deep neural networks, was introduced in DeepSDF (Park et al. 2019). Due to the superior generalisation ability of deep models compared to more "classical" counterparts, it was possible to continuously model the boundary, which removed the need to discretise the input and output. Similar work was presented in Occupancy Networks (Mescheder et al. 2019), Convolutional Occupancy Networks (Peng et al. 2020), Deep level sets (Michałkiewicz et al. 2019), IM-NET (Z. Chen and H. Zhang 2019).

2.5 Datasets

Most of the datasets presented here are mesh-based and provided as a collection of points in 3-dimensional space, along with the pairs of point indices that form edges. Since the primary objective of this work is focused on point clouds, the data for the experiments described in the latter part of this thesis was usually preprocessed by generating a 3-dimensional mesh from the provided data, which was then transformed into a point cloud by uniformly sampling a given number of points laying on the mesh surface.

ModelNet The ModelNet 10 (MN10) and its expansion ModelNet40 (MN40) (Z. Wu et al. 2015; Sedaghat et al. 2017) were among the first 3-dimensional benchmark datasets that are still being widely used today. In terms of relative popularity, they can be compared to such image datasets as MNIST (LeCun et al. 1998), CIFAR10 (Krizhevsky 2009) or SVHN (Netzer et al. 2011).

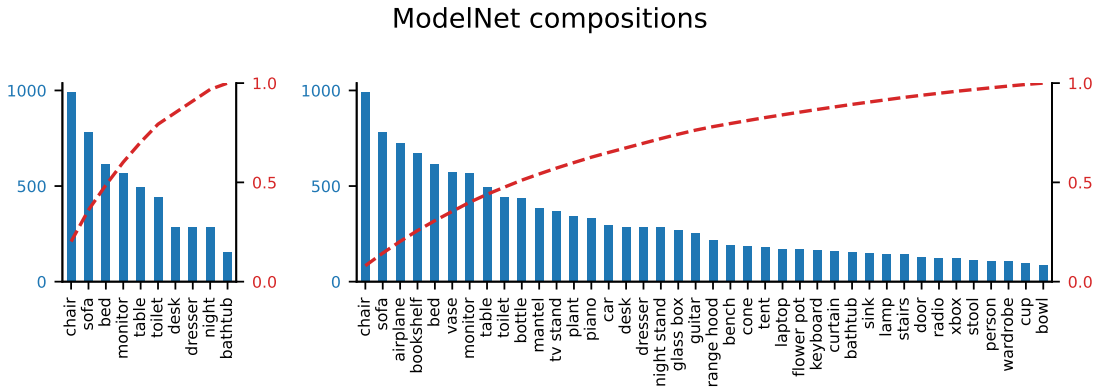

Figure 2.8: Class distribution for ModelNet datasets.

MN10 contains 4899 samples that belong to 10 classes, while MN40 distributes 12311 shapes among 40 categories. The datasets offer shapes cleaned of any scanning artefacts and aligned to the vertical axis.

However, the class distribution is heavily imbalanced, with over 50 % of all the samples belonging to the top-3 in the case of ModelNet10 and the top-9 most rep-

resented shapes of ModelNet40 (see figure 2.8).

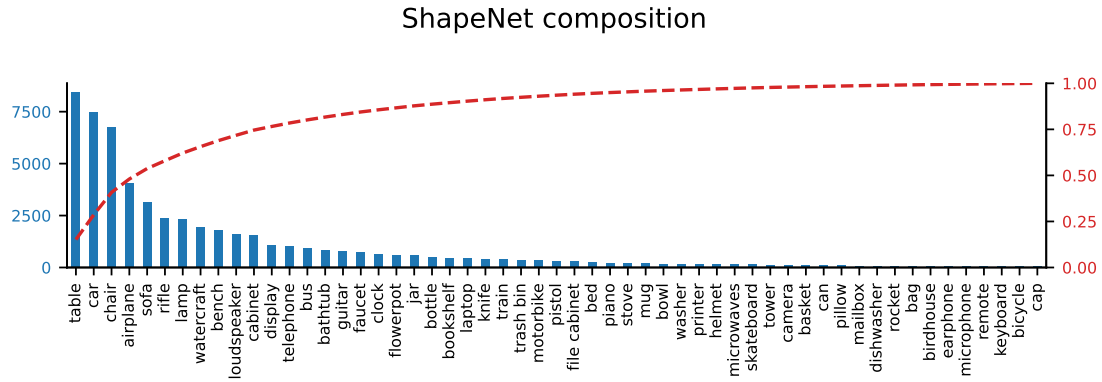

Figure 2.9: Class distribution for ShapeNet dataset.

ShapeNet In ShapeNet (Chang et al. 2015) the problem of imbalanced classes is even more noticeable. While containing over 55,000 models, over 50 % of all the samples belong to the four biggest out of 55 shape classes in total. However, this imbalance proves challenging when training the machine learning models on classes that are barely represented in the dataset, especially when considering the variety of shapes and their level of detail.

ScanNet The ScanNet (Dai et al. 2017a) dataset features over 1500 samples for over 700 unique scenes. The samples present indoor environments and are semantically segmented into 20 classes that cover around 90 % of the total surface of the shapes (Y. Xie et al. 2020).

The data was captured by scanning the surroundings using RGB-D cameras and was later reconstructed into mesh scenes. The dataset was converted into voxel representation for segmentation purposes and annotated manually.

The dataset has seen usage in tasks of 3-dimensional object detection and segmentation (Tchapmi et al. 2017; Qiangui Huang et al. 2018; Qi et al. 2019; Guo et al. 2020), monocular depth estimation (Fu et al. 2018; Ranftl et al. 2020), as well as shape and scene completion (Dai et al. 2017b; Yuan et al. 2018; Y. Zhang and

Funkhouser 2018).

D-FAUST The Dynamic FAUST (D-FAUST) dataset (Bogo et al. 2017) consists of full-body 3-dimensional scans of 10 human subjects performing a total of 129 activities, such as jumping, running etc. The dataset is a further work based on the FAUST dataset (Bogo et al. 2014) extending it to 4-dimensional data by providing over 40,000 spatio-temporally coherent meshes. Captured by the multi-camera setup, they are recorded at 60 frames per second, offering a smooth transition between each step for accurate modelling of a human silhouette in motion. In addition to registering the general shape of participants, D-FAUST contains information about texture, making it possible to present the data realistically.

D-FAUST dataset is often used in tasks of reconstructing 3-dimensional models of people based on images and videos (Alldieck et al. 2018a; Alldieck et al. 2018b; Lazova et al. 2019; Y. Chen et al. 2020), as well as object key-point tracking and body segmentation (Pons-Moll et al. 2017; Niemeyer et al. 2019; Paschalidou et al. 2020).

Pix3D The Pix3D (Sun et al. 2018) provides three collections of data – the dataset of images, the dataset of 3-dimensional voxel-based shapes and the image-shape pairs for aligning the shapes of different modalities. It consists of almost 400 3-dimensional shapes belonging to nine categories. In total, over 10,000 pixel-accurate pairings between 3-d and 2-d were provided.

The Pix3D dataset was created by extending the existing IKEA dataset (Lim et al. 2013) with new images found on the web as well as by scanning an additional RGBD data. The shapes were then aligned by solving for the 3-d pose of the object in the image using the Efficient Perspective-n-Point (EPnP) algorithm (Levenberg 1944).

This dataset is often used for the problems of multi-modal matching (Georgakis et al. 2019; Popov et al. 2020) and generating 3-dimensional shape from a single- or multi-view (H. Xie et al. 2019; Nie et al. 2020).

Table 2.1: The comparison of the selected datasets used in machine learning research applied to spatial data. The datasets used for training and evaluation purposes in the approaches presented in the following dissertation are marked with the **bolded font**.

Dataset	Type	# samples	# classes	Balanced	Annotations
MNIST	2-d point cloud	50,000	10	✓	class labels
ModelNet10	3-d mesh	4,899	10	✗	class labels
ModelNet40	3-d mesh	12,311	40	✗	class labels
ShapeNet	3-d mesh	over 55,000	55	✗	class, part and keypoint labels
ScanNet	rgb-d	1513	19	✗	voxel semantic segmentation labels
D-FAUST	mesh + texture	over 40,000	129	✓	class labels
Pix3D	image and voxel	400	9	✗	class labels, correspondence between 2-d and 3-d

Datasets summary The table 2.1 contains a summary of datasets discussed in this section. For training and evaluation purposes of the methods that constitute the contribution of this dissertation, four of the presented datasets were selected: MNIST, ModelNet 10 & 40 and ShapeNet. The choice of the datasets is dictated by their popularity in similar methods from the literature, therefore providing an opportunity to compare the proposed approaches against state-of-the-art methods directly. The ModelNet 10 is widely regarded as *the MNIST of point clouds*, making it the standard dataset for model evaluation. Additionally, both ModelNet 40 and ShapeNet contain a large number of samples belonging to diverse classes. Moreover, ModelNet and ShapeNet datasets share a substantial amount of class labels, making it possible to evaluate the abstraction and transfer of features between datasets, as described in section 2.2. On top of that, the 2-d point cloud MNIST dataset serves the purpose of evaluating the proposed Random Compression Rehearsal approach (chapter 4) on lower-dimensional point clouds.

2.6 Metrics

Metrics constitute the indispensable way to evaluate the examined approaches. However, representations obtained from machine learning models consist of abstract,

numerical features that cannot be easily deciphered, therefore posing a challenge in creating a well-defined and interpretable metric of the quality of a given representation. Instead of measuring the representation quality directly, the standard approach is to evaluate it on downstream tasks, such as classification or clustering, in which the representation is passed as an input. This section outlines the essential metrics for assessing machine learning models, presented from the point cloud data point of view.

2.6.1 Classification

The fundamental metric for any classification model is *accuracy*, describing the fraction of predictions performed correctly. Formally, it is defined as:

$$\text{accuracy}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} I(\mathbf{y}^{(i)} = \tilde{\mathbf{y}}^{(i)}), \tag{2.33}$$

where $I(\cdot)$ is the indicator function, $\mathbf{y}$ – true labels, $\tilde{\mathbf{y}}$ – predicted labels and n – the length of vectors $\mathbf{y}$ and $\tilde{\mathbf{y}}$.

The definition of accuracy provided above counts each classified example equally. In the case of unbalanced datasets (i.e. when dataset classes are represented with different number of examples) one can also use *balanced accuracy*:

$$\text{balanced accuracy}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{c} \sum_{c_i=1}^{c} \frac{1}{s_i} \sum_{s=1}^{s_i} I(\tilde{\mathbf{y}}_{c_i}^{(s)} = c_i), \tag{2.34}$$

where c - number of classes in the dataset, s_i - number of samples that belong to class c_i, $\tilde{\mathbf{y}}_{c_i}$ - vector of predicted labels that correspond to true label $\mathbf{y} = c$ and the rest as in equation (2.33).

In case of binary classification task (i.e. classification with only two possible classes), accuracy can also be defined in terms of positives and negatives:

$$\text{accuracy}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.35}$$

where tp – number of true positives ($\tilde{y} = 1 = y$), tn – number of true negatives ($\tilde{y} = 0 = y$), fp – number of false positives ($\tilde{y} = 1 \neq y$), fn – number of false negatives ($\tilde{y} = 0 \neq y$) and the rest as in equation (2.33).

The accuracy scores range from 0 (worst, when none of the predicted class labels matches the corresponding true label) to 1 (best, when $\mathbf{y} = \tilde{\mathbf{y}}$). Although the higher the accuracy score, the better it usually fails to faithfully describe the classifier quality, especially in the case of a heavily unbalanced dataset (e.g. the dataset consisting of 1 example of a positive class and 99 examples of a negative class, marked by a classifier as all negative will have 99% accuracy). Therefore, there is a need for additional metrics of *precision* (the proportion of correct, positive identification) and *recall* (the proportion of identified actual, positives). They are defined as follows:

$$\begin{aligned} \text{precision} &= \frac{tp}{tp + fp} \\ \text{recall} &= \frac{tp}{tp + fn}, \end{aligned} \tag{2.36}$$

with tp, fp, fn defined as in equation (2.35). Same as with accuracy, precision and recall values range from 0 (worst) to 1 (best result). For the extreme example described above, they would both be equal to 0 (no true positives).

It is often convenient to have only one value for comparison purposes. Thus the value of harmonic mean between precision and recall is calculated, called *F-measure* or *F_1-score*, defined as:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{tp}{tp + 0.5(fp + fn)}, \tag{2.37}$$

with tp, fp, fn as in equation (2.35).

2.6.2 Information retrieval

One of the key metrics for measuring the model quality on information retrieval tasks is mean average precision (mAP). It is defined based on precision and average precision (AP) metrics. Average precision at place k is defined as the change of recall based

$$\text{AP}_k(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{\sum_{i=1}^{k} \mathbf{y}^{(k)}} \sum_{i=1}^{k} \text{precision}(\mathbf{y}^{(1:k)}, \tilde{\mathbf{y}}^{(1:k)}) \cdot I(\mathbf{y}^{(k)} = \tilde{\mathbf{y}}^{(k)} = 1), \tag{2.38}$$

where $\mathbf{y}$ – vector of ground truth values, $\tilde{\mathbf{y}}$ – vector of predictions. Therefore it can be observed, that the highest value of Average Precision occur when two condition are met: 1) all the searched values have been found and 2) the relevant samples have been retrieved as soon as possible (e.g. when returning 10 results with 5 relevant samples, it is best if they are placed at spots 1-5.)

Mean average precision is defined as a mean of AP over all possible values of k:

$$\text{mAP}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n} \sum_{k=1}^{n} \text{AP}_k(\mathbf{y}, \tilde{\mathbf{y}}), \tag{2.39}$$

where n - length of the vectors ($\mathbf{y}$ and $\tilde{\mathbf{y}}$).

2.6.3 Distribution fit

One of the key metrics in any domain related to information theory is *Kullback-Leibler divergence* (KLD, also called *relative entropy*) (Kullback and Leibler 1951). Given probability distributions P and Q, the KLD is defined as in equation (2.40):

$$D_{\text{KL}}(P\|Q) = \int_{-\infty}^{\infty} P(x) \log \frac{P(x)}{Q(x)} dx, \tag{2.40}$$

or with the integral replaced by sum if the P and Q are discrete distributions instead, as in equation (2.41):

$$D_{\text{KL}}(P\|Q) = \sum_{x \in \mathbb{X}} P(x) \log \frac{P(x)}{Q(x)}. \tag{2.41}$$

The Kullback-Leibler divergence can be described as the average number of additional bits of data needed for encoding the true distribution P with the approximate distribution Q (Murphy 2012). Therefore, the better the Q distribution matches P, the closer the KLD will be to 0 i.e.

$$D_{\text{KL}}(P\|Q) = 0 \iff P = Q. \tag{2.42}$$

Thus, in the machine learning domain, the Kullback-Leibler divergence is mainly used for applications of learning the approximation of (often difficult or impossible

to compute) true distribution or as regularisation of a given space to a specified *a priori* distribution. The notable examples of using the KLD to train the neural networks are Variational Autoencoders (Kingma and Welling 2014).

However, in general, KLD is not symmetrical, i.e. $D_{\mathrm{KL}}(P \parallel Q) \neq D_{\mathrm{KL}}(Q \parallel P)$, making it unfit to be used as a distance function between two probability distributions. Therefore, in order to obtain the symmetrical divergence between two distributions P and Q the *Jensen-Shannon divergence* (JSD) (Menéndez et al. 1997) is used instead, which is defined as:

$$D_{\mathrm{JS}}(P\|Q) = \frac{D_{\mathrm{KL}}(P\|M) + D_{\mathrm{KL}}(Q\|M)}{2}, \tag{2.43}$$

where M is an average between distributions P and Q, i.e.

$$M = \frac{P+Q}{2}. \tag{2.44}$$

The square root of JSD is also often used as a metric, called *Jensen-Shannon distance* function (Endres and Schindelin 2003).

The Kullback-Leibler divergence, defined as in equation (2.41), can be rewritten as:

$$D_{\mathrm{KL}}(P\|Q) = \overbrace{\sum_{x\in\mathbb{X}} P(x)\log P(x)}^{-\mathbb{H}(P)} - \overbrace{\sum_{x\in\mathbb{X}} P(x)\log Q(x)}^{-\mathbb{H}(P,Q)}. \tag{2.45}$$

The first term $\mathbb{H}(P)$ called *entropy* measures the uncertainty of probability distribution P. Therefore it is the highest in the case of the uniform and 0 (lowest) for the degenerate distribution. The second entropy term $\mathbb{H}(P,Q)$ is called *cross-entropy*. From an information theory perspective, it can be interpreted as an expected number of bits needed to compress the distribution P with Q (Murphy 2022).

Cross-entropy is primarily used as a loss function in classification tasks, in which the distribution P is represented with one-hot vectors, i.e. binary vectors with only a single value set. In this case, cross-entropy can be defined as:

$$\mathbb{H}(P = \delta(X = c), Q) = -\sum_{x\in\mathbb{X}} \delta(x = c)\log Q(x) = -\log Q(c), \tag{2.46}$$

where $\delta(\cdot)$ – Dirac delta and c – the position of one-hot vector set to 1. The equation (2.46) is known as *negative log-likelihood loss*.

2.6.4 Distance between sets

Due to the unordered nature of point clouds, the usual distance functions calculating the element-wise difference between objects are not applicable. Therefore, to represent the dissimilarity between two point clouds, one needs to use the metric calculating the distance between sets. The most common are the *Chamfer Distance (CD)* and the *Earth Mover's Distance (EMD)* or *Wasserstein Distance* (Rubner et al. 2000).

Given in equation (2.47), Chamfer Distance between two point clouds $\mathcal{P}_1$ and $\mathcal{P}_2$ is defined as an average of two sums – a sum of the shortest distances from each point in $\mathcal{P}_1$ to it's closest point in $\mathcal{P}_2$ and analogically, from each point in $\mathcal{P}_2$ to its closest point in $\mathcal{P}_1$:

$$CD(\mathcal{P}_1, \mathcal{P}_2) = \sum_{\mathbf{p}_1 \in \mathcal{P}_1} \min_{\mathbf{p}_2 \in \mathcal{P}_2} \|\mathbf{p}_1 - \mathbf{p}_2\|_2^2 + \sum_{\mathbf{p}_2 \in \mathcal{P}_2} \min_{\mathbf{p}_1 \in \mathcal{P}_1} \|\mathbf{p}_1 - \mathbf{p}_2\|_2^2 , \tag{2.47}$$

where $\|\cdot\|_2^2$ – squared ℓ^2 norm.

On the other hand, Earth Mover's Distance, given in equation (2.48), solves the optimal transport problem (Kolouri et al. 2017) between point clouds $\mathcal{P}_1$ and $\mathcal{P}_2$, i.e., finds a bijection between input sets that results in the infimum distance between matched points. It can be written as

$$EMD(\mathcal{P}_1, \mathcal{P}_2) = \inf_{\Psi: \mathcal{P}_1 \to \mathcal{P}_2} \sum_{\mathbf{p} \in \mathcal{P}_1} \frac{\|\mathbf{p} - \Psi(\mathbf{p})\|_2^2}{2}, \tag{2.48}$$

where $\|\cdot\|_2^2$ – squared ℓ^2 norm and Ψ is a bijection.

However, due to the vast search space of possible pairings, calculating the exact value of an EMD is computationally infeasible (Villani 2009) and various relaxation methods (Bertsekas 1985) are used to obtain approximate results quickly.

2.6.5 Data generation

The measure of *fidelity*, or *minimum matching distance* (MMD) (Achlioptas et al. 2018) describes the quality of created point clouds (either reconstructed or generated). It is computed as an average of distances between every point cloud in the

reference set and its nearest neighbour from the set of created shapes.

$$\mathrm{MMD}(\mathcal{X}, \tilde{\mathcal{X}}) = \frac{1}{|\mathcal{X}|} \sum_{\mathcal{P}_r \in \mathcal{X}} \min_{\mathcal{P}_c \in \tilde{\mathcal{X}}} \mathrm{dist}(\mathcal{P}_r, \mathcal{P}_c), \tag{2.49}$$

where $\mathrm{dist}(\cdot, \cdot)$ is the distance metric (e.g. Chamfer Distance or Earth Mover's Distance), $\mathcal{X}$ – reference set, $\tilde{\mathcal{X}}$ – created set and $|\cdot|$ denotes the cardinality of the set.

The *coverage* (COV) (Achlioptas et al. 2018) is a metric complementary to fidelity, that assesses the diversity of created point clouds. It is measured as a fraction of point clouds in the reference set that were matched as a nearest neighbour for at least one sample in the created set.

$$\mathrm{COV}(\mathcal{S}_r, \mathcal{S}_c) = \frac{\left|\left\{\arg\min_{\mathcal{P}_r \in \mathcal{S}_r} \mathrm{dist}(\mathcal{P}_r, \mathcal{P}_c) \mid \mathcal{P}_c \in \mathcal{S}_c\right\}\right|}{|\mathcal{S}_r|}, \tag{2.50}$$

with notation as in equation (2.49).

The *1-nearest neighbour accuracy* (1-NNA) (Lopez-Paz and Oquab 2017) is often used for comparing two empirical distributions (Xu et al. 2018). Let us assume $\mathcal{S}_r$ – reference set and $\mathcal{S}_c$ – created set. In this case, for each point cloud $\mathcal{P} \in \mathcal{S}_r \cup \mathcal{S}_c$, let us denote the set $\mathcal{S}_{-\mathcal{P}} = (\mathcal{S}_r \cup \mathcal{S}_c) \setminus \mathcal{P}$, as well as the closest neighbour $\mathcal{C}_\mathcal{P}$ of the sample $\mathcal{P}$ in the set $\mathcal{S}_{-\mathcal{P}}$. Then the nearest neighbour classifier classifies the point cloud $\mathcal{P}$ as to whether it is taken from the reference set $\mathcal{S}_r$ or the created set $\mathcal{S}_c$. The 1-NNA is therefore an accuracy of a such leave-one-out classifier over all samples in $\mathcal{S}_r \cup \mathcal{S}_c$, and can be written down as:

$$\text{1-NNA}(\mathcal{S}_r, \mathcal{S}_c) = \frac{1}{|\mathcal{S}_r| + |\mathcal{S}_c|} \left(\sum_{\mathcal{P} \in \mathcal{S}_c} I\left(\mathcal{C}_\mathcal{P} \in \mathcal{S}_c\right) + \sum_{\mathcal{P} \in \mathcal{S}_r} I\left(\mathcal{C}_\mathcal{P} \in \mathcal{S}_r\right) \right) \tag{2.51}$$

where $I(\cdot)$ is the indicator function with the rest of denotations as in equation (2.49).

For $\mathcal{S}_r$ and $\mathcal{S}_c$ coming from the identical distributions, the leave-one-out classifier should predict the $\mathcal{P}$ target class at random, i.e. $\text{1-NNA}(\mathcal{S}_r, \mathcal{S}_c) = 1/2$. For point cloud applications, the closest neighbour is usually picked using the Chamfer Distance or Earth Mover's Distance.

The advantage of calculating the 1-NNA over Jensen-Shannon Divergence lies in the simplicity of use. Contrary to JSD, where samples in the point cloud sets have to be discretised into voxels, the 1-NNA requires no preprocessing of the data.

2.7 Summary

The section 2.4 described state of the art for deep learning methodologies for processing 3-dimensional data [2]. The presented methods for point cloud data offer unsatisfactory results regarding the quality of extracted features and the ability to learn representations. At the time of research, there were no methods to generate arbitrary-sized point clouds and meshes. Moreover, the continual learning for 3-d data was underdeveloped and offered subpar classification accuracy during subsequent re-trainings. Therefore, it was necessary to develop new methods for representation learning on 3-dimensional point clouds, as described later in this thesis in chapters 3 to 6.

[2] The literature review was done continuously at the time of conducting the research described in this dissertation and may not contain the developments announced since then.

3 Representation learning for generative modelling

3.1 Research objective

This chapter describes the research on representation learning in the area of generative modelling, applied to the problem of point cloud generation.

Generative modelling of the data distribution is one of the key areas of machine learning, with application in out-of-distribution detection (J. Yang et al. 2021), computer vision (Creswell et al. 2018) and medical imaging (Ilse et al. 2020). However, due to its complexity, it usually requires the data to be low-dimensional or described with meaningful features (Murphy 2012). As mentioned in section 2.4.3, set-based representation of the point clouds makes them order-less, and the order of magnitude of a single cloud's possible representations grows at factorial speed with respect to its size. Therefore, extracting defining features in the point cloud-permutation invariant way is paramount to reducing the model complexity.

At the time of the publication, the generative approaches applied to point cloud data were based on the two-step models, consisting of training an autoencoder and subsequently training GAN (Goodfellow et al. 2014) on latent representations (Achlioptas et al. 2018). However, they did not offer a possibility to regularise the latent space or condition the generative process.

This section contains three approaches to generative modelling of point clouds. First, the 3dAAE (Zamorski et al. 2020a) – family of models based on adversar-

ial autoencoder architecture (Makhzani et al. 2016) – is presented. By utilising PointNet (Qi et al. 2017a) as the encoder network, 3dAAE offers feature space regularisations to an arbitrary probability distribution. Second, HyperCloud (Spurek et al. 2020), an extension of 3dAAE, by replacing the generator with the hypernetwork (Ha et al. 2017) that, instead of generating the shape directly, produces the weights for an additional network that transforms the point from the sampled 3-dimensional ball into the data space. This approach allows for generating point clouds consisting of an arbitrary number of points. Third, the CIF (Stypułkowski et al. 2021), a normalising flow (Dinh et al. 2015; Dinh et al. 2017) approach capable of conditional generation of shapes, and out-of-distribution detection by direct log-likelihood optimisation.s

3.2 Problem formulation

Let's consider a dataset $\mathcal{D}$ consisting of m point clouds, each represented as a set of n points, i.e.:

$$\mathcal{D} = \left\{ \mathcal{P}_i = \left\{ \mathbf{p}_j \in \mathbb{R}^3 \right\}_{j=1}^{n} \right\}_{i=1}^{m}. \tag{3.1}$$

The goal is to create a model able to embed the data into the d-dimensional latent space $\mathbb{Z} \subseteq \mathbb{R}^d$ and later generate the reconstruction based on this embedding. Therefore, the aim is to obtain functions called the encoder $E : \mathbb{X} \to \mathbb{Z}$ and the generator $G : \mathbb{Z} \to \mathbb{X}$, that minimise the reconstruction error between $\mathcal{P}$ and $G(E(\mathcal{P}))$.

Furthermore, one wishes to regularise the encoder such that the produced embeddings $\mathbf{z} \in \mathbb{Z}$ follow the prior distribution $P_{\mathbb{Z}}$ specified beforehand. After fitting the model, said regularisation would make shape generating possible by sampling from prior distribution to obtain the shape embedding $\tilde{\mathbf{z}} \sim P_{\mathbb{Z}}$ and passing it as an input to the generator, therefore obtaining a new shape $\tilde{\mathcal{P}} = G(\tilde{\mathbf{z}})$.

3.3 Methods

3.3.1 3dAAE – 3-d Adversarial Autoencoder

This section describes the introduced architecture named *3-dimensional Adversarial Autoencoder (3dAAE)*. Development of 3dAAE contributes to the research question concerning the generative modelling applied to representation learning on 3-dimensional point clouds that can be used for tasks such as shape generation, reconstruction, compression, and clustering. The detailed description of the research question is provided in section 1.2.2.

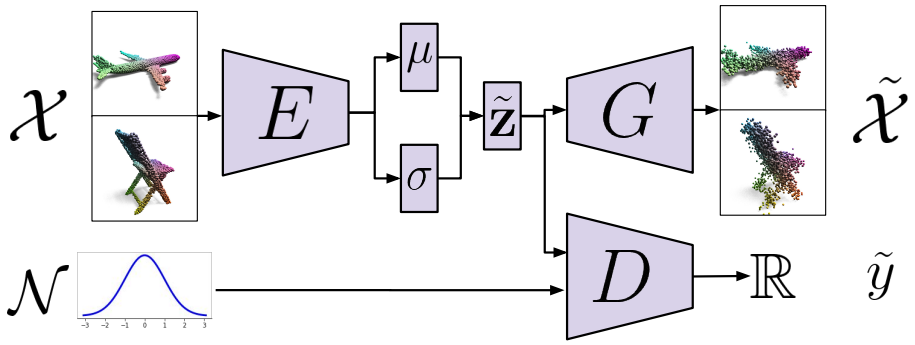

Figure 3.1: The architecture of the 3dAAE. The model consists of three neural networks: an encoder E, a generator G and a discriminator D. The encoder, implemented using PointNet, extracts two feature vectors interpreted as means μ and standard deviations σ from the input sample $\mathcal{X}$. After reparametrisation trick (3.2) they result in the shape representation $\tilde{\mathbf{z}}$. This representation is then a starting point of creating a reconstruction $\tilde{\mathcal{X}}$ by the generator G and is checked by the discriminator D for fit to the normal distribution.

Architecture The scheme of the 3dAAE architecture is presented in figure 3.1. The model is based on the Adversarial Autoencoder architecture, described in section 2.3.4. The 3dAAE is composed of three main modules:

- An encoder E implemented as PointNet (Qi et al. 2017a), extracting repre-

sentations $\tilde{\mathbf{z}}$ of input shapes $\mathcal{X}$[1]. The PointNet architecture is presented in figure 2.7. For the purpose of extracting representation, the *feature extraction* and *aggregation* parts are used. Therefore, it consists of four feed-forward neural networks – two TNets performing affine transforms in 3 and 64 dimensions and two multi-layer perceptrons (of sizes 64-64 and 64-128-h, where h is the size of the obtained representations), implemented as 1-dimensional convolutions. Each TNet is built out of six layers, three 1-d convolutional and three fully-connected, of sizes $64 - 128 - 1024 - 512 - 256 - k^2$, where k is the dimensionality of the affine transform the TNet models. Multi-layer perceptrons consist of $64 - 64$ and $64 - 128 - h$ layers. After every TNet and MLP layer, the 1-d batch normalisation and ReLu non-linearities are added, except for the last layer of the MLP2, containing the permutation-invariant `max` function and the last layers of the TNet models.

Moreover, to facilitate the regularization of obtained representation to the prior distribution $\mathcal{N}(0, \mathbf{I})$, two parallel layers μ and σ (each of size h) were added. By transforming the values of those layers with the so-called *reparameterisation trick*, it is possible to sample from the parameterised normal distribution while maintaining the ability for optimisation with backpropagation. The trick is defined as follows:

$$\tilde{\mathbf{z}} = \mu + \sigma\varepsilon, \tag{3.2}$$

where μ and σ are vectors of approximated posterior means and standard deviations, as presented in figure 3.1, and $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$. In the further part of the 3dAAE description, the reparameterisation trick will be omitted for brevity, and the encoding part will be denoted as $\tilde{\mathbf{z}} = E(\mathcal{X})$.

[1]In the earlier part of this dissertation, the point cloud shape was denoted as $\mathcal{P}$. However, the notation $\mathcal{X}$ will be used from here on out when describing point clouds from the perspective of their role as an input to the model. This adjustment aims to keep consistency with the rest of the machine learning literature, where the usual nomenclature for input data is x, for target label – y and hidden variables – z (with various cases and font styles).

- A generator G, constructing a sample $\tilde{\mathcal{X}}$ based on a provided representation $\mathbf{z}$ or $\tilde{\mathbf{z}}$. It is implemented as a feed-forward network consisting of five fully-connected layers of sizes $64 - 128 - 512 - 1024 - 3n$, where n is a number of generated points. After each layer (excluding the last) non-linear function is applied in the form of ReLU.

- A discriminator D, providing feedback to the encoder about the fit between posterior and prior distribution. It is implemented as a feed-forward network consisting of five fully-connected layers of sizes $512 - 512 - 128 - 64 - 1$. After each layer (excluding the last) ReLU non-linearity is applied.

Additionally, for comparison purposes, two more generative models are prepared:

1. *3dVAE* – a model similar to 3dAAE architecturally, with the exception of regularising the posterior to the standard normal distribution, is done with Kullback-Leibler divergence instead of a discriminator.

2. *3dAAE-GMM* – a variation of the 3dAAE model, in which the prior distribution is given as a mixture of 32 isotropic Gaussians instead of the standard normal distribution.

Adversarial training The standard training procedure of generative adversarial networks (Goodfellow et al. 2014) consists of a competition between two modules. One of the modules is network learning to approximate the data distribution. In the case of the 3dAAE, an encoder E is learning to produce samples from the normal distribution. On the other hand, the discriminator D tries to differentiate between samples from the true (i.e. prior) distribution and generated (i.e. posterior) distribution. This results in behaviour, defined in the game-theoretical terms as a two-player min-max game (Von Neumann and Morgenstern 2007). The value function for the game between those two actors can be written down as:

$$\min_E \max_D V_{\mathrm{GAN}}(E, D) = \mathbb{E}_{\mathbf{z} \sim P_{\mathbb{Z}}} \log D(\mathbf{z}) + \mathbb{E}_{\mathcal{X} \sim P_{\mathbb{X}}} \log(1 - D(\tilde{\mathbf{z}})), \tag{3.3}$$

where $P_{\mathbb{Z}}$ is a prior distribution and $\tilde{\mathbf{z}} = E(\mathcal{X})$ is an approximated posterior.

Based on the value function defined in the equation (3.3) above, the loss functions for the encoder E and the discriminator D are defined as opposites, i.e. V and $-V$, respectively. Additionally, the reconstruction objective is added for the encoder-generator pair to incorporate the generative training into this framework. Therefore, the loss functions for the adversarial training of 3dAAE could be defined as:

$$L_{E,G} = V_{\text{GAN}}(E, D) + \mathbb{E}_{\mathcal{X} \sim P_{\mathbb{X}}} \text{dist}(\mathcal{X}, \tilde{\mathcal{X}}) \tag{3.4}$$

$$L_D = -V_{\text{GAN}}(E, D), \tag{3.5}$$

where $\text{dist}(\cdot, \cdot)$ is the distance function between two sets (CD or EMD).

However, training GANs with optimisation objective defined in equation (3.3) leads to unstable training, with the loss values diverging rapidly once one of the sides will obtain an advantage over the other (Creswell et al. 2018; Zamorski et al. 2019), a phenomenon known as *mode collapse*. One of the leading solutions to improve training stability is to reformulate adversarial training using *Wasserstein* (or *Earth-Mover*) objective (Arjovsky et al. 2017) with *gradient penality* regularisation (Gulrajani et al. 2017). Wasserstein criterion changes the role of the discriminator[2] D from the function that judges from which distribution the input representation came from. Instead, the discriminator is interpreted as a 1-Lipschitz function, that follows the constraint:

$$|D(\mathbf{a}_1) - D(\mathbf{a}_2)| \leqslant 1 \cdot \|\mathbf{a}_1 - \mathbf{a}_2\|_2^2 , \tag{3.6}$$

where $\mathbf{a}_1, \mathbf{a}_2$ are any real-valued vectors of the same length.

The enforcement of the constraint (3.6) is performed with the gradient penalty regularisation. Since a differentiable function D is 1-Lipschitz if and only if it has

[2]In literature related to adversarial training using the Wasserstein objective, due to the different purpose it serves in the training process, the discriminator is referred to as *critic*. However, for the sake of consistency of names and symbols between solutions presented in this dissertation, the term *discriminator* will continue to be used instead.

gradients with the norm at most 1 everywhere, the penalty can be defined as:

$$GP(D) = \mathbb{E}_{\mathbf{z}\sim P_{\mathbb{Z}}}\mathbb{E}_{\mathcal{X}\sim P_{\mathbb{X}}}\left(\|\nabla D(\alpha\mathbf{z} + (1-\alpha)\tilde{\mathbf{z}})\|_2 - 1\right)^2, \tag{3.7}$$

where $\alpha \sim \mathcal{U}(0,1)$, $\tilde{\mathbf{z}} = E(\mathcal{X})$ and $\|\cdot\|_2$ is the L^2-norm of the gradient values. Based on Wasserstein distance between two distributions given in equation (2.48) the value function is defined as:

$$\min_E \max_{D\in\mathcal{L}_1} V_{\text{WGAN-GP}}(E, D) = \mathbb{E}_{\mathbf{z}\sim P_{\mathbb{Z}}} D(\mathbf{z}) - \mathbb{E}_{\mathcal{X}\sim P_{\mathbb{X}}} D(\tilde{\mathbf{z}}), \tag{3.8}$$

where $\mathcal{L}_1$ is a set of all possible 1-Lipschitz functions and the rest of the denotations as in equation (3.3). By adding terms for reconstruction objective and gradient penalty constraint, the explicit losses for discriminator and the encoder-generator pair are defined as follows:

$$L_{E,G} = -V_{\text{WGAN-GP}}(E, D) + \mathbb{E}_{\mathcal{X}\sim P_{\mathbb{X}}}\text{dist}(\mathcal{X}, \tilde{\mathcal{X}}) \tag{3.9}$$

$$L_D = V_{\text{WGAN-GP}}(E, D) + \lambda GP(D), \tag{3.10}$$

where λ is a scaling hyperparameter and $GP(\cdot)$ is the gradient penalty regularisation, as defined in the equation (3.7). The adversarial training procedure implementing the min-max game given in equations equations (3.9) and (3.10) is presented in algorithm 3.1.

3.3.2 3dAAE-Beta – 3dAAE for binary representations

The previous section described the solution for applying generative modelling to point cloud data, dubbed 3dAAE. By utilising an adversarial training, the discriminator regularised the representations produced by 3dAAE's encoder so that their distribution followed specified prior, given as normal distribution.

However, while effective, the real-valued features also have drawbacks. In comparison to binary features (and assuming the same number of features in both cases), real-valued representations require more storage space (16, 32 or 64 times, depending on the precision) and are much slower to process, as the arithmetic on the floating-point numbers is in general much more expensive than on the integers. On the other

Algorithm 3.1: The adversarial training procedure of 3dAAE. The Wasserstein GAN with gradient penalty regularisation is used to stabilise the min-max optimisation. Contrary to the usual GAN training, Wasserstein GAN performs several discriminator training iterations per one encoder-generator pair iteration.

In : $\mathcal{D} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$ – training samples, $P_{\mathbb{Z}}$ – prior distribution, $m \leqslant n$ – minibatch size, κ – number of discriminator iterations per one encoder-generator pair iteration, λ – gradient penalty weight

Out : $\theta_E^*, \theta_G^*, \theta_D^*$ – optimal parameters for encoder, generator and discriminator

$\theta_E, \theta_G, \theta_D \leftarrow$ Random initialisation

while *convergence not reached* **do**

  `// discriminator training`

  for $i \leftarrow 1$ **to** κ **do**

    $\mathcal{X} \leftarrow \mathcal{X} \subseteq \mathcal{D}$ `// sample minibatch` $\mathcal{X}$ `of size` m `from` $\mathcal{D}$

    $\tilde{\mathbf{Z}} \leftarrow E(\mathcal{X})$ `// encode minibatch samples`

    $\mathbf{Z} \leftarrow [\mathbf{z}_j \sim P_{\mathbb{Z}}]_{j=1}^{m}$ `// sample equal amount of embeddings from prior`

    $\boldsymbol{\alpha} \leftarrow [\alpha_j \sim \mathcal{U}(0,1)]_{j=1}^{m}$ `// sample cut-offs from the uniform dist.`

    $L_D \leftarrow D(\mathbf{Z}) - D(\tilde{\mathbf{Z}}) + \lambda \left(\|\nabla D(\boldsymbol{\alpha}\mathbf{Z} + (1-\boldsymbol{\alpha})\tilde{\mathbf{Z}})\|_2 - 1 \right)^2$ `//` (3.10)

    $\theta_D \leftarrow optimiser(\nabla L_D, \theta_D)$ `// update model weights`

  `// encoder-generator training`

  $\mathcal{X} \leftarrow \mathcal{X} \subseteq \mathcal{D}$ `// sample minibatch` $\mathcal{X}$ `of size` m `from` $\mathcal{D}$

  $\tilde{\mathbf{Z}} \leftarrow E(\mathcal{X})$ `// encode minibatch samples`

  $\mathbf{Z} \leftarrow [\mathbf{z}_j \sim P_{\mathbb{Z}}]_{j=1}^{m}$ `// sample equal amount of embeddings from prior`

  $\tilde{\mathcal{X}} \leftarrow G(\tilde{\mathbf{Z}})$ `// reconstruct input shapes`

  $L_{E,G} \leftarrow D(\tilde{\mathbf{Z}}) - D(\mathbf{Z}) + \mathrm{dist}(\mathcal{X}, \tilde{\mathcal{X}})$ `//` (3.9)

  $\theta_E, \theta_G \leftarrow optimiser(\nabla L_{E,G}, \theta_E, \theta_G)$ `// update model weights`

$\theta_E^*, \theta_G^*, \theta_D^* \leftarrow \theta_E, \theta_G, \theta_D$

return $\theta_E^*, \theta_G^*, \theta_D^*$

hand, calculating the distance between two binary vectors consists of just two machine instructions on almost all modern processors. Given that many point cloud applications, such as 3-d processing for autonomous vehicles, are time-sensitive, there is a need for an approach that extracts binary features from 3-dimensional data.

This section presents *3dAAE-Beta*, a modification to the 3dAAE model, that contributes to answering the research question about the method of obtaining the compact binary representations of 3-dimensional shapes, as stated in section 1.2.3.

Since it is impossible to obtain the binary representations from the PointNet encoder directly, the 3dAAE-Beta approach utilises a continuous approximation in the form of the Beta distribution.

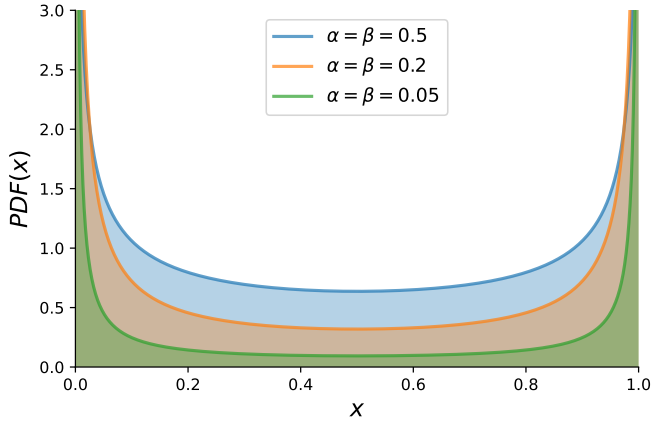

Figure 3.2: Probability density functions of Beta distribution, as given by equation (3.11) for the three sets of distribution parameters α, β. It can be observed that by setting $\alpha, \beta < 1$ the density function takes the "U" shape, putting most of the probabilistic density into regions close to (but excluding) 0 and 1, since $PDF(x) \to \infty$ as $x \to 0 \vee 1$.

Beta distribution Beta distribution is a continuous probability distribution parameterised by two shape parameters, α and β and defined on the $[0, 1] \subset \mathbb{R}$ or (if $\alpha < 1$ or $\beta < 1$) on $(0, 1) \subset \mathbb{R}$ interval. The probability density function of the Beta distribution is defined as:

$$PDF(x, \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}, \tag{3.11}$$

where $\alpha \in \mathbb{R}^+, \beta \in \mathbb{R}^+$ are shape parameters, and $B(\cdot, \cdot)$ is the beta function defined as in the equation (3.12) below:

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt, \tag{3.12}$$

where $\Gamma(\cdot)$ is the gamma function.

Figure 3.2 presents three examples of probability density functions of the Beta distribution. As can be observed, the closer the parameters α, β are to the zero, the

more probabilistic density is concentrated near values 0 and 1. Therefore, setting sufficiently low values will continuously approximate the binary random variable.

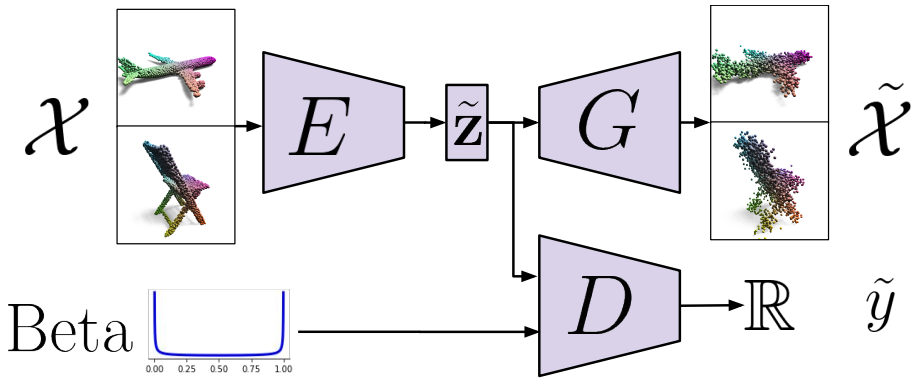

Figure 3.3: The architecture of the 3dAAE-Beta model. In comparison to the base 3dAAE architecture, presented in figure 3.1, this modification does not contain the μ and σ layers for distribution modelling. Prior distribution is modelled as Beta distribution with parameters $\alpha = \beta = 0.01$.

Architecture The architecture of the introduced approach is presented in figure 3.3. In comparison to the original 3dAAE architecture depicted in figure 3.1, 3dAAE-Beta does not consist of μ and σ layers of an encoder E, since in place of the Normal distribution, the Beta distribution with parameters $\alpha = \beta = 0.01$ is used instead. The choice of the parameter value 0.01 is motivated by the following lemma:

Lemma 3.3.1. *Given the random variable X that is beta-distributed with parameters $\alpha = \beta = 0.01$, the probability of sampling a value further than $r = 0.01$ from the probability support's boundaries is less than 5 %, i.e. $P(r < X \leqslant 1 - r) < 0.05$.*

Proof. The cumulative distribution function (CDF) for Beta distribution is given as follows:

$$CDF(x; \alpha, \beta) = \frac{B(x; \alpha, \beta)}{B(\alpha, \beta)} = \frac{\int_0^x t^{a-1}(1-t)^{b-1}dt}{B(\alpha, \beta)}, \tag{3.13}$$

where $B(\cdot; \alpha, \beta)$ is the incomplete beta function, and $B(\alpha, \beta)$ is the beta function

defined in equation (3.12). Therefore:

$$\begin{aligned} P(0.01 < X \leqslant 0.99) &= 1 - (P(X \leqslant 0.01) + P(0.99 < X)) \\ &= 1 - 2P(X \leqslant 0.01) \\ &= 1 - 2CDF(0.01) \\ &= 1 - 2 \cdot \frac{B(x = 0.01; \alpha = 0.01, \beta = 0.01)}{B(\alpha = 0.01, \beta = 0.01)} \\ &= 1 - 2 \cdot \frac{\int_0^{0.01} t^{-0.99}(1-t)^{-0.99} dt}{\int_0^1 t^{-0.99}(1-t)^{-0.99} dt} \end{aligned}$$

Numerical approximation using the WolframAlpha engine evaluates the expression above to:

$$P(0.01 < X \leqslant 0.99) \approx 1 - 2 \cdot \frac{95.5087}{199.9675} \approx 0.0448 < 0.05. \tag{3.14}$$

□

The training procedure for the 3dAAE-Beta model is conducted in a similar manner to the original 3dAAE approach, described in algorithm 3.1, with the difference lying in the choice of the prior distribution for regularisation. The generator module is trained on the representations consisting of the real-valued features close to 0 and 1. However, to evaluate the encoder and the generator's ability to work on actual binary representations, the randomly sampled prior and extracted posterior distribution embeddings will be processed according to the formula:

$$Bin(\mathbf{z}) = [\text{sgn}(2z - 1) \,|\, z \in \mathbf{z}], \tag{3.15}$$

where sgn$(\cdot)$ is a sign function returning ± 1. By representing the binary embeddings as ± 1 values, the model is able to utilise the entire range of parameters and activations during training and inference. In the case of the standard 0 and 1 encoding, all of the parameters that were multiplied by 0 (around half of them, since the model is regularised with symmetrical Beta distribution) would provide no information to the downstream modules.

3.3.3 3dAAE-C – 3-d Adversarial Autoencoder for unsupervised clustering

The previous section described the generative modelling approach to obtaining binary representations of 3-dimensional shapes. Taking this idea to the extreme would entail binary vectors with precisely one feature value set to one while the rest are being set to zero. In literature. this type of representations is called *one-hot encoding*. The most frequently used one-hot representations can be seen in classification and clustering tasks. Classes (or clusters) are then represented as one-hot vectors of size equal to the total number of classes (or clusters) with the bit's position set according to the class label value. For example, the third out of the ten classes would be represented as:

$$\mathbf{c} = [0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^T . \tag{3.16}$$

This section describes *3dAAE-C*, an extension of the 3d Adversarial Autoencoders that contributes to solving the problem of unsupervised clustering of the 3-dimensional point clouds, as stated in the research question in section 1.2.4. The extension is implemented by an additional posterior representation, regularised with the categorical distribution.

Categorical distribution Categorical distribution is a discrete probability distribution defined with probabilistic mass function:

$$PMF(x, c) = \begin{cases} \frac{1}{c} & \text{if } x \in \{1, 2, \ldots, c\} \\ 0 & \text{otherwise,} \end{cases} \tag{3.17}$$

where c is a number of classes.

However, using discrete distributions in neural network training is not feasible. This is because the argmax function, used to convert real-valued vectors into categorical ones, is non-differentiable. Therefore, in situations when sampling from the discrete distributions is required for the training of the machine learning model,

the often-times used solution is to apply the categorical reparameterisation using Gumbel-Softmax distribution (Jang et al. 2017).

Another advantage of the proposed approach is the ability to optimise for the categorical distribution directly, without the need to perform reparameterisation and sampling from the Gumbel distribution.

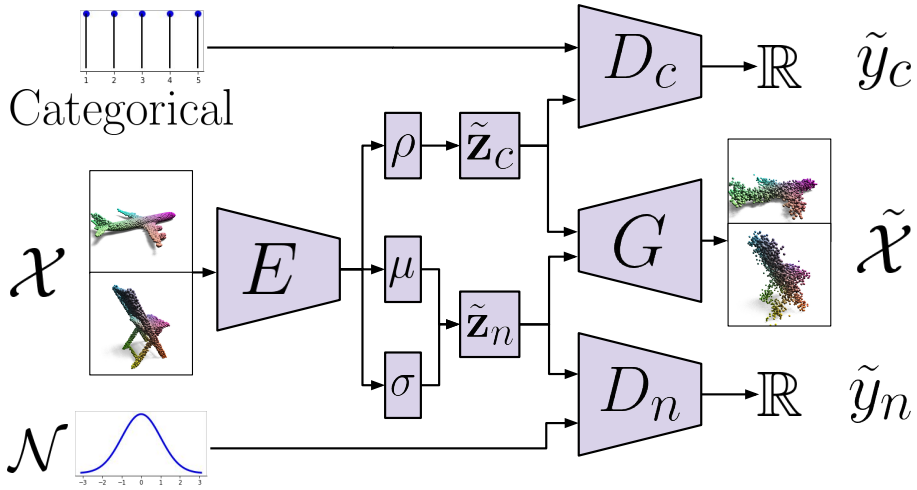

Figure 3.4: The architecture of the 3dAAE-C model. In comparison to the base 3dAAE version, presented in figure 3.1, this extension consists of an additional output $\tilde{\mathbf{z}}_c$ from an encoder E, as well as additional discriminator D_c that regularises $\tilde{\mathbf{z}}_c$ to the categorical prior distribution. For categorical distribution, every possible one-hot encoding is sampled with uniform probability.

Architecture The categorical distribution alone is not sufficiently complex to model the entire data distribution of 3-dimensional point clouds. Therefore, the 3dAAE-C, as presented in figure 3.4, extracts two types of representations simultaneously:

1. Real-valued representation $\tilde{\mathbf{z}}_n$ – an encoding of the 3-dimensional shape regularised to the Normal distribution with discriminator D_n, as presented in section 3.3.1.

2. Categorical representation $\tilde{\mathbf{z}}_c$ – an additional encoding regularised with the Categorical distribution with uniform probabilities on the bits.

For an easier approximation of one-hot vectors by the categorical part of the encoder, the additional fully-connected layer ρ, followed by the softmax activation, given as in equation (3.18) is applied on the $\tilde{\mathbf{z}}_c$ embedding.

$$\text{Softmax}\left(\tilde{\mathbf{z}}_c^{(i)}\right) = \frac{\exp\left(\tilde{\mathbf{z}}_c^{(i)}\right)}{\sum_{j=1}^{|\tilde{\mathbf{z}}_c|}\exp\left(\tilde{\mathbf{z}}_c^{(j)}\right)}, \text{ for } i = 1, 2, \ldots, |\tilde{\mathbf{z}}_c| \tag{3.18}$$

As the 3dAAE-C model is trained in an unsupervised manner, no additional constraints are put on the $\tilde{\mathbf{z}}_c$ extraction procedure. The only objective of the encoder, in terms of the distribution modelling, is to produce indistinguishable samples for the respective discriminators.

Training of the 3dAAE-C is a generalisation of the base 3dAAE approach to the adversarial game between encoder producing two outputs and two discriminators. Therefore, the value function $V_{\text{WGAN-GP}}$ extended for this approach can be written down as:

$$\begin{aligned}\min_{E}\max_{D_n, D_c \in \mathcal{L}_1} V_{\text{WGAN-GP}}(E, D_n, D_c) =& \mathbb{E}_{\mathbf{z}_n \sim P_{\mathbb{Z}_n}} D_n(\mathbf{z}_n) + \mathbb{E}_{\mathbf{z}_c \sim P_{\mathbb{Z}_c}} D_c(\mathbf{z}_c) \\ &- \mathbb{E}_{\mathcal{X} \sim P_{\mathbb{X}}} D_n(\tilde{\mathbf{z}}_n) - \mathbb{E}_{\mathcal{X} \sim P_{\mathbb{X}}} D_c(\tilde{\mathbf{z}}_c),\end{aligned} \tag{3.19}$$

where $\mathcal{L}_1$ is a set of 1-Lipschitz functions and the rest as in equation (3.3).

Based on the value function defined in equation (3.19), the loss functions for encoder-generator pair and discriminators pair is defined as follows:

$$L_{E,G} = -V_{\text{WGAN-GP}}(E, D_n, D_c) + \mathbb{E}_{\mathcal{X} \sim P_{\mathbb{X}}} \text{dist}(\mathcal{X}, \tilde{\mathcal{X}}) \tag{3.20}$$

$$L_D = V_{\text{WGAN-GP}}(E, D_n, D_c) + \lambda \left(GP(D_n) + GP(D_c)\right), \tag{3.21}$$

where λ is a scaling hyperparameter and $GP(\cdot)$ is the gradient penalty regularisation, as defined in the equation (3.7).

The detailed training procedure, implementing the objectives given by equations (3.20) and (3.21) is provided in algorithm 3.2.

Algorithm 3.2: The training procedure of 3dAAE-C model, based on the Wasserstein GAN with gradient penalty regularisation routine. In contrast to procedure provided in algorithm 3.1, the 3dAAE-C encoder produces two representation vectors $\tilde{\mathbf{z}}_n, \tilde{\mathbf{z}}_c$ that are regularised by two discriminators D_n and D_c simultaneously, inducing the normal and categorical distribution, respectively.

In : $\mathcal{D} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$ – training samples, $P_{\mathbb{Z}_n}, P_{\mathbb{Z}_c}$ – normal and categorical prior distributions, $m \leqslant n$ – minibatch size, κ – number of discriminator iterations per one encoder-generator pair iteration, λ – gradient penalty weight

Out : $\theta_E^*, \theta_G^*, \theta_{D_n}^*, \theta_{D_c}^*$ – optimal parameters for encoder, generator and discriminators

$\theta_E, \theta_G, \theta_{D_n}, \theta_{D_c} \leftarrow$ Random initialisation

while *convergence not reached* **do**

  `// discriminators training`

  for $i \leftarrow 1$ **to** κ **do**

    $\mathcal{X} \leftarrow \mathcal{X} \subseteq \mathcal{D}$ `// sample minibatch` $\mathcal{X}$ `of size` m `from` $\mathcal{D}$

    $\tilde{\mathbf{Z}}_n, \tilde{\mathbf{Z}}_c \leftarrow E(\mathcal{X})$ `// encode minibatch samples`

    $\mathbf{Z}_n \leftarrow [\mathbf{z}_j \sim P_{\mathbb{Z}_n}]_{j=1}^m$ `// sample` m `embeddings from normal prior`

    $\mathbf{Z}_c \leftarrow [\mathbf{z}_j \sim P_{\mathbb{Z}_c}]_{j=1}^m$ `// sample` m `embeddings from categorical prior`

    $\boldsymbol{\alpha} \leftarrow [\alpha_j \sim \mathcal{U}(0,1)]_{j=1}^m$ `// sample cut-offs from the uniform dist.`

    $L_{D_n} \leftarrow D_n(\mathbf{Z}_n) - D_n(\tilde{\mathbf{Z}}_n) + \lambda \left(\|\nabla D_n(\boldsymbol{\alpha}\mathbf{Z}_n + (1-\boldsymbol{\alpha})\tilde{\mathbf{Z}}_n)\|_2 - 1 \right)^2$

    $L_{D_c} \leftarrow D_c(\mathbf{Z}_c) - D_c(\tilde{\mathbf{Z}}_c) + \lambda \left(\|\nabla D_c(\boldsymbol{\alpha}\mathbf{Z}_c + (1-\boldsymbol{\alpha})\tilde{\mathbf{Z}}_c)\|_2 - 1 \right)^2$

    $L_D \leftarrow L_{D_n} + L_{D_c}$ `// equation (3.21)`

    $\theta_D \leftarrow optimiser(\nabla L_D, \theta_{D_n}, \theta_{D_c})$ `// update model weights`

  `// encoder-generator training`

  $\mathcal{X} \leftarrow \mathcal{X} \subset \mathcal{D}$ `// sample minibatch` $\mathcal{X}$ `of size` m `from` $\mathcal{D}$

  $\tilde{\mathbf{Z}}_n, \tilde{\mathbf{Z}}_c \leftarrow E(\mathcal{X})$ `// encode minibatch samples`

  $\mathbf{Z}_n \leftarrow [\mathbf{z}_j \sim P_{\mathbb{Z}_n}]_{j=1}^m$ `// sample` m `embeddings from normal prior`

  $\mathbf{Z}_c \leftarrow [\mathbf{z}_j \sim P_{\mathbb{Z}_c}]_{j=1}^m$ `// sample` m `embeddings from categorical prior`

  $\tilde{\mathcal{X}} \leftarrow G(\tilde{\mathbf{Z}}_n, \tilde{\mathbf{Z}}_c)$ `// reconstruct input shapes`

  $L_{E,G} \leftarrow \left(D_n(\tilde{\mathbf{Z}}_n) - D_n(\mathbf{Z}_n) \right) + \left(D_c(\tilde{\mathbf{Z}}_c) - D_c(\mathbf{Z}_c) \right) + \text{dist}(\mathcal{X}, \tilde{\mathcal{X}})$ `// equation (3.20)`

  $\theta_E, \theta_G \leftarrow optimiser(\nabla L_{E,G}, \theta_E, \theta_G)$ `// update model weights`

$\theta_E^*, \theta_G^*, \theta_{D_n}^*, \theta_{D_c}^* \leftarrow \theta_E, \theta_G, \theta_{D_n}, \theta_{D_c}$

return $\theta_E^*, \theta_G^*, \theta_{D_n}^*, \theta_{D_c}^*$

3.3.4 HyperCloud – Hypernetwork for 3-d point clouds

The proposed approaches constituting the 3dAAE family of models, described in sections 3.3.1 to 3.3.3 above, have one major drawback – generation of fixed-size point clouds, equal in size to the shapes at training time. This constraint is caused by the architectural design of the generator module, which is implemented as a fully-connected neural network.

To alleviate this shortcoming, this section proposes *HyperCloud*, the adversarial autoencoder and hypernetwork-based approach that provides the solution of representation learning for the generative modelling of point clouds consisting of an arbitrary number of points, as described in the research question provided in section 1.2.5. Instead of modelling the generative module with the network producing the fixed number of points, HyperCloud implements hypernetwork, i.e. network outputting parameters for another function, called *target*, which in turn transfers points from the prior distribution onto the generated shape surface.

Hypernetworks Hypernetworks (Schmidhuber 1992; Ha et al. 2017) consists of two parameterised functions, usually implemented as neural networks:

- *hypernetwork* (also called *generator network*) – the function that, given the input in the form of the one or more representations, produces the weights for another parameterised function
- *target network* – the parameterised function, in which parameters are not trained directly and are obtained from the hypernetwork output instead.

Therefore, the hypernetworks approach can be defined as generative modelling on the function space, allowing for obtaining a variety of smaller, but specialised to the given task target networks.

Architecture The HyperCloud approach combines two types of models: adversarial autoencoders and hypernetworks, as presented in figure 3.5. Similarly to 3dAAE,

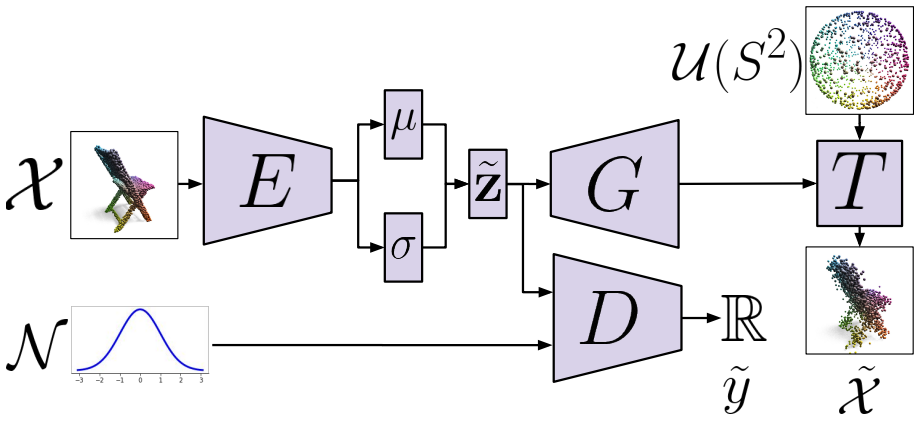

Figure 3.5: The architecture of the HyperCloud approach. The model is an extension of the 3dAAE, presented in figure 3.1, substituting a generator implemented as a fully-connected network for a one implemented as hypernetwork G and target network T pair. The target network produces output based on parameters provided by G and input points sampled from the uniform distribution on the unit sphere.

Algorithm 3.3: The training procedure of the HyperCloud approach. The following pseudo-code snippet substitutes line line 13 of algorithm 3.1.

1 $\theta_T \leftarrow G(\tilde{\mathbf{Z}})$ `// generate weights for` T `based on` $\mathbf{Z}$
2 $\mathcal{S} \leftarrow \{\mathcal{S}_j = \{\mathbf{s}_i \sim P_{S^2}\}_{i=1}^{k}\}_{j=1}^{m}$ `// sample points from shape prior`
3 $\tilde{\mathcal{X}} \leftarrow T_{\theta_T}(\mathcal{S})$ `// transform sampled priors` $\mathcal{S}$ `to reconstruct input`

the HyperCloud model employs a PointNet-like encoder for feature extraction from 3-dimensional point clouds. Moreover, the regularisation of obtained shape representation to the normal distribution prior is modelled with adversarial training.

As a generator, HyperCloud introduces a hypernetwork architecture, as described in section 3.3.4 above. Instead of generating a fixed-size point cloud (as was the case with the 3dAAE models), HyperCloud generates a set of weights for a target network. It is worth noting that thanks to processing each point individually, the hypernetwork approach can generate point clouds of arbitrary size, which constitutes the main improvement over the base 3dAAE model.

The training procedure of the HyperCloud approach, described in algorithm 3.3 consists of an adversarial training and data reconstruction. The loss functions for

the discriminator and the encoder-generator pair, on which the training is based, are as follows:

$$L_{E,G} = -V_{\text{WGAN-GP}}(E, D) + \mathbb{E}_{\mathcal{X}\sim P_{\mathbb{X}}}, \mathbb{E}_{\mathcal{S}\sim P_{S^2}} \text{dist}(\mathcal{X}, \tilde{\mathcal{X}} = T_{\theta=G(\tilde{\mathbf{z}})}(\mathcal{S})) \tag{3.22}$$

$$L_D = V_{\text{WGAN-GP}}(E, D) + \lambda GP(D), \tag{3.23}$$

where $V_{\text{WGAN-GP}}$ is a value function as defined in equation (3.8), dist is a distance metric between two sets (in this case EMD), $T_{\theta=G(\tilde{\mathbf{z}})}$ is a target network with parameters θ given by generator G based on an embedding $\tilde{\mathbf{z}}$ of shape $\mathcal{X}$, $\mathcal{S}$ is a set of points sampled from the surface of the S^2 sphere and GP is the gradient penalty regularisation, scaled by a hyperparameter λ.

3.3.5 CIF – Conditional Invertible Flow

Previous approaches relay on adversarial generative modelling for learning representations of the 3-dimensional shapes. While the Wasserstein criterion with gradient penalty regularisation significantly reduces the possibility of mode collapse during the training, it does not eliminate it completely.

An alternative approach assumes model optimisation with a standard log-likelihood criterion. By employing a model based on normalising flows, one is able to train an invertible function, embedding samples from the data distribution into the given prior distribution, making maximum likelihood estimation possible. Moreover, since the modelled function is a bijection, sampling new shapes is straightforward and done by passing the sample from the prior distribution as the input to the inverted function. The description of the generative modelling with normalising flows is provided in section 2.1.2.2.

This section describes *Conditional Invertible Flow (CIF)* networks, the normalising flow-based approach to the generative representation learning on 3-dimensional point clouds. The CIF networks contribute to the solution to the research question on representation learning for the generative modelling of point clouds consisting of an arbitrary number of points, as described in section 1.2.5. This model, similarly to

HyperCloud, allows sampling from the prior distribution of any number of embeddings, which will be transformed into points on the surface of the generated point cloud. Additionally, the whole architecture is possible to be trained with explicit likelihood optimisation.

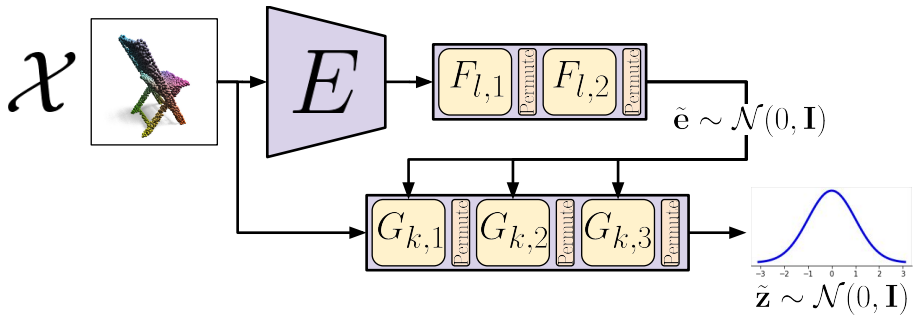

Figure 3.6: The architecture of the Conditional Invertible Flow networks. The model consists of three modules: an encoder E implemented as PointNet, conditioning flow F providing the shape representation that is normally-distributed and generating flow G, embedding input points into the normal distribution, based on a shape representation from F.

Architecture The Conditional Invertible Flow models point cloud shapes as probability densities in the 3-dimensional space. The architecture of the proposed approach is presented in figure 3.6 and consists of the three modules:

- The encoder module E implemented as a PointNet model for extracting feature vectors from input samples in an order-invariant way,
- The normalising flow F, taking PointNet features as an input and transforming them into encoded representation $\mathbf{e}$ that follows the standard normal distribution given as a prior,
- The normalising flow G, conditioned on features $\mathbf{e}$ returned by F, processing the points of an input cloud $\mathcal{X}$ into the standard normal distribution given as a prior.

Algorithm 3.4: The training procedure of the Conditional Invertible Flow networks. Contrary to the approaches presented in sections 3.3.1 to 3.3.4, the CIF optimisation criterion is defined in terms of negative log-likelihood minimisation and does not invoke the generation of reconstruction samples.

In : $\mathcal{D} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$ – training samples, $P_{\mathbb{Z}}$ – prior distribution, $m \leqslant n$ – minibatch size

Out : $\theta_E^*, \theta_F^*, \theta_G^*$ – optimal parameters for encoding, conditioning and generating modules

$\theta_E, \theta_F, \theta_G \leftarrow$ Random initialisation
while *convergence not reached* **do**
  $\mathcal{X} \leftarrow \mathcal{X} \subseteq \mathcal{D}$ `// sample minibatch` $\mathcal{X}$ `of size` m `from` $\mathcal{D}$
  $\mathcal{X}^{(\mathbf{g})}, \mathcal{X}^{(\mathbf{f})} \leftarrow \mathcal{X}$ `// randomly split samples from` $\mathcal{X}$ `into halves`
  $\tilde{\mathbf{e}} = (F \circ E)(\mathcal{X}^{(\mathbf{f})})$ `// obtain conditioning vectors in posterior dist.`
  $\tilde{\mathbf{z}} = G\left(\mathcal{X}^{(\mathbf{g})}, \tilde{\mathbf{e}}\right)$ `// obtain point representations in posterior dist.`
  $L \leftarrow -L_G\left(\mathcal{X}^{(\mathbf{g})}, \mathcal{X}^{(\mathbf{f})}\right) - L_{E,F}\left(\mathcal{X}^{(\mathbf{f})}\right)$ `// equations (3.24) to (3.26)`
  $\theta_E, \theta_F, \theta_G \leftarrow optimiser(\nabla L, \theta_E, \theta_F, \theta_G)$ `// update model weights`
$\theta_E^*, \theta_F^*, \theta_G^* \leftarrow \theta_E, \theta_F, \theta_G$
return $\theta_E^*, \theta_F^*, \theta_G^*$

The model is trained by minimising the following loss, defined as negative conditional log-likelihood:

$$L = -L_G\left(\mathcal{X}^{(\mathbf{g})}, \mathcal{X}^{(\mathbf{f})}\right) - L_{E,F}\left(\mathcal{X}^{(\mathbf{f})}\right) \tag{3.24}$$

where $\mathcal{X}^{(\mathbf{f})}, \mathcal{X}^{(\mathbf{h})}$ are two subsets sampled from a point cloud $\mathcal{X}$, $L_{E,F}(\cdot)$ is log-likelihood as defined in equation (3.25) and $L_G(\cdot,\cdot)$ is log-likelihood as defined in equation (3.26):

$$L_{E,F}(\mathcal{X}^{(\mathbf{f})}) = \log P_{\mathbb{Z}}(\tilde{\mathbf{e}}) + \log \left| \det \frac{\partial \tilde{\mathbf{e}}}{\partial E(\mathcal{X}^{(\mathbf{f})})} \right| \tag{3.25}$$

$$L_G(\mathcal{X}^{(\mathbf{g})}, \mathcal{X}^{(\mathbf{f})}) = \sum_{\mathbf{x} \in \mathcal{X}^{(\mathbf{g})}} \log P_{\mathbb{Z}}(\tilde{\mathbf{z}}) + \log \left| \det \frac{\partial \tilde{\mathbf{z}}}{\partial \mathbf{x}} \right|, \tag{3.26}$$

where $P_{\mathbb{Z}}(\cdot)$ – likelihood based on $\mathcal{N}(0, \mathbf{I})$, $\tilde{\mathbf{e}} = (F \circ E)(\mathcal{X}^{(\mathbf{f})})$ – extracted conditional embedding and $\tilde{\mathbf{z}} = G(\mathbf{x}, \tilde{\mathbf{e}})$ – extracted point embedding, conditioned on $\tilde{\mathbf{e}}$.

In order to generate a new point cloud $\tilde{\mathcal{X}}$ composed of n points, one needs to sample $n+1$ embeddings from the $\mathcal{N}(0, \mathbf{I})$ distribution. Next, one of the sampled

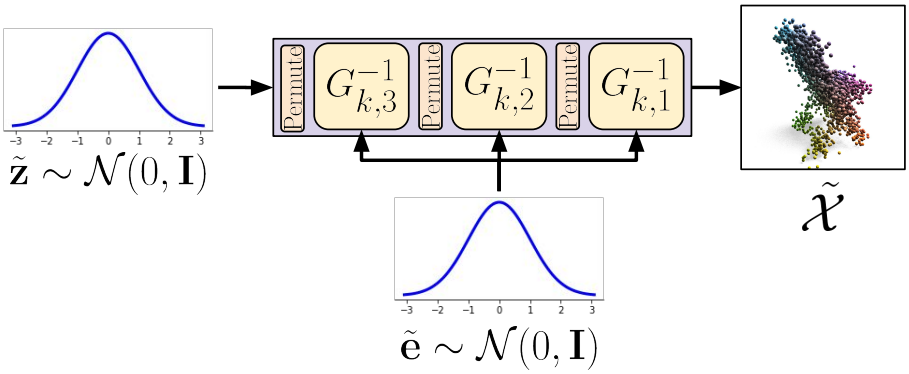

Figure 3.7: The point cloud generation procedure using the Conditional Invertible Flow networks. The shape representation $\tilde{\mathbf{e}}$ and point representations $\tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2, \ldots, \tilde{\mathbf{z}}_n$ are sampled from the standard normal distribution defined as a prior. Then, each point encoding is processed by an inverse normalising flow G^{-1} conditioned on $\tilde{\mathbf{e}}$ into the data points distribution.

embeddings is used as a conditioning vector $\mathbf{e}$, while the rest is passed as point representations $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$ to the inverted flow G^{-1}.

3.4 Experiments

This section describes the results obtained during the quantitative and qualitative evaluation of the proposed architectures for generative modelling of point cloud shapes. The conducted experiments evaluate the capabilities of the models and obtained representations to estimate data density when reconstructing and generating new data as well as separability of the latent representation space.

The experiments were conducted on the following point cloud datasets:

1. ModelNet 40 (Z. Wu et al. 2015) – as described in the section 2.5,
2. ShapeNet (Chang et al. 2015) – as described in the section 2.5.

Each shape was preprocessed by sampling 2048 points from the provided mesh's surface. Additionally, every point cloud was augmented by being randomly rotated around $\vec{z}$ (vertical) axis, and had added noise $\varepsilon \sim \mathcal{N}(\mu = 0, \sigma = 0.01)$ during

Table 3.1: Reconstruction capabilities of proposed generative models trained with earth mover's distance measured as minimum matching distance on the test split of the chair class. Autoencoder and l-GAN results from (Achlioptas et al. 2018). PointFlow results from (G. Yang et al. 2019).

Method	MMD-CD (↓)	MMD-EMD (↓)
AE	13.00	5.20
l-GAN	8.85	5.26
PointFlow	**7.54**	5.18
3dVAE	10.00	5.20
3dAAE	9.00	5.20
3dAAE-GMM	8.00	**5.10**

training. No augmentation was performed for the validation and test splits of the datasets.

Details of the metrics and distance functions, that are used for evaluation, are provided in sections 2.6.4 and 2.6.5.

3.4.1 Reconstruction capabilities

Before evaluating the ability of proposed approaches to generate data from the representations sampled from the prior distribution, the models are assessed in a more straightforward scenario. In this experiment, the proposed models – 3dVAE, 3dAAE and 3dAAE-GMM (Gaussian Mixture prior) – are tested on reconstruction task, i.e. data generation based on the representation extracted from the input 3d shape. This test checks for generalisation to the unseen, real data. Low reconstruction fidelity (i.e. high minimum matching distance) scores may suggest over-regularisation of the model or models. Over-regularization is an issue of sacrificing reconstruction and representation quality, favouring fitness to the prior distribution. An encoder has failed to embed data instance features into the hidden vector when the prior and

posterior distributions are indistinguishable. Control for over-regularisation is usually done by monitoring the statistics of posterior distributions during training and then manually balancing the reconstruction and regularisation losses by introducing a hyperparameter (Higgins et al. 2016).

Table 3.1, presents the obtained minimum matching distances (see section 2.6.5) on test data reconstructed by Autoencoder, 3d Variational Autoencoder, and 3d Adversarial Autoencoder with standard normal and a mixture of normals prior distributions. It can be noted that none of the proposed models suffers from an over-regularisation problem. The results show significant improvement of the fidelity of the reconstructions compared to the baseline autoencoder. Minimum matching distance scores are comparable with other methods from the literature, despite using a much simpler, 1-stage architecture.

3.4.2 Generation capabilities

This section contains the evaluation of representations modelled by the proposed generative approaches in application to the generation tasks. The representations are measured with Jensen-Shannon divergence, minimum matching distance (MMD, also named fidelity), coverage (COV) and 1-NNA, with MMD and coverage calculated using both chamfer distance and earth mover's distance.

The parameters for models are selected based on the Jensen-Shannon Divergence on the validation split of the dataset measurement on the validation set. However, depending on the architecture, the evaluation approach slightly differs:

1. 3dAAE methods – to reduce the sampling bias, each generator produces a set of synthetic samples thrice the population of the comparative set (i.e. test or validation). It is denoted with grey background in table 3.2.

2. HyperCloud, CIF – each generator produces a number of samples equal to the population of the comparative set. It is denoted with neutral background in table 3.2.

Table 3.2: The results of generative modelling experiments for proposed approaches (3dAAE, HyperCloud, CIF) compared to relevant methods from the literature (l-GAN, PointFlow). Values on the grey background indicate results obtained by generating from the model three times the number of samples compared to the number of shapes in the reference dataset. Values on the neutral background indicate results from the experiments involving equinumerours generated and reference data, hence lower overall scores. The best value is marked in bold separately for grey and neutral backgrounds. The results are scaled by: MMD-CD $\times 10^3$, MMD-EMD $\times 10^2$, JSD $\times 10^2$.

Category	Methods	JSD (↓)	MMD (↓)		COV (%, ↑)		1-NNA (%, 50)	
			CD	EMD	CD	EMD	CD	EMD
Airplane	l-GAN	**3.61**	0.269	3.29	**47.90**	50.62	87.65	85.68
	PointFlow	4.92	**0.217**	3.24	46.91	48.40	**75.68**	75.06
	HyperCloud	4.84	0.266	3.28	39.75	43.70	93.80	88.95
	CIF	4.24	0.221	**3.14**	47.57	**52.67**	77.08	**72.59**
	Training set	6.61	0.226	3.08	42.72	49.14	70.62	67.53
Chair	l-GAN	2.00	1.80	6.50	68.90	67.40	-	-
	3dAAE	**1.40**	**1.70**	**6.22**	67.30	67.00	-	-
	3dAAE-GMM	**1.40**	**1.70**	6.43	**69.60**	**68.70**	-	-
	l-GAN	2.27	2.61	7.85	40.79	41.69	64.73	65.56
	PointFlow	1.74	2.42	7.87	**46.83**	46.98	**60.88**	**59.89**
	HyperCloud	2.73	2.56	**7.84**	41.54	46.67	68.20	68.80
	CIF	**1.42**	**2.38**	7.85	44.01	**47.03**	62.71	63.39
	Training set	1.50	1.92	7.38	57.25	55.44	59.67	58.46
Car	l-GAN	-	-	4.10	-	65.30	-	-
	3dAAE	-	-	4.00	-	66.20	-	-
	3dAAE-GMM	-	-	**3.90**	-	**67.60**	-	-
	l-GAN	2.21	1.48	5.43	39.20	39.77	69.74	68.32
	PointFlow	0.87	0.91	5.22	44.03	46.59	**60.65**	62.36
	HyperCloud	1.07	1.14	5.30	**45.74**	47.44	64.63	62.78
	CIF	**0.79**	**0.90**	**5.12**	44.79	**49.24**	64.82	**61.36**
	Training set	0.86	1.03	5.33	48.30	51.42	57.39	53.27

The difference is cased by the paradigm shift in the literature concerning the common way of evaluating the generative performance of the model, between the time

(Zamorski et al. 2020a) and (Spurek et al. 2020; Stypułkowski et al. 2021) was published. While (Achlioptas et al. 2018) proposed the evaluation based on the former style, the work (G. Yang et al. 2019) standardised using the the latter. Due to the different approaches to evaluation, the results reported for l-GAN in table 3.2 are provided twice, once for each evaluation methodology. The presented results are the averages after repeating the process three times.

Table 3.3: Results of point cloud retrieval and embedding classification with Linear SVM on ModelNet 40 dataset. The 3dAAE model has been trained with initial value of a hyperparameter $\lambda = 2.0$ that was exponentially decayed as the training proceeds. Its purpose is to keep a balance between reconstruction and adversarial losses, as demonstrated in (Higgins et al. 2016).

Method	Continuous	Binary	
	Accuracy	Accuracy	mAP
3dGAN	83.30	-	-
l-WGAN	84.50	-	-
PointFlow	86.80	-	-
AE	84.85	78.12	41.76
3dAAE	84.35	79.78	44.09
HyperCloud	84.70	-	-

Table 3.2 contains generation results achieved by the proposed 3dVAE, 3dAAE, 3dAAE-GMM, HyperCloud and CIF architectures, measured by fidelity, coverage and 1-NNA metrics on different classes from the ShapeNet dataset. They are compared against the relevant methods from the literature, namely (Achlioptas et al. 2018) and (G. Yang et al. 2019). Additionally, the memorisation model is provided as a reference result, in which "generating" consists of sampling the required amount of data from the training set.

It can be noted that those models match or surpass the state-of-the-art results in

all of the metrics while offering desired properties, such as generating an arbitrary number of points or conditional data generation.

In addition to regularising the 3dAAE approach with Gaussian-based prior distributions, the Beta($\alpha = \beta = 0.01$) distribution was applied to impose binarisation on the embeddings during training. Table 3.3 contains results for two proposed models on retrieval and classification (by using LinearSVM on learned representations) tasks using real-valued and binary embeddings. The first model has been trained with $\lambda = 2.0$, the best value found in the ablation studies, where λ is a balance between reconstruction and adversarial losses. The second model uses an exponential decay to reduce λ as the training proceeds. It can be observed that, while the binary embedding causes a drop in classification accuracy compared to real-valued representations, the mean average precision (see section 2.6.2) improves in comparison to binarised features from standard autoencoder.

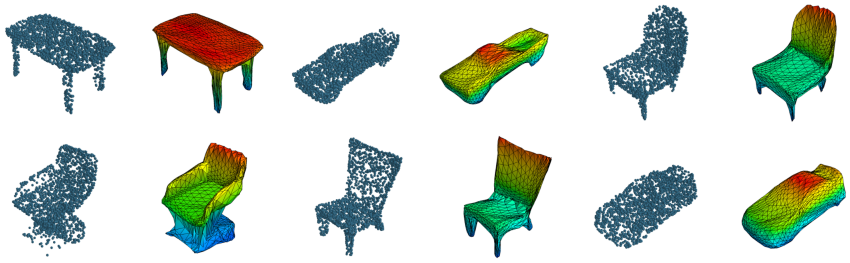

Figure 3.8: The samples of the generated point clouds and mesh-based shapes from the HyperCloud model. The samples shown above belong to table, chair and car categories. Each cloud (mesh) consists of 2048 sampled points (vertices). The shapes are obtained by processing 2048 points sampled from the surface of the S^2 sphere (for point clouds) or 2048 vertices from generated triangle mesh of the sphere.

In addition to the quantitative results presented above, figures 3.8 and 3.9 present qualitative samples of generative modelling capabilities of the proposed approaches. The point clouds are randomly generated by HyperCloud and CIF networks based on the embeddings sampled from the respective prior distributions. It can be observed that the quality of produced samples is similar to true data, and the mismatched points happen sporadically.

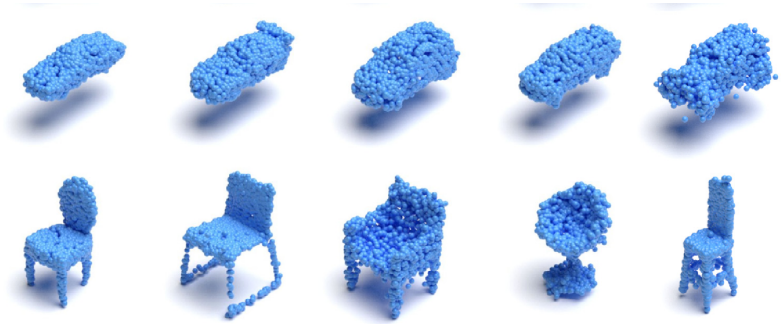

Figure 3.9: Samples of the generated point clouds from the Conditional Invertible Flow networks model, obtained with G normalising flow. Rows show samples from car and chair categories. Each cloud consists of 2048 sampled points.

Moreover, the figure 3.8 represents the ability of HyperCloud to generate realistic, watertight meshes. Notably, the colour-coding of the meshes' triangles suggests that vertices initially close to each other stay that way after transformation. It shows that produced meshes contain smooth surfaces without generation artefacts, such as holes.

3.4.3 Latent space coverage

One of the characteristics of a good generative model is producing realistic samples based on interpolated representations. By generating samples based on such embeddings, it is possible to qualitatively assess whether the generative model smoothly covers the joint data and representation distributions. The concept is visualised in figure 3.10. In this figure, one can notice the interpolation gap between two chairs

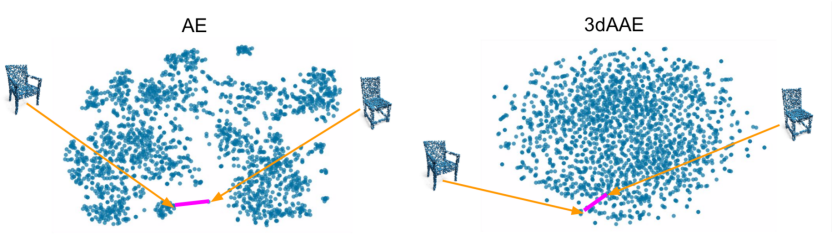

Figure 3.10: The t-SNE plot of the latent spaces obtained from AE and 3dAAE models.

for AE. On the other side, the encodings obtained from the 3dAAE model do not suffer from this phenomenon and allow for smooth transition within the data space with a much denser latent space. This confirms that the model learns to embed meaningful semantic features into the hidden vector. Furthermore, such a result shows generalisation capability beyond the samples from the training dataset. This section presents the results of data generation by transitioning underlying representations from one state to another. The results are provided for 3dAAE, HyperCloud and CIF architectures for real-valued representations and for 3dAAE-Beta for binary ones.

Figure 3.11 shows that the interpolation of latent representations produces smooth transitions between two distinct types of objects belonging to the same class. The 3dAAE model is able to change multiple characteristics of objects at once, as can be seen with the shape and legs of the table.

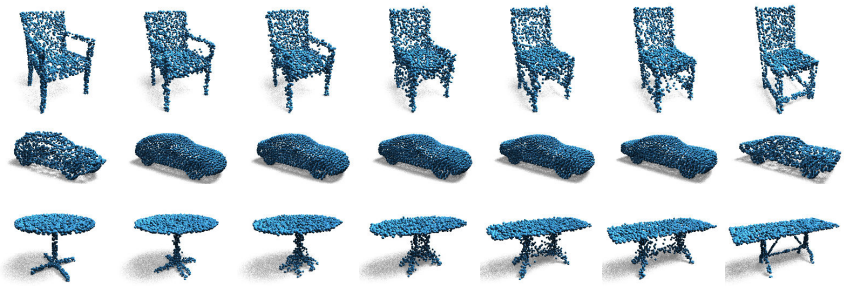

Figure 3.11: Shape interpolations generated by 3dAAE. The leftmost and rightmost objects are samples from the test split of the ShapeNet dataset. Point clouds in between are the result of generating shapes from the linear interpolation between the representations of the samples from the test set.

A similar experiment has been performed on compact, binary embeddings. Contrary to the real-valued representations, it is impossible to perform smooth interpolation over every latent dimension simultaneously. Therefore, the transition between two binary representations $\tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2$ was performed by calculating the difference (binary `xor` operation) between representations of two shapes from the test split of the dataset. Next, the places where binary representations $\tilde{\mathbf{z}}_1$ and $\tilde{\mathbf{z}}_2$ were different were

split into s equinumerous[3] steps, which can be written as:

$$\Delta_b = \frac{1}{\sum_i^{|\tilde{\mathbf{z}}_1|} \left(\tilde{\mathbf{z}}_1 \oplus \tilde{\mathbf{z}}_2\right)^{(i)}} \times 100\%, \tag{3.27}$$

where Δ_b is a percentage of bits that change between two representations and $\oplus$ is a binary `xor` operation.

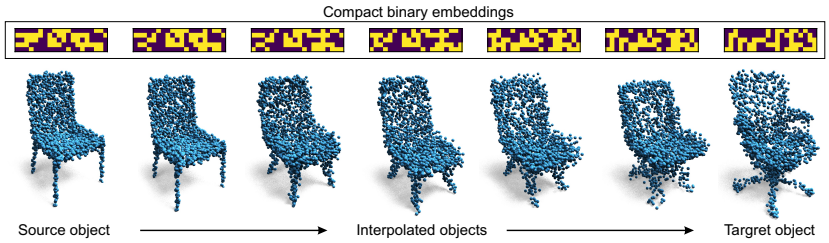

Figure 3.12: Compact binary representations (100 bits) of 3-d point clouds. For each of the 3-d shapes, we provide corresponding binary codes.

The visualisation of binary interpolation and data generation results from those interpolated representations are shown in figure 3.12. Although the transition in binary space is ultimately limited in regard to the number of possible interpolation steps, the presented results show smooth transposition between two chair shapes even when generating point clouds from the compact binary representation.

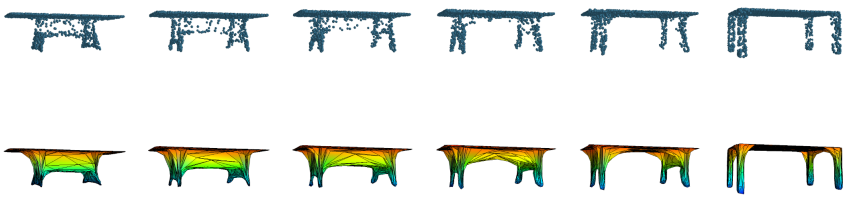

Figure 3.13: Interpolations between two 3-d point clouds and their mesh representations.

The interpolation experiments have also been conducted for the HyperCloud, and CIF networks approaches. The qualitative results are provided in the figures 3.13 and 3.14. In the case of the mesh generation with HyperCloud, it can be observed that the generative capabilities of the model generalise to producing coherent meshes on a smooth cover of the representation space.

[3]With possible difference of 1, due to the integer division.

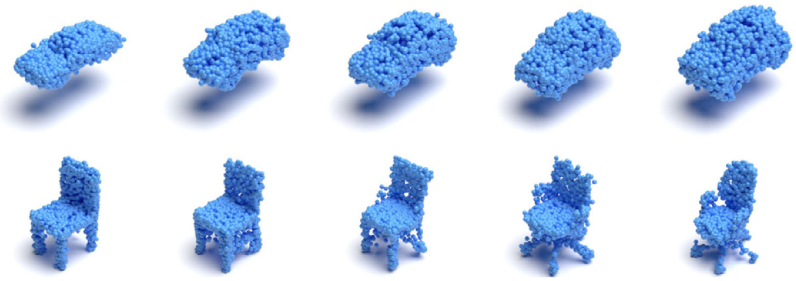

Figure 3.14: Interpolation of latent vectors **z** makes smooth transition between their reconstructions (from left to right). Each row shows interpolation on different classes of shapes.

3.4.4 Disentangled and abstract representations

Section 2.2 provided disentanglement and abstraction of features as some of the traits of good representations. By performing the linear algebra on the extracted embeddings, it can be deduced if the encoder learned to put semantically similar features in the same latent regions and if those regions only code one factor of the data.

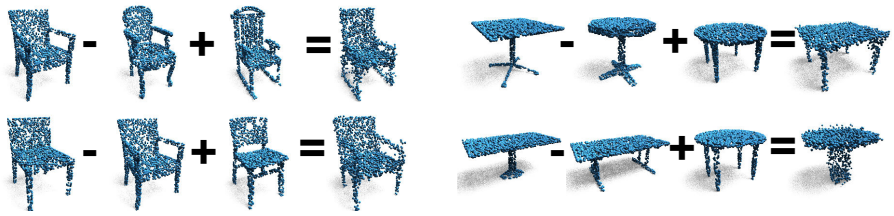

Figure 3.15: Modifying point clouds by performing additive algebra on latent space encodings.

The figure 3.15 presents 3dAAE ability to learn abstract representations that allow to perform addition and subtraction in latent space. Those operations modify existing point clouds according to the differences between point clouds in the equation while leaving other characteristics unchanged. The example shows adding rockers and armrests to chairs and changing the shape and legs of the table.

Figure 3.16 presents clustering results of the 3dAAE-C model trained with a separate regularization to 32-dimensional one-hot distribution. Presented are three

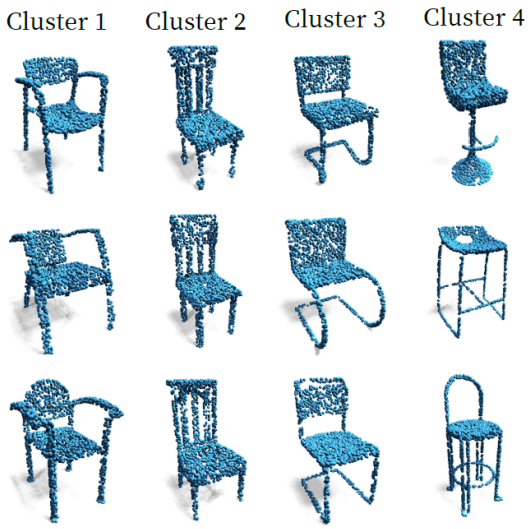

Figure 3.16: Selected examples from test set clustered with adversarial autoencoder with an additional categorical unit.

randomly selected representatives from the four most dominant clusters. It can be seen that among the detected clusters, the chairs contain characteristics for the subgroup features and shapes.

3.5 Discussion

This chapter presents the contributions to the area of representation learning in application to generative modelling of 3-dimensional point clouds. The first section motivated research in this area. Next, section 3.3 describes original contributions during the course of PhD studies. Sections 3.3.1 to 3.3.3 present approaches I have introduced, namely 3-dimensional Adversarial Autoencoder regularised with normal distribution along with its extensions, regularised with beta and categorical distributions. These models aim to answer research questions concerning unsupervised training of representations (with real-valued or binary features) with an application to data distribution estimation and clustering, as stated in sections 1.2.2 to 1.2.4. Moreover, in sections 3.3.4 and 3.3.5 I present the methods that I have collaborated on, namely HyperCloud, and Conditional Invertible Flow network, which relax the

restriction of generating the same number of points every time, as described in section 1.2.5. The chapter is concluded with section 3.4, which discusses quantitative and qualitative results obtained by proposed architectures.

4 Representation learning for continual learning

4.1 Goal of the studies

This chapter describes the results of research related to the application of the proposed method Random Compression Rehearsal (RCR) in the domain of continual learning of deep learning models on 3-dimensional data. A rehearsal technique of point cloud undersampling is introduced as a way to re-train machine learning architectures.

The growing popularity of deep learning and its applications to increasingly complex tasks and a vast amount of data collected result in machine learning models of a rapidly rising number of parameters. While some problems require simple approaches, most of the usages evolve or are being redefined during the model's life cycle. Therefore, incorporating the new knowledge by re-training the model is a vital dilemma from the computational, performance, and storage standpoint. Simply re-training on a new dataset causes the model's performance on the original data rapidly degrade, a phenomenon known as *catastrophic forgetting*. Training the model from scratch whenever new data is available is not always possible due to time or monetary constraints. Consequently, it is paramount to arrive at a solution that can continuously train machine learning models with minimal computational and storage overhead.

The solution introduced in this section contributes to the problem of learning data

representations for catastrophic forgetting mitigation during continual training on point cloud shapes, as stated in the research question provided in section 1.2.6. The existing methods for continual classification were constructed for an application to the image data domain and therefore offer inadequate performance when used on 3-dimensional data. By tailoring the proposed solution to point cloud datasets, it is possible to achieve high accuracy and efficiency. To minimise the proposed solution overhead, it was necessary to achieve a small additional cost in terms of extra computation and storage required for RCR to work while maintaining the satisfactory quality of classification accuracy.

4.2 Problem formulation

Let's consider a machine learning model, trained on a dataset $\mathcal{D}_1$, consisting of m pairs of data samples $\mathcal{X} \in \mathbb{X}$[1] and corresponding class labels $y \in \mathbb{Y}_1 \subset \mathbb{N}$, i.e.

$$\mathcal{D}_1 = \{(\mathcal{X}_m, y_m)\}_{m=1}^{M}. \tag{4.1}$$

The goal is to re-train the model on the new dataset $\mathcal{D}_2$, that contains samples belonging to a different class domain $\mathbb{Y}_2 \subset \mathbb{N}$, i.e.

$$\mathbb{Y}_1 \cap \mathbb{Y}_2 = \varnothing, \tag{4.2}$$

without degrading the classification performance on samples from the original dataset $\mathcal{D}_1$. Each dataset $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_t$ that the model was trained on, will be referred to as a *task*.

The research will be evaluated on two continual learning scenarios, as described in (Ven and Tolias 2019) – *task incremental learning* (*Task-IL*, i.e. solve task, knowing beforehand which one it is) and *class incremental learning* (*Class-IL*, i.e. solve the task after inferring it).

Several approaches to mitigate catastrophic forgetting have been proposed, including the usage of progressive growing of neural network architecture (Rusu et al.

[1] See footnote 1 on page 52.

2016), regularising re-training with soft labels (Zhizhong Li and Hoiem 2017) as well as restricting the gradient on specific parameters that are vital for previous tasks (Kirkpatrick et al. 2017). The other popular direction employs the replay and rehearsal approaches, in which re-training is supplemented with the past data provided by generative model (Shin et al. 2017) or by storing a portion of original data respectively (Rebuffi et al. 2017).

However, all of those methods were developed with the intention of training models on the image data. Some of the very few continual learning approaches for point clouds are extensions of existing methods (Chowdhury et al. 2021; Dong et al. 2021).

4.3 Methods

This section presents the proposed architecture of Random Compression Rehearsal. The aim is to create a model that is able to capture general and high-level, as well as discriminative and low-level features of 3D point cloud shapes. While detailed features are fundamental to training any classifier, extracting the all-around representation provides twofold benefits:

- regularisation method – the application of autoencoder architecture that shares the latent space and uses reconstruction loss, prevents classifier from overfitting,
- future-proofing – mitigates catastrophic forgetting by conditioning classifier on features that may be useful in future tasks.

Discriminative modelling and representation learning are employed jointly to ensure the embedding of both types of features described above. Discriminative modelling allows learning specific, low-level features, while the representation learning approach focuses more on the broad properties of the data samples.

The RCR consists of three parameterised functions, implemented as neural networks (see architecture schema in figure 4.1).

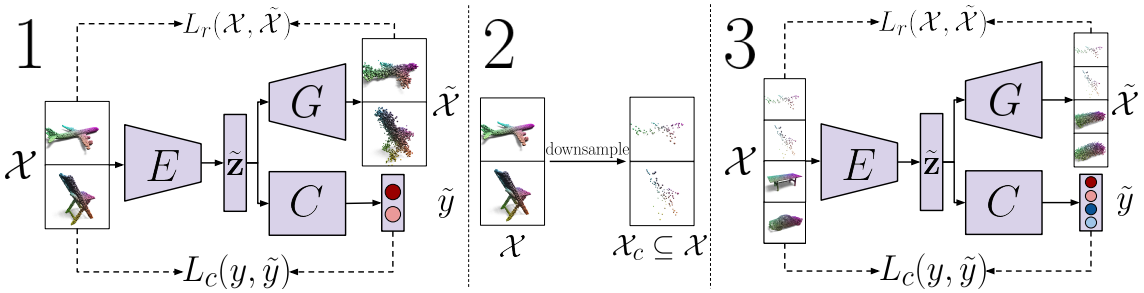

Figure 4.1: The architecture and the training procedure of Random Compression Rehearsal. The RCR consists of three neural networks: an encoder E, a generator G and a classifier C. The model trains by solving in tandem the classification and reconstruction task, as shown in step 1. The data used for the current task is then sub-sampled (step 2) and stored for rehearsal purposes, as presented in step 3.

- An encoder E, implemented as a PointNet module, extracting representations $\tilde{\mathbf{z}}$ from the input sample $\mathcal{X}$. Detailed description is provided in section 3.3.1.

- A generator G, constructing a sample $\tilde{\mathcal{X}}$ based on an encoded representation $\tilde{\mathbf{z}}$. It is implemented as a feed-forward network consisting of five fully-connected layers of sizes $64 - 128 - 512 - 1024 - 3n$, where n is a number of generated points. After each layer (excluding the last) non-linear function is applied in the form of ReLU.

- A classifier C, predicting the class of the input sample $\mathcal{X}$ based on extraction representation $\tilde{\mathbf{z}}$. It is implemented as a feed-forward network consisting of three fully-connected layers of sizes $512 - 256 - 2t$, where t is the number of the current task. After each layer (excluding the last) batch normalisation, ReLU non-linear function and dropout are applied. When first created (i.e. for the task $t = 1$) the classifier's last layer consists of only two outputs (figure 4.1, step 1). When expanding the last layer for re-training, additional outputs are created (figure 4.1, step 3). It is implemented by adding randomly initialised rows to the last parameter matrix and extending the corresponding bias vector with zero values.

The training procedure of the Random Compression Rehearsal is outlined in figure 4.1. The course of learning is as follows:

1. First, the model is trained to simultaneously reconstruct and classify an input shape $\mathcal{X}$. Those tasks are performed based on features $\tilde{\mathbf{z}}$ extracted by PointNet backbone E. The input shape reconstruction $\tilde{\mathcal{X}}$ is obtained using generator G, and prediction of class label y is acquired with classifier C.

   The optimisation criteria consists of minimising a reconstruction loss L_r and classification loss L_c, as defined in equation (4.4), based on outputs $\tilde{\mathcal{X}}$ and $\tilde{y}$.

2. Next, the training dataset is compressed for future rehearsal use. The compression is performed by uniformly selecting a subset of each point cloud $\mathcal{X}$ in the training dataset $\mathcal{D}$. Each compression sample from the input cloud $\mathcal{X}$ a fixed number of points without replacement. Then, the new dataset $\mathcal{D}_c$ is created by pairing the compressed samples with the corresponding class labels. Whole operation can be written down as:

$$\mathcal{D}_c = \left\{ \mathcal{X}^{(\mathbf{c})} \subset \mathcal{X}, y) \mid (\mathcal{X}, y) \in \mathcal{D} \right\}. \tag{4.3}$$

3. For re-training, the classification layer of the classifier C is expanded to accommodate new class labels. Then, the new dataset $\mathcal{D}_{new}$ is merged with the compressed dataset $\mathcal{D}_c$ obtained in the previous step. The training procedure is then conducted in the matter described in point 1, but based on the joined dataset $\mathcal{D} = \mathcal{D}_{new} \cup \mathcal{D}_c$.

4. Afterwards, the compressed dataset $\mathcal{D}_c$ is extended by the samples from the new dataset $\mathcal{D}_{new}$ in the manner described in step 2.

As mentioned above, the classification module tends to benefit from more detailed features, while the reconstruction counterpart focuses more on the overall shape of the data samples. Therefore encoder extracts latent representations of the given 3D shapes with focus balancing between discriminative and more general features.

Thus, the combined optimisation objective can is defined as:

$$\begin{aligned} L(\mathcal{X}, y, \tilde{\mathcal{X}}, \tilde{y}) &= L_c(y, \tilde{y}) + L_r(\mathcal{X}, \tilde{\mathcal{X}}) \\ &= \text{CrossEntropy}(y, \tilde{y}) + \text{ChamferDistance}(\mathcal{X}, \tilde{\mathcal{X}}), \end{aligned} \tag{4.4}$$

where Cross Entropy is the classification loss as defined in equation (2.46), Chamfer Distance is the distance function defined in equation (2.47), $\mathcal{X}, y$ refer to original point cloud and its label, while $\tilde{\mathcal{X}}, \tilde{y}$ refer to point cloud reconstruction and predicted class returned by the generator G and the classifier C respectively.

Algorithm 4.1: Random Compression Rehearsal training procedure for t tasks. By $\mathcal{A} \subset_k \mathcal{B}$ we denote a point cloud $\mathcal{A}$ that is a random sub-sampling consisting of k points form point cloud $\mathcal{B}$.

In : t – number of tasks, $\mathcal{D}_{\hat{t}} = \{(\mathcal{X}_{\hat{t},1}, y_{\hat{t},1}), \ldots, (\mathcal{X}_{\hat{t},n}, y_{\hat{t},n})\}$ – training samples with corresponding class labels for the task $\hat{t} \in \{1, 2, \ldots, t\}$

Out : $\theta_E^*, \theta_G^*, \theta_C^*$ – optimal model parameters for encoder, generator and classifier modules, $\mathcal{D}_c$ – dataset of compressed past samples

$\theta_E, \theta_G, \theta_C \leftarrow$ Initialisation `// parameters for encoder, generator and classifier`

$\mathcal{D}_c \leftarrow \varnothing$ `// create a dataset for compressed rehearsal data`

for $\hat{t} \leftarrow 1$ **to** t **do**

  while *convergence not reached* **do**

    $\mathcal{X}, \mathbf{y} \leftarrow (\mathcal{X}, \mathbf{y}) \subseteq (\mathcal{D}_t \cup \mathcal{D}_c)$ `// sample minibatch`

    $\tilde{\mathbf{Z}} \leftarrow E(\mathcal{X})$ `// encode point clouds in` $\mathcal{X}$

    $\tilde{\mathcal{X}} \leftarrow G(\tilde{\mathbf{Z}})$ `// reconstruct shapes from` $\mathcal{X}$ `based on` $\tilde{\mathbf{Z}}$

    $\tilde{\mathbf{y}} \leftarrow C(\tilde{\mathbf{Z}})$ `// classify shapes from` $\mathcal{X}$ `based on` $\tilde{\mathbf{Z}}$

    $L \leftarrow L_r(\mathcal{X}, \tilde{\mathcal{X}}) + L_c(\mathbf{y}, \tilde{\mathbf{y}})$ `// loss function as in equation (4.4)`

    $\theta_E, \theta_G, \theta_C \leftarrow optimiser(\nabla L, \theta_E, \theta_G, \theta_C)$ `// update weights`

  $\mathcal{D}_c \leftarrow \mathcal{D}_c \cup \{(\tilde{\mathcal{X}}_{\hat{t}} \subset_k \mathcal{X}_{\hat{t}}, y_{\hat{t}}) \mid (\mathcal{X}_{\hat{t}}, y_{\hat{t}}) \in \mathcal{D}_{\hat{t}}\}$ `// Compress current data and add it to rehearsal dataset`

$\theta_E^*, \theta_G^*, \theta_C^* \leftarrow \theta_E, \theta_G, \theta_C$

return $\theta_E^*, \theta_G^*, \theta_C^*, \mathcal{D}_c$

Further mitigation of catastrophic forgetting is obtained with the rehearsal procedure. This procedure consists of saving a portion of the original data that will be added to the new task inputs in order to preserve the model's performance quality on the previous tasks. The mentioned portion can consists of picking a number of

complete samples out of the original dataset, picking a small portion of each sample of the original dataset or a combination of those two. Random Compression Rehearsal implements the second approach and selects a small subset, consisting of 100 points of every point cloud, significantly reduces storage requirements to keep past tasks' data. These compressed samples are subsequently used as a supplementary dataset during model re-training. The procedure pseudocode is described in algorithm 4.1.

4.4 Experiments

This section presents the results of the empirical evaluation of the Random Compression Rehearsal approach. The experiments were conducted on the following point cloud datasets:

1. Point cloud MNIST – the 2-dimensional point cloud equivalent to the popular MNIST dataset (LeCun et al. 1998), created by sampling the points lying inside the digit's area. Due to the dataset containing only 10 classes, it was used only for evaluating RCR on the 5-task distance,

2. ModelNet 10 (Z. Wu et al. 2015) – similarly to MNIST, only 5-task experiments were performed, due to the dataset only having 10 classes,

3. ModelNet 40 (Z. Wu et al. 2015) – the dataset was used for 5- and 20-task distance,

4. ShapeNet (Chang et al. 2015) – although containing over 50 classes, the ShapeNet dataset was used for 5- and 20-task distance, for comparison reasons with ModelNet 40.

Detailed description of the datasets, are provided in the section 2.5.

All datasets were prepared as a collection of point clouds consisting of 2048 points per shape. In the case of the 3-d MNIST, the points were sampled from the area

of a white digit on black background with the probability based on a pixel intensity value. The points were sampled uniformly from the corresponding 3-dimensional meshes for the rest of the specified datasets.

The task datasets were prepared by first sorting class labels by their popularity in the dataset and then assigning the two largest classes to task number one, the third and the fourth largest to task number two, etc.

During optimisation the samples were augmented by random rotation ($\pm 15°$ in case of a MNIST dataset and $\pm 180°$ around vertical axis $\vec{z}$ otherwise), random noise $\varepsilon \sim \mathcal{N}(0, 0.02)$ capped at 0.05 and random flip (with chances of a flip vs $\vec{xy}$ plane – 25% , $\vec{yz}$ plane – 25% and no flip – 50 % chance). Optimisation was performed using Adam optimiser (Kingma and Ba 2015), with the learning rate $5 \cdot 10^{-4}$ and hyperparameters $\boldsymbol{\beta} = (0.9, 0.999)$. Exponential decay on the learning rate and the batch normalisation rate was also applied. For every 100,000 training iterations, the learning rate was multiplied by 0.7, with the lowest possible value of 10^{-5}. The batch normalisation rate, starting at 0.5, was also decayed every 100,000 iterations. The decay rate was set to 0.5 with the minimum b.n. rate value of 0.01.

The experiment results for RCR and related works are shown in the table 4.1. The models were compared based on the classification accuracy for 5 and 20 tasks (i.e. after 4 and 19 re-trainings). Moreover, two scenarios were tested – when the task information was known (Task-IL) and unknown (Class-IL) at evaluation time. Additionally, the following results were provided for comparison purposes:

- baseline – fine-tuning, i.e. re-training with no catastrophic forgetting mitigation,
- toplines – supervised learning on all classes from the start and incremental training using complete datasets from the past tasks.

As we can observe, by using Random Compression Rehearsal the catastrophic forgetting effect has been severely reduced compared to other continual learning approaches, including I3DOL (Dong et al. 2021), explicitly prepared for 3-dimensional

Table 4.1: Continual classification results after 5 and 20 tasks. For reference, two topline and one baseline experiments are provided. Topline include single supervised training on a complete data on all classes and incremental training.with carrying over complete past data. Baseline was obtained by fine-tuning the model only on the new data, without any catastrophic forgetting preventing measures. All experiments were scored over three independent runs and are presented as a mean with a standard deviation value. Class-IL values for LwF and EWC are omitted since these models assume knowledge about the task. The I3DOL results are taken from (Dong et al. 2021) in which the authors evaluate training with four (ModelNet 40) and six (ShapeNet) classes per task. Therefore, the error for ShapeNet on 40 classes is obtained by interpolating error values for 36 and 42 classes.

Dataset	Tasks	Supervised	Task type	Fine-tune	Incremental	LwF	EWC	iCaRL	I3DOL	RCR
MNIST	5	98.3 ± 0.1	Task-IL	71.2 ± 5.5	99.3 ± 0.1	84.5 ± 0.3	86.8 ± 0.1	93.4 ± 0.3	-	**99.7** ± 0.1
			Class-IL	19.5 ± 0.3	95.1 ± 0.4	-	-	60.5 ± 1.8	-	**97.0** ± 0.1
ModelNet10	5	92.1 ± 0.2	Task-IL	72.3 ± 1.1	96.6 ± 0.4	91.5 ± 0.4	86.3 ± 0.8	87.1 ± 0.2	-	**96.8** ± 0.2
			Class-IL	15.0 ± 0.1	90.4 ± 0.2	-	-	72.2 ± 1.5	-	**90.2** ± 0.1
ModelNet40	5	97.6 ± 0.2	Task-IL	52.4 ± 1.1	99.7 ± 0.1	91.7 ± 0.4	92.4 ± 0.8	90.2 ± 0.4	-	**99.8** ± 0.1
			Class-IL	20.0 ± 0.1	96.7 ± 0.2	-	-	82.2 ± 2.4	-	**94.2** ± 0.2
	20	88.2 ± 0.8	Task-IL	56.4 ± 2.7	99.3 ± 0.1	91.2 ± 0.4	87.1 ± 1.6	89.2 ± 0.3	-	**99.0** ± 0.1
			Class-IL	1.7 ± 0.1	88.6 ± 0.5	-	-	53.3 ± 0.9	61.5	**86.9** ± 0.2
ShapeNet	5	97.3 ± 0.1	Task-IL	85.3 ± 2.8	99.8 ± 0.1	96.4 ± 0.6	85.9 ± 0.9	93.5 ± 1.4	-	**99.5** ± 0.1
			Class-IL	8.4 ± 0.1	96.5 ± 0.1	-	-	77.3 ± 1.0	-	**95.1** ± 0.3
	20	90.8 ± 0.2	Task-IL	62.4 ± 4.4	99.8 ± 0.1	69.1 ± 0.3	69.6 ± 0.6	81.0 ± 0.7	-	**99.2** ± 0.1
			Class-IL	0.6 ± 0.1	91.5 ± 0.3	-	-	30.5 ± 4.2	74.4	**87.3** ± 0.1

shapes while being significantly more straightforward and easier to train.

The results presented in figures 4.2 and 4.3 show comparisons between Random Compression Rehearsal and other relevant approaches for task-by-task rolling accuracy for Task-IL (a task known at test time) and Class-IL (task unknown at test time) scenarios, respectively. All models have a length of the feature vector set to 2048. The RCR saves 100 points of each point cloud for rehearsal. We set the size of the memory buffer for iCaRL proportionally to the compression ratio of RCR (i.e., 5% of the total number of samples). The results for I3DOL are calculated based on

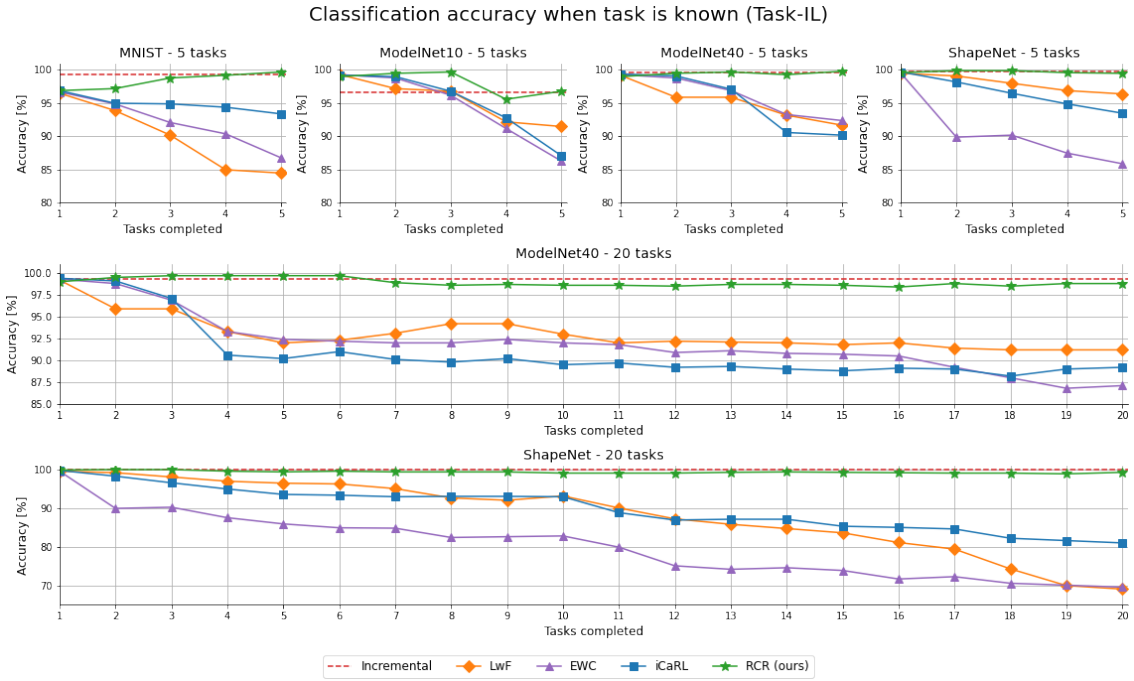

Figure 4.2: Average accuracy after t tasks completed, with task *known* during the inference. The confidence intervals are omitted for brevity.

the data reported in (Dong et al. 2021)[2].

As presented, the RCR approach manifests a much slower rate of performance degradation compared to methods from the literature.

4.5 Discussion

This section presents an application of representation learning to the continual training of a 3-dimensional point cloud classifier, with which I intend to answer the research question stated in section 1.2.6, concerning learning representations capable of retaining learned features of 3-dimensional shapes. The main objective of this approach is to solve the problem of re-training the model on a new dataset without

[2]I3DOL results are reported when training with 4 and 6 classes per task for ModelNet 40 and ShapeNet, respectively. Error for ShapeNet trained on 40 classes was obtained by interpolating error values for 36 and 42 classes.

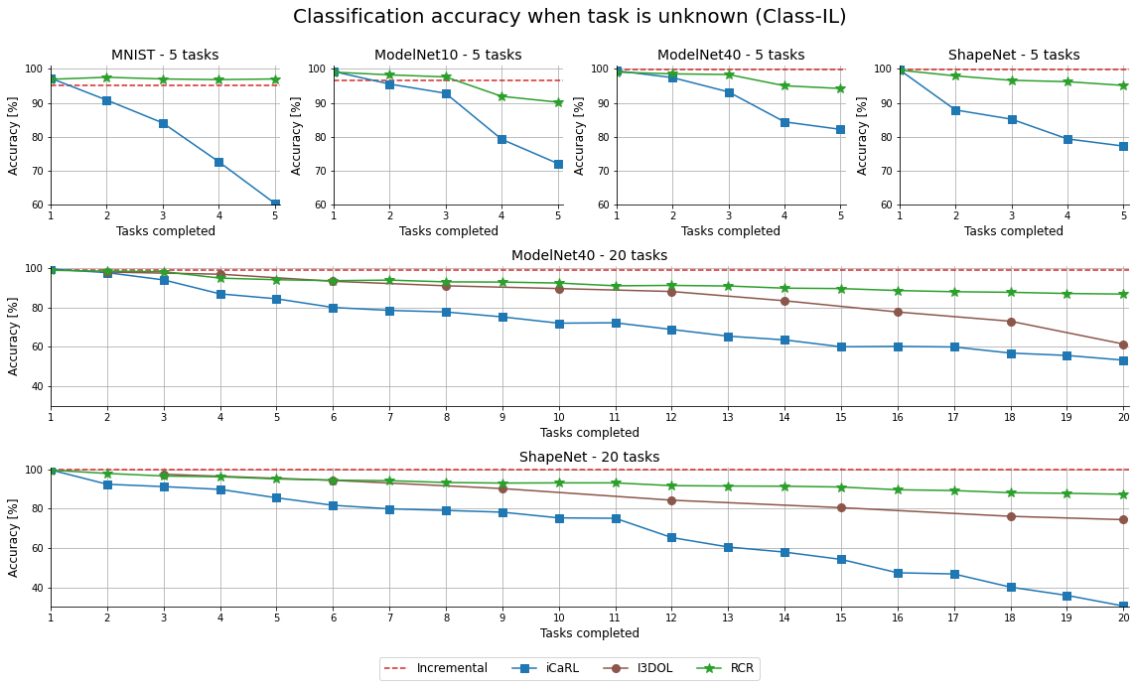

Figure 4.3: Average accuracy after t tasks completed, with task *unknown* during the inference. The confidence intervals are omitted for brevity.

losing the performance on the data it was trained on before, known as catastrophic forgetting. First, a precise description of the research objective was provided, along with a review of the current state-of-the-art solutions. Following, the proposed architecture of Random Compression Rehearsal was introduced, along with the training and evaluating methodology. In the end, the results in the table 4.1 and in figures 4.2 and 4.3 were presented, showing the superior classification ability of RCR after multiple re-trainings, compared to the literature.

5 Representation learning for point cloud completion

5.1 Goal of the studies

This chapter describes the research related to the application of proposed autoencoder-based architectures to learning holistic representations for the problem of point cloud completion, as stated in section 1.2.7.

The vast amount of visual data is often available to record for just a brief period of time. Even with a modern capturing device, taking more than one exact copy is often impossible due to many random, uncontrollable factors. Therefore, the reconstruction of corrupted data is an essential problem in many areas of information technology. In the context of 3-dimensional point clouds, data corruption, in addition to physical damage, may consist of objects of interest being occluded by other moving targets, synchronisation errors or uneven sampling of the object's surface. Thus, it is vital to create a solution that, given a corrupted sample, would be able to reconstruct the original shape.

This section considers the problem of reconstructing defective shapes, defined as representation learning of complete point cloud data. The task assumes the feature extraction from input data and subsequent generation of the missing part of the point cloud. However, depending on the severity of the data corruption, the input sample may not provide enough information about the rest of the shape. Therefore, in such cases, it may be impossible to reconstruct the data perfectly and predicting

the probable completion is also considered a success (Yuan et al. 2018).

5.2 Problem formulation

Let us consider a point cloud $\mathcal{P}$, represented as a set of 3-dimensional points $\mathbf{p} \in \mathbb{R}^3$. Furthermore, let us denote $\mathcal{P}_{\vec{a},1}$ and $X_{\vec{a},2}$ the new point clouds created by splitting $\mathcal{P}$ in two equinumerous, disjoint sets along the axis $\vec{a} \in \{\vec{x}, \vec{y}, \vec{z}\}$, such that:

$$\mathcal{X}_{\mathbf{a},1} \cup \mathcal{X}_{\mathbf{a},2} = \mathcal{X} \text{ and } \mathcal{X}_{\mathbf{a},1} \cap \mathcal{X}_{\mathbf{a},2} = \emptyset \tag{5.1}$$

Given one such part of the point cloud, the goal of the point cloud completion task is to predict the other part (from the quantitative standpoint) and to do it in semantically coherent manner (from the qualitative results perspective).

5.3 Methods

This section describes the proposed approaches to point cloud completion tasks.

This research (Zamorski et al. 2020c) compares three popular representation models: Autoencoders (AE), Variational Autoencoders (VAE) and Adversarial Autoencoders (AAE). A detailed description of those architectures is provided in section 2.3.4. In order to adapt those models to 3-dimensional data, the PointNet feature extractor (as described in the figure 2.7) was used as an encoder backbone. Moreover, Adversarial Autoencoders were evaluated with two prior distribution regularizations – using standard, isotropic Gaussian distribution $\mathcal{N}(0, \mathbf{I})$ and Beta distribution $Beta(\alpha = 0.01, \beta = 0.01)$. Such Beta distribution has the probability mass concentrated heavily towards the ends of the distribution support range $[0, 1]$. This property allows for easy binarisation of features by setting the cutoff threshold value at 0.5.

The two AAE variants are denoted AAE-Normal and AAE-Beta, for regularisation with Normal and Beta distributions, respectively.

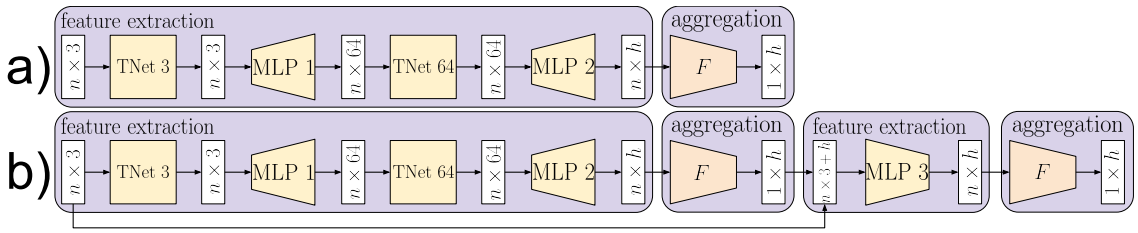

Figure 5.1: Architecture of (a) PointNet and (b) Double PointNet encoders used as feature extracting backbones in the proposed approaches. Double PointNet architecture consists of four parts: 1) feature extraction encoding each point separately to a vector of h features, 2) aggregation function, combining vectors feature-wise across all points to obtain intermediate shape representation. 3) an additional network calculating global features based on a concatenated vectors of point location and intermediate shape representation, 4) aggregation function as in point 2. `TNet k` denotes k-dimensional affine transformation (Qi et al. 2017a).

In the original approach, features extracted from each point are calculated without considering the whole shape. To consider the distribution of all points during the feature extraction procedure, an extension to the PointNet architecture, called *Double PointNet* is proposed. The approach is presented in the figure 5.1. Inspired by PointNet's approach to segmentation, the obtained representation of the shape is then appended to every point 3-dimensional vector of position, resulting in a matrix of size $n \times (3 + h)$. Afterwards, said matrix is processed by an additional neural network (`MLP 3` in the figure 5.1). The returned features are aggregated once more to produce the final global representation of the point cloud shape.

Representing shape as a matrix of size $n \times (3 + h)$ ties the point location to the features of an overall shape, providing information about each point based on the cloud as a whole. The justification for this process comes from the fact that selecting a global feature vector just by max-pooling individually obtained local feature vectors may result in representations concerned with most discriminative features while dropping information about an entire shape. By double processing, the encoder is able to produce representations that are obtained based on raw point

location and point location relative to the distribution of points.

5.4 Experiments

This section describes the results obtained during the empirical evaluation of the proposed reconstruction architectures for learning representations in the context of the point cloud completion task.

Table 5.1: Comparison of a reconstruction fidelity using different models, measured as the EMD distance between original and completed point cloud. Prefix D symbolizes architectures using Double PointNet encoder. Suffixes N and B stand for Normal and Beta, respectively.

Class	AE	VAE	AAE-N	AAE-B	D-AE	D-VAE	D-AAE-N	D-AAE-B
Chair	64.99	216.22	68.79	72.58	**56.29**	122.59	64.76	70.69
Airplane	39.65	143.24	48.19	51.08	**31.08**	104.94	36.05	43.47

All experiments were conducted on the shapes belonging to *chair* and *airplane* classes of the ShapeNet dataset. The dataset details are provided in section 2.5. The point clouds were prepared by sampling 2048 points from the surface of each mesh and were augmented with random rotations around the $\vec{z}$-axis. After augmentation, the split was performed by randomly choosing an axis ($\vec{x}$, $\vec{y}$, $\vec{z}$) and taking points with the upper or lower half of values along that dimension.

For each autoencoder model (AE, VAE, AAE-Normal, AAE-Beta), two architectures are considered – with the original and with Double PointNet, for a total of eight evaluated approaches. Prefix *Double* will be added to the name of every model using Double PointNet as an encoder to distinguish between architectures. All the experiments were conducted with a fixed representation size $h = 100$.

The table 5.1 presents the reconstruction fidelity of input samples joined with their completions with regards to the ground truth data, measured with the Earth

Mover's Distance, as defined in the equation (2.48). By comparing VAE and AAE-Normal models, two generative approaches regularised with Gaussian distribution prior, it can be observed that VAE is unable to learn any meaningful completions. It leads to the conclusion that Kullback-Leibler divergence-based regularisation may be too strong of a constraint for the encoder considering the completion task. On the other hand, AAE-Normal and D-AAE-Normal do not seem to suffer from imposing a latent space regularisation, having only a slight decrease in completion quality compared to the baseline.

Comparing the results of adversarial autoencoders regularised with normal and beta distributions shows that the application of binary encoding results in a minor drop in the model's quality. It suggests that meaningful binary representations for shape completions that match the real-valued features are possible to obtain.

It can be observed that approaches based on the Double PointNet significantly outperformed methods with the standard encoder. Out of those "double" architectures, the best reconstruction fidelity was achieved using the basic Autoencoder model. However, while the autoencoder approach offers the best reconstruction quantitatively, it does not offer a possibility to regularise the latent distribution and create new completions in a generative manner.

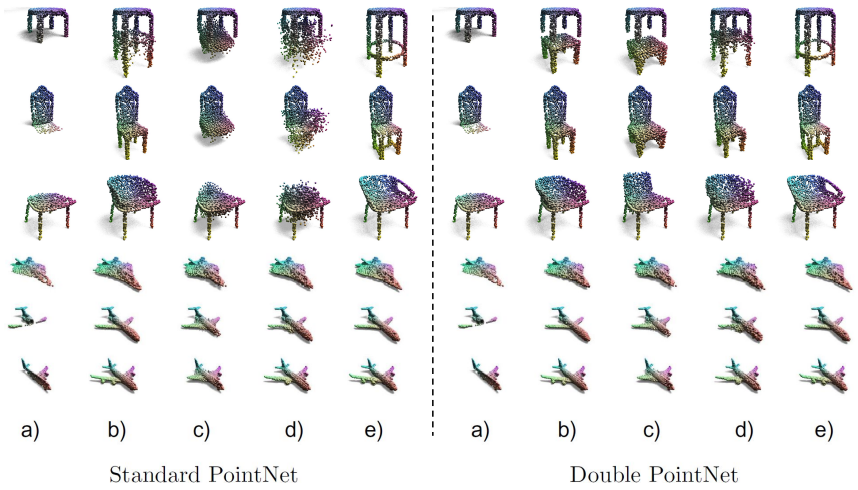

Figure 5.2: Visualisations of the point clouds completions for 3 examples from the test set. Columns represent: a) input, b) AE, c) VAE, d) AAE-Normal, e) ground truth.

Figure 5.2 presents the qualitative results achieved by compared approaches on selected samples from the *chair* and the *airplane* classes. By comparing results for models using standard and double PointNet, the difference in favour of "double" can be observed. The presented visual results show that the Adversarial Autoencoder models learned the point cloud completion task instead of underfitting or landing early in the bad local minima, as with Variational Autoencoders. Moreover, the quality drop is minuscule despite the quantitative differences between AE and AAE-based models.

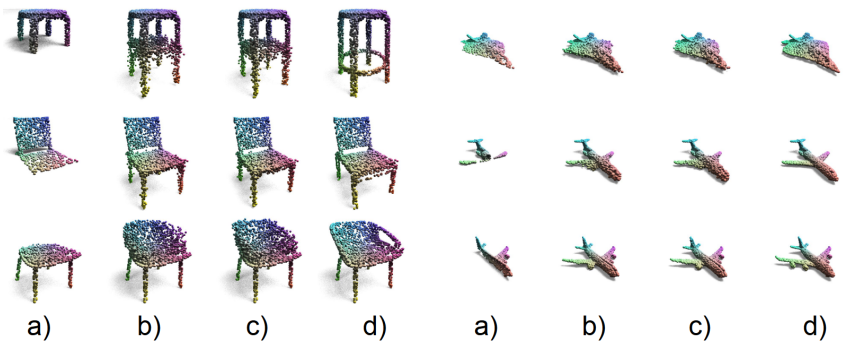

Figure 5.3: Visualisations of the point clouds completions for 3 examples from the test set. Columns represent: a) input, b) D-AAE-Normal, c) D-AAE-Beta, d) ground truth.

Figure 5.3 shows examples comparing the results of creating point cloud completion by D-AAE-Normal and D-AAE-Beta models. It can be noted that the AAE model with the double PointNet encoder can encode both point-wise and structure-wise information to the latent representation and obtain good completions from binary vectors. Qualitative results obtained from the D-AAE-Beta model confirm that a minuscule drop in EMD fidelity in comparison to D-AAE-Beta (see table 5.1) translates with no noticeable negative consequences to visual quality produced from binary representations.

5.5 Discussion

This chapter presents an application of representation learning and generative modelling to the problem of shape completion of 3-dimensional point clouds as my proposed answer to the research question outlined in section 1.2.7 concerning extracting holistic features describing the entire shape based on incomplete input. First, PointNet, along with the proposed approach dubbed *DoublePointNet* to missing part generation, were described along with the employed prior distributions (Normal and Beta) to latent space regularisations. Next, the evaluation procedure was outlined, and its quantitative results were provided in the table 5.1. Lastly, the qualitative results for all the models were shown in the figure 5.2 with architectures producing real-valued and binary features being compared in figure 5.3.

6 Representation learning for classification

6.1 Goal of the studies

This chapter describes the results of research concerning the problem of feature selection strategy for PointNet architecture, as described by the research question in section 1.2.8.

Obtaining the meaningful representations of data is the cornerstone of all machine learning methods. Doing so efficiently is especially important for 3-dimensional point clouds, usually structured as sets of points, which causes the factorial number of points of the possible orderings, ultimately representing the same shape. Although the proposed models for dealing with volumetric data have shown steady progress in classification and semantic segmentation accuracy, the point cloud domain architectures were mainly unsuccessful until the introduction of PointNet. By treating each point individually with the shared-weight network and then merging the point-wise features with the order-invariant function, it was able to achieve the identical representation regardless of the permutation of the input (detailed description of the model is provided in the section 2.4.3).

However, the PointNet architecture assumes a specific feature-aggregation function – max pooling. While giving the state-of-the-art classification and segmentation scores, selecting a different aggregation method may lead to further progress on those key machine learning tasks. This chapter considers several different approaches to

selecting shape-wise features from point-wise ones. Moreover, while the `max` function takes only the extreme value of the feature distribution, the proposed alternatives consider the entire range of features or employ multi-statistic calculation for obtaining the global representation.

6.2 Problem formulation

Let us consider a PointNet (Qi et al. 2017a) classification architecture to predict shape classes of the 3-dimensional point cloud. Part of the model architecture (see figure 2.7) is the permutation invariant function F that can be described as

$$F : \mathbb{R}^{n \times h} \to \mathbb{R}^{h}, \tag{6.1}$$

where n – number of points in the point cloud and h – size of the latent representation. In (Qi et al. 2017a) the function F is implemented as a max-pooling operation that calculates the global, shape-wise feature vector from the local, point-wise features. The goal is to construct different permutation-invariant function $\Phi : \mathbb{R}^{n \times h} \to \mathbb{R}^{h}$ that improves the classification accuracy and balanced accuracy scores on point cloud data.

6.3 Methods

The original PointNet approach of aggregation with `max` function comes with limitations. Constructing feature using `max` ignores the shape of the distribution of the activation and reduces it to just the most extreme value, which may be an outlier. Creating a global feature vector using other aggregation functions that consider local feature distribution in its entirety may benefit the quality of produced representations.

This research focuses on comparing three classes of approaches to permutation-invariant feature selection:

1. **Single-statistic** – The `sum`, `median`, `mean` and `max` functions, obtaining vectors of size h each,

2. **Multi-statistic** – Combinations of single-statistics approaches `Max-Mean-Sum` (`MMS`) and `Max-Mean-Median` (`MMM`), resulting in vectors of size $3h$. Those results were subsequently passed through a two-layer neural network (with layers of length $2h$ and h) to reduce the global vector size from $3h$ to h for a fair comparison with single-statistic functions.

3. **Multi-valued single-statistic** – The `top-`k function, a modification of described `max` function that returns k biggest values from the input set, instead of just the maximal one, resulting in the feature vector of size $k \cdot h$. Those values are later passed through a one-layer neural network to reduce the vector size from $k \cdot h$ to h in order to compare this method with other approaches.

6.4 Experiments

This section describes the results obtained during the empirical evaluation of the aggregation functions. All experiments were conducted on the ShapeNet (Chang et al. 2015) dataset, prepared as a collection of point clouds made of 2048 points per shape, augmented with random rotations around the $\vec{z}$-axis. The proposed methods of obtaining shape embeddings are evaluated using the downstream classification task and are compared using prediction accuracy (equation (2.33)) and balanced accuracy (equation (2.34)). Additionally, two functions that return a feature of a specific point from the distribution are compared based on a number of unique points used to create the global representation. For representing this diversification, the diversity metric was used, defined as:

$$\text{diversity}(F) = \frac{1}{n} \sum_{i=1}^{n} \frac{\left| \text{set}\left(\hat{F}(\mathcal{X}_i) \right) \right|}{h}, \tag{6.2}$$

where $\hat{F}$ is an arg-version of an aggregation function F (i.e. `argmax` instead of `max` and `argmedian` instead of `median`), set($\mathbf{a}$) – unique elements of a vector $\mathbf{a}$, $|\mathbf{a}|$ – number of elements of a vector $\mathbf{a}$, h – size of a feature vector, n – number of samples.

Five sizes h of the feature vector $\mathbf{z}$ were used for experiments, i.e. $h \in \{8, 16, 25, 50, 100\}$.

Table 6.1: Comparison of a classification accuracy given different aggregation functions and lengths h of the global feature vector.

aggregation	$h = 8$	$h = 16$	$h = 25$	$h = 50$	$h = 100$
max	0.8426	0.8468	0.8494	0.8471	0.8484
mean	0.8563	0.8610	0.8632	0.8648	0.8703
median	0.8536	0.8559	0.8623	0.8636	0.8677
MMS	0.8561	0.8580	0.8642	0.8677	0.8717
MMM	**0.8735**	**0.8671**	**0.8701**	**0.8739**	**0.8765**

The table 6.1 presents the global model accuracy (i.e. number of correctly predicted samples divided by the total number of samples). It can be noted that combining different types of features (e.g. `MMM`) offers better results than using just a single statistic or three with heavy correlation (`MMS`). However, when restricting the results to just single statistic function, then `mean` and `median` aggregations offer superior performance than the baseline statistic `max`.

The different conclusions can be drawn by taking the class imbalance of the ShapeNet dataset into account. By calculating the local model accuracy (i.e. the average of accuracy scores calculated among each class individually), the results (presented in the table 6.2) suggest that when using a very narrow feature vector (i.e. $h \leqslant 25$) it is enough to use the single statistic that takes every feature into account (such as `mean` or `median`) instead of the one that focuses only the most extreme case (as is with `max`). Moreover, combining multiple statistics offers no noticeable improvement. Combined feature vectors offer a minimal accuracy gain even for cases on more extensive feature vectors (i.e. $h \geqslant 50$).

Table 6.2: Comparison of a classification balanced accuracy given different aggregation functions and lengths h of the global feature vector.

aggregation	$h = 8$	$h = 16$	$h = 25$	$h = 50$	$h = 100$
max	0.6704	0.6785	0.6805	0.6826	0.6879
mean	**0.6966**	**0.7029**	**0.7091**	0.6938	0.7075
median	0.6861	0.6824	0.6954	0.6956	0.7094
MMS	0.6682	0.6767	0.7082	0.7056	0.7021
MMM	0.6854	0.6880	0.7035	**0.7072**	**0.7182**

Table 6.3: Comparison of accuracy and balanced accuracy between top-k of max values. Note that using the top-1 aggregation is the same as using the standard `max` function.

top-k	accuracy	b. accuracy
$k = 1$	0.8468	0.6785
$k = 2$	0.8547	0.6802
$k = 3$	**0.8560**	0.6796
$k = 4$	0.8554	**0.6987**
$k = 5$	0.8555	0.6869

The table 6.3 compares the results of classification accuracy, when the usage of k instead of just one maximum feature value is considered for $h = 100$. The values of k are in the range $k \in \{1, 2, 3, 4, 5\}$, as there was no noticeable improvement above those values. Based on the outcomes, it can be concluded that adding the second or the third-highest feature value may offer minor improvement in classification balanced and unbalanced accuracy.

The results in the table 6.4 offer a view of the number of repetitions of samples in which feature values were selected into global feature vector $\tilde{\mathbf{z}}$. Depending on the characteristic of a downstream task, it may be beneficial to construct a latent representation that consists of a more diverse subset of points from the point cloud. Much

Table 6.4: Comparison of the diversity of the points that compose the feature vector. A higher value means that a bigger portion of the feature vector was created by taking values of the unique points from the point cloud.

aggregation	$h = 8$	$h = 16$	$h = 25$	$h = 50$	$h = 100$
max	0.9333	0.8829	0.8659	0.7736	0.6409
median	**0.9821**	**0.9530**	**0.9197**	**0.8063**	**0.7151**

higher values obtained using `median` aggregation suggest that there usually exists a much smaller subset of points that are considered important from the statistic-picking point of view than when using the `max` function. This result suggests that `median` aggregation may be better suited to any problems related to the generation or reconstruction of point clouds, where it is usually beneficial to represent as much detail of the shape as possible.

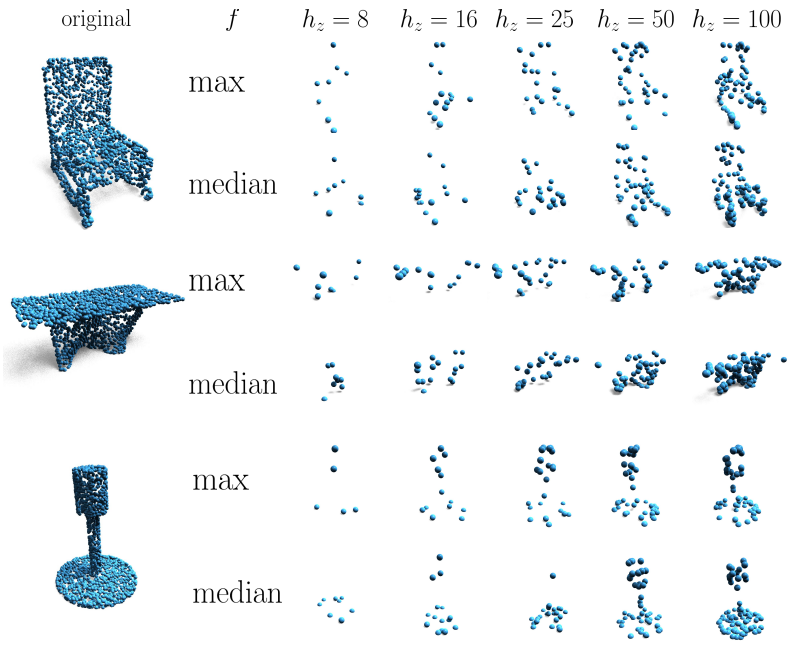

Figure 6.1: Visualisation of points that were used for global feature vector composition for different types of aggregation function F and feature vector length h. Original point clouds are presented in the left column.

The diversity scores can be compared qualitatively by plotting the *critical points* (see figure 6.1) of shapes, i.e. points whose feature values were picked by the aggregation functions for the construction of the global feature vector. The critical points form either the outline of the shape (when using `max`) or a more uniform, complete shape (`median` aggregation).

6.5 Discussion

This section presents an application of representation learning to train a neural network classifier to solve the task of predicting the 3-dimensional point cloud data shapes, with which I aim to answer the research question stated in section 1.2.8 concerning the assembly of partial representation into holistic one. This research focuses on determining the optimal permutation-invariant function, which serves as a selector from the collection of local, point-wise features into a global, shape-wise feature vector. First, the justification for choosing the selected function for evaluation was provided along with the analysis methodology. Following, the results for the classification experiments are provided in tables 6.1 to 6.3. Next, the model selection of critical points was assessed in the context of the shape type, feature vector length and aggregation function used (figure 6.1). In the end, the results in table 6.4 describe the diversity of selected points used to obtain a latent representation of the input data.

7 Conclusions

7.1 Original contribution

The main achievements of the work that contributes to the domain of machine learning are as follows:

- **The introduction of three generative modelling approaches for feature extraction and 3-dimensional point cloud data generation.** This thesis describes three novel architectures for representation learning on point clouds[1]:

    1. *3dAAE* – adversarial-autoencoder based model, capable of regularising the obtained embeddings to the arbitrary probability distribution. This flexibility was presented by obtaining features following the Normal, Mixture of Gaussians, Beta and Categorical distributions.

    2. *HyperCloud* – hypernetwork-based approach, capable of generating an arbitrary number of points per cloud, as well as creating mesh-based shapes.

    3. *CIF Network* – normalising flow-based architecture, with a conditional generation of arbitrarily sized point clouds.

[1] I am the primary author of the first approach, i.e. 3dAAE and worked as one of the secondary authors on research concerning HyperCloud and CIF models. My contributions as a secondary author consist of designing and running experiments, analysing the results, as well as reviewing the manuscript.

For each of those solutions, an extensive assessment was performed. These empirical evaluations confirmed the high quality of the proposed methods compared to the related works from the literature.

- **The development of an autoencoder-based architecture for a problem of completing missing parts of the point cloud shapes.** This work tackles the challenging problem of generating the missing parts that seamlessly fit the input shape. For this purpose, several generative models were introduced and presented with both qualitative and quantitative results.

- **The introduction of a novel procedure to perform continual learning for point cloud data.** The proposed procedure tackles the problem of re-training the model on a set of new data without losing the prediction quality on the past samples. By regularising the training with the reconstruction module of an autoencoder-like model and high compression of the past training samples, the high degree of *catastrophic forgetting* mitigation was achieved. The evaluation of this approach based on the preservation of the classification accuracy as well as in terms of possible re-trainings approach rendered better results than relevant existing work.

- **The analysis of the effect of the feature aggregation method on obtaining the representation of 3-dimensional point cloud shapes with subsequent improvement on the classification accuracy task.** This work compares several permutation-invariant approaches for calculating the shape-wise latent representation on point clouds. Considering the procedures that produce features based on the entire point-wise feature distribution yielded an improvement in representation quality measured by the accuracy of the classification task.

The presented research topics that constitute this thesis are described based on the following research publications: (Zamorski and Zięba 2019; Spurek et al. 2020;

Zamorski et al. 2020a; Zamorski et al. 2020b; Zamorski et al. 2020c; Stypułkowski et al. 2021; Zamorski et al. 2022).

7.2 Proposed directions of the future work

The analysis of the subject of deep representation learning and the obtained results lead to the following indication about the directions of future work:

1. The introduction of the generative approach of thermodynamics-based diffusion probabilistic approaches (Sohl-Dickstein et al. 2015; Y. Song and Ermon 2019; Ho et al. 2020) and score-based models (Vahdat et al. 2021) to the problem of learning representation and generation of point clouds,

2. The proposal of generative models able to work with multi-modal (i.e. different types of shapes) point cloud and mesh-based data,

3. Proposing of continual learning strategies that are capable of re-training without explicitly specifying the task information (so-called *task-agnostic continual learning* (Kirichenko et al. 2021)) to the domain of 3-dimensional point cloud data,

4. The development of methods to learn the optimal permutation-invariant selection of features during the training procedure.

Bibliography

Achlioptas, Panos, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J Guibas (2018). "Learning Representations and Generative Models for 3D Point Clouds." In: *International Conference on Machine Learning (ICML)*, pp. 40–49.

Ackley, David H, Geoffrey E Hinton, and Terrence J Sejnowski (1985). "A learning algorithm for Boltzmann machines." In: *Cognitive science* 9.1, pp. 147–169.

Alldieck, Thiemo, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll (2018a). "Detailed human avatars from monocular video." In: *International Conference on 3D Vision (3DV)*. IEEE, pp. 98–109.

– (2018b). "Video based reconstruction of 3d people models." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8387–8397.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein generative adversarial networks." In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 214–223.

Arsalan Soltani, Amir, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum (2017). "Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1511–1519.

Aydin, Ali-Kemal, William D Haselden, Yannick Goulam Houssen, Christophe Pouzat, Ravi L Rungta, Charlie Demené, Mickael Tanter, Patrick J Drew, Serge Charpak, and Davide Boido (2020). "Transfer functions linking neural calcium to single voxel functional ultrasound signal." In: *Nature communications* 11.1, pp. 1–10.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013). "Representation learning: A review and new perspectives." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 35.8, pp. 1798–1828.

Bengio, Yoshua, Pascal Lamblin, Dan Popovici, and Hugo Larochelle (2006). "Greedy layer-wise training of deep networks." In: *Advances in Neural Information Processing Systems (NeurIPS)* 19.

Bertsekas, Dimitri P (1985). "A distributed asynchronous relaxation algorithm for the assignment problem." In: *1985 24th IEEE Conference on Decision and Control*. IEEE, pp. 1703–1704.

Bishop, Christopher M (2006). *Pattern Recognition and Machine Learning*. Springer.

Bogo, Federica, Javier Romero, Matthew Loper, and Michael J Black (2014). "FAUST: Dataset and evaluation for 3D mesh registration." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3794–3801.

Bogo, Federica, Javier Romero, Gerard Pons-Moll, and Michael J. Black (July 2017). "Dynamic FAUST: Registering Human Bodies in Motion." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Brock, Andrew, Theodore Lim, James M Ritchie, and Nick Weston (2016). "Generative and discriminative voxel modeling with convolutional neural networks." In: *Computing Research Repository arXiv:1608.04236*.

Cao, Pei, Hao Chen, Ye Zhang, and Gang Wang (2019). "Multi-view frustum pointnet for object detection in autonomous driving." In: *IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 3896–3899.

Chang, Angel X, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. (2015). "Shapenet: An information-rich 3d model repository." In: *Computing Research Repository arXiv:1512.03012*.

Chen, Ricky T. Q., Yulia Rubanova, Jesse Bettencourt, and David Duvenaud (2018). "Neural Ordinary Differential Equations." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey E Hinton (2020). "A simple framework for contrastive learning of visual representations." In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 1597–1607.

Chen, Weihua, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang (2017). "Beyond triplet loss: a deep quadruplet network for person re-identification." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 403–412.

Chen, Yucheng, Yingli Tian, and Mingyi He (2020). "Monocular human pose estimation: A survey of deep learning-based methods." In: *Computer Vision and Image Understanding (CVIU)* 192, p. 102897.

Chen, Zhiqin and Hao Zhang (2019). "Learning implicit fields for generative shape modeling." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5939–5948.

Cheng, Zezhou, Qingxiong Yang, and Bin Sheng (2015). "Deep colorization." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 415–423.

Chopra, Sumit, Raia Hadsell, and Yann LeCun (2005). "Learning a similarity metric discriminatively, with application to face verification." In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE, pp. 539–546.

Chowdhury, Townim, Mahira Jalisha, Ali Cheraghian, and Shafin Rahman (2021). "Learning Without Forgetting for 3D Point Cloud Objects." In: *International Work-Conference on Artificial Neural Networks*. Springer, pp. 484–497.

Choy, Christopher B, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese (2016). "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction." In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 628–644.

Çiçek, Özgün, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger (2016). "3D U-Net: learning dense volumetric segmentation from

sparse annotation." In: *nternational conference on medical image computing and computer-assisted intervention. (MICCAI)*. Springer, pp. 424–432.

Creswell, Antonia, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath (2018). "Generative adversarial networks: An overview." In: *IEEE Signal Processing Magazine* 35.1, pp. 53–65.

Dai, Angela, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner (2017a). "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dai, Angela, Charles Ruizhongtai Qi, and Matthias Nießner (2017b). "Shape completion using 3d-encoder-predictor cnns and shape synthesis." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5868–5877.

Deprelle, Theo, Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry (2019). "Learning elementary structures for 3d shape generation and matching." In: *Advances in Neural Information Processing Systems (NeurIPS)* 32.

Dinh, Laurent, David Krueger, and Yoshua Bengio (2015). "Nice: Non-linear independent components estimation." In: *International Conference on Learning Representations (ICLR) Workshop*.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density estimation using real nvp." In: *International Conference on Learning Representations (ICLR)*.

Dong, Jiahua, Yang Cong, Gan Sun, Bingtao Ma, and Lichen Wang (2021). "I3DOL: Incremental 3D Object Learning without Catastrophic Forgetting." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 35. 7, pp. 6066–6074.

Dosovitskiy, Alexey, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox (2014). "Discriminative unsupervised feature learning with convolutional neural networks." In: *Advances in Neural Information Processing Systems (NeurIPS)* 27.

Duch, Włodzisław, Maciej Nałęcz, and Andrzej Lang Akademicka Oficyna Wydawnicza Exit (2000). *Sieci neuronowe*. Exit.

Endres, Dominik Maria and Johannes E Schindelin (2003). "A new metric for probability distributions." In: *IEEE Transactions on Information theory* 49.7, pp. 1858–1860.

Feng, Yinglong, Shuncheng Wu, Okan Köpüklü, Xueyang Kang, and Federico Tombari (2019). "Unsupervised Monocular Depth Prediction for Indoor Continuous Video Streams." In: *Computing Research Repository arXiv:1911.08995*.

Feng, Yutong, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao (2019). "Meshnet: Mesh neural network for 3d shape representation." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 33. 01, pp. 8279–8286.

Fitzpatrick, Fiona, Herje Schagerlöf, Thomas Andersson, Sara Richardson, Folke Tjerneld, Karl-Gustav Wahlund, and Bengt Wittgren (2006). "NMR, cloud-point measurements and enzymatic depolymerization: complementary tools to investigate substituent patterns in modified celluloses." In: *Biomacromolecules* 7.10, pp. 2909–2917.

Forgy, Edward W (1965). "Cluster analysis of multivariate data: efficiency versus interpretability of classifications." In: *Biometrics* 21, pp. 768–769.

Fouhey, David F, Abhinav Gupta, and Martial Hebert (2013). "Data-driven 3D primitives for single image understanding." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3392–3399.

Frosst, Nicholas, Nicolas Papernot, and Geoffrey E Hinton (2019). "Analyzing and improving representations with the soft nearest neighbor loss." In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 2012–2020.

Fu, Huan, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao (2018). "Deep ordinal regression network for monocular depth estimation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2002–2011.

Georgakis, Georgios, Srikrishna Karanam, Ziyan Wu, and Jana Kosecka (2019). "Learning local rgb-to-cad correspondences for object pose estimation." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 8967–8976.

Ghahramani, Zoubin, Geoffrey E Hinton, et al. (1996). *The EM algorithm for mixtures of factor analyzers*. Tech. rep. Technical Report CRG-TR-96-1, University of Toronto.

Girshick, Ross (2015). "Fast r-cnn." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1440–1448.

Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587.

Gkioxari, Georgia, Jitendra Malik, and Justin Johnson (2019). "Mesh r-cnn." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 9785–9795.

Godard, Clément, Oisin Mac Aodha, and Gabriel J Brostow (2017). "Unsupervised monocular depth estimation with left-right consistency." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 270–279.

Godard, Clément, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow (2019). "Digging into self-supervised monocular depth estimation." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 3828–3838.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adver-

sarial nets." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2672–2680.

Grathwohl, Will, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud (2019). "FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models." In: *International Conference on Learning Representations (ICLR)*.

Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko (2020). "Bootstrap your own latent-a new approach to self-supervised learning." In: *Advances in Neural Information Processing Systems (NeurIPS)* 33, pp. 21271–21284.

Groueix, Thibault, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry (2018). "A papier-mâché approach to learning 3d surface generation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 216–224.

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville (2017). "Improved training of wasserstein gans." In: *Advances in Neural Information Processing Systems (NeurIPS)* 30.

Guo, Yulan, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun (2020). "Deep learning for 3d point clouds: A survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 43.12, pp. 4338–4364.

Gutmann, Michael and Aapo Hyvärinen (2010). "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models." In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR, pp. 297–304.

Ha, David, Andrew Dai, and Quoc V Le (2017). "Hypernetworks." In: *International Conference on Learning Representations (ICLR)*.

Hamming, Richard W (1950). "Error detecting and error correcting codes." In: *The Bell system technical journal* 29.2, pp. 147–160.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (2017). "Mask r-cnn." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2961–2969.

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2016). "beta-vae: Learning basic visual concepts with a constrained variational framework." In.

Hinton, Geoffrey E (2012). "A practical guide to training restricted Boltzmann machines." In: *Neural networks: Tricks of the trade*. Springer, pp. 599–619.

Hinton, Geoffrey E and Sam Roweis (2002). "Stochastic neighbor embedding." In: *Advances in Neural Information Processing Systems (NeurIPS)* 15.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks." In: *Science* 313.5786, pp. 504–507.

Hinton, Geoffrey E, Terrence J Sejnowski, et al. (1986). "Learning and relearning in Boltzmann machines." In: *Parallel distributed processing: Explorations in the microstructure of cognition* 1.282-317, p. 2.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising diffusion probabilistic models." In: *Advances in Neural Information Processing Systems (NeurIPS)* 33, pp. 6840–6851.

Hoang, Long, Suk-Hwan Lee, Oh-Heum Kwon, and Ki-Ryong Kwon (2019). "A deep learning method for 3D object classification using the wave kernel signature and a center point of the 3D-triangle mesh." In: *Electronics* 8.10, p. 1196.

Hoiem, Derek, Alexei A Efros, and Martial Hebert (2005). "Geometric context from a single image." In: *IEEE International Conference on Computer Vision (ICCV)*. Vol. 1. IEEE, pp. 654–661.

Huang, Qiangui, Weiyue Wang, and Ulrich Neumann (2018). "Recurrent slice networks for 3d segmentation of point clouds." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2626–2635.

Huang, Shih-Cheng, Anuj Pareek, Saeed Seyyedi, Imon Banerjee, and Matthew P Lungren (2020). "Fusion of medical imaging and electronic health records using deep learning: a systematic review and implementation guidelines." In: *NPJ digital medicine* 3.1, pp. 1–9.

Hutton, Chloe, Enrico De Vita, John Ashburner, Ralf Deichmann, and Robert Turner (2008). "Voxel-based cortical thickness measurements in MRI." In: *Neuroimage* 40.4, pp. 1701–1710.

Ilse, Maximilian, Jakub M Tomczak, Christos Louizos, and Max Welling (2020). "Diva: Domain invariant variational autoencoders." In: *Medical Imaging with Deep Learning*. PMLR, pp. 322–348.

Jang, Eric, Shixiang Gu, and Ben Poole (2017). "Categorical reparameterization with gumbel-softmax." In: *International Conference on Learning Representations (ICLR)*.

Jiang, Huaizu and Erik Learned-Miller (2017). "Face detection with the faster R-CNN." In: *IEEE International Conference on Automatic Face & Gesture Recognition*. IEEE, pp. 650–657.

Jiang, Jianwen, Di Bao, Ziqiang Chen, Xibin Zhao, and Yue Gao (2019). "MLVCNN: Multi-loop-view convolutional neural network for 3D shape retrieval." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 33. 01, pp. 8513–8520.

Jing, Longlong and Yingli Tian (2020). "Self-supervised visual feature learning with deep neural networks: A survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 43.11, pp. 4037–4058.

Jonschkowski, Rico, Austin Stone, Jonathan T Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova (2020). "What matters in unsupervised optical flow." In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 557–572.

Kanezaki, Asako, Yasuyuki Matsushita, and Yoshifumi Nishida (2018). "Rotationnet: Joint object categorization and pose estimation using multiviews from unsu-

pervised viewpoints." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5010–5019.

Kanzler, Mathias, Marc Rautenhaus, and Rüdiger Westermann (2018). "A voxel-based rendering pipeline for large 3D line sets." In: *IEEE Transactions on Visualization and Computer Graphics* 25.7, pp. 2378–2391.

Kaufman, Leonard and Peter J Rousseeuw (1990). "Partitioning around medoids (program pam)." In: *Finding groups in data: an introduction to cluster analysis* 344, pp. 68–125.

Khan, Faisal, Saqib Salahuddin, and Hossein Javidnia (2020). "Deep learning-based monocular depth estimation methods—a state-of-the-art review." In: *Sensors* 20.8, p. 2272.

Kim, Mingyu, Jihye Yun, Yongwon Cho, Keewon Shin, Ryoungwoo Jang, Hyun-jin Bae, and Namkug Kim (2019). "Deep learning in medical imaging." In: *Neurospine* 16.4, p. 657.

Kim, Pileun, Jingdao Chen, and Yong K Cho (2018). "SLAM-driven robotic mapping and registration of 3D point clouds." In: *Automation in Construction* 89, pp. 38–48.

Kingma, Diederik P and Jimmy Ba (2015). "Adam: A method for stochastic optimization." In: *International Conference on Learning Representations (ICLR)*.

Kingma, Diederik P and Prafulla Dhariwal (2018). "Glow: Generative flow with invertible 1x1 convolutions." In: *Advances in Neural Information Processing Systems (NeurIPS)* 31.

Kingma, Diederik P, Tim Salimans, Rafał Józefowicz, Xi Chen, Ilya Sutskever, and Max Welling (2016). "Improved variational inference with inverse autoregressive flow." In: *Advances in Neural Information Processing Systems (NeurIPS)* 29.

Kingma, Diederik P and Max Welling (2014). "Auto-encoding variational bayes." In: *International Conference on Learning Representations (ICLR)*.

Kirichenko, Polina, Mehrdad Farajtabar, Dushyant Rao, Balaji Lakshminarayanan, Nir Levine, Ang Li, Huiyi Hu, Andrew Gordon Wilson, and Razvan Pascanu

(2021). “Task-agnostic continual learning with hybrid probabilistic models.” In: *International Conference on Machine Learning (ICML) Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models Workshop*.

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. (2017). “Overcoming catastrophic forgetting in neural networks.” In: *Proceedings of the National Academy of Sciences (PNAS)* 114.13, pp. 3521–3526.

Kolouri, Soheil, Se Rim Park, Matthew Thorpe, Dejan Slepcev, and Gustavo K Rohde (2017). “Optimal mass transport: Signal processing and machine-learning applications.” In: *IEEE signal processing magazine* 34.4, pp. 43–59.

Krizhevsky, Alex (2009). *Learning multiple layers of features from tiny images*. Tech. rep.

Kullback, Solomon and Richard A Leibler (1951). “On information and sufficiency.” In: *The annals of mathematical statistics* 22.1, pp. 79–86.

Lake, Brenden M, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman (2017). “Building machines that learn and think like people.” In: *Behavioral and brain sciences* 40.

Lazova, Verica, Eldar Insafutdinov, and Gerard Pons-Moll (2019). “360-degree textures of people in clothing from a single image.” In: *International Conference on 3D Vision (3DV)*. IEEE, pp. 643–653.

LeCun, Yann, Corinna Cortes, and CJ Burges (1998). “MNIST handwritten digit database.” In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2.

Ledig, Christian, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. (2017). “Photo-realistic single image super-resolution using a generative adversarial network.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4681–4690.

Levenberg, Kenneth (1944). "A method for the solution of certain non-linear problems in least squares." In: *Quarterly of applied mathematics* 2.2, pp. 164–168.

Levoy, Marc (1990). "Efficient ray tracing of volume data." In: *ACM Transactions on Graphics (TOG)* 9.3, pp. 245–261.

Li, Chun-Liang, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov (2019). "Point cloud gan." In: *International Conference on Learning Representations (ICLR) Workshop*.

Li, Ying, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A Chapman, Dongpu Cao, and Jonathan Li (2020). "Deep learning for lidar point clouds in autonomous driving: A review." In: *IEEE Transactions on Neural Networks and Learning Systems (NNLS)* 32.8, pp. 3412–3432.

Li, Zhizhong and Derek Hoiem (2017). "Learning without forgetting." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 40.12, pp. 2935–2947.

Lim, Joseph J, Hamed Pirsiavash, and Antonio Torralba (2013). "Parsing ikea objects: Fine pose estimation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2992–2999.

Liu, Zhijian, Haotian Tang, Yujun Lin, and Song Han (2019). "Point-voxel cnn for efficient 3d deep learning." In: *Advances in Neural Information Processing Systems (NeurIPS)* 32.

Lloyd, Stuart (1982). "Least squares quantization in PCM." In: *IEEE transactions on information theory* 28.2, pp. 129–137.

Lopez-Paz, David and Maxime Oquab (2017). "Revisiting classifier two-sample tests." In: *International Conference on Learning Representations (ICLR)*.

Ma, Chao, Yulan Guo, Jungang Yang, and Wei An (2018). "Learning multi-view representation with LSTM for 3-D shape recognition and retrieval." In: *IEEE Transactions on Multimedia* 21.5, pp. 1169–1182.

Macukow, Bohdan (2016). "Neural networks–state of art, brief history, basic models and architecture." In: *IFIP international conference on computer information systems and industrial management*. Springer, pp. 3–14.

Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow (2016). "Adversarial Autoencoders." In: *International Conference on Learning Representations (ICLR)*.

Mantiuk, Rafał, Karol Myszkowski, and Hans-Peter Seidel (2015). "High Dynamic Range Imaging." In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. Wiley, pp. 1–42.

Maturana, Daniel and Sebastian Scherer (2015). "Voxnet: A 3d convolutional neural network for real-time object recognition." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 922–928.

Menéndez, ML, JA Pardo, L Pardo, and MC Pardo (1997). "The jensen-shannon divergence." In: *Journal of the Franklin Institute* 334.2, pp. 307–318.

Mescheder, Lars, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger (2019). "Occupancy networks: Learning 3d reconstruction in function space." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4460–4470.

Michałkiewicz, Mateusz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson (2019). "Deep level sets: Implicit surface representations for 3d shape inference." In: *Computing Research Repository arXiv:1901.06802*.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective*. MIT press.

– (2022). *Probabilistic Machine Learning: An introduction*. MIT Press. URL: probml.ai.

Netzer, Yuval, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng (2011). "Reading Digits in Natural Images with Unsupervised Feature Learning." In.

Nie, Yinyu, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang (2020). "Total3dunderstanding: Joint layout, object pose and mesh recon-

struction for indoor scenes from a single image." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 55–64.

Niemeyer, Michael, Lars Mescheder, Michael Oechsle, and Andreas Geiger (2019). "Occupancy flow: 4d reconstruction by learning particle dynamics." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 5379–5389.

Oh Song, Hyun, Yu Xiang, Stefanie Jegelka, and Silvio Savarese (2016). "Deep metric learning via lifted structured feature embedding." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4004–4012.

Olshausen, Bruno A and David J Field (1996). "Emergence of simple-cell receptive field properties by learning a sparse code for natural images." In: *Nature* 381.6583, pp. 607–609.

Palomer, Albert, Pere Ridao, and David Ribas (2019). "Inspection of an underwater structure using point-cloud SLAM with an AUV and a laser scanner." In: *Journal of field robotics* 36.8, pp. 1333–1344.

Pan, Sinno Jialin and Qiang Yang (2009). "A survey on transfer learning." In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359.

Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan (2021). "Normalizing flows for probabilistic modeling and inference." In: *Journal of Machine Learning Research (JMLR)* 22.57, pp. 1–64.

Papamakarios, George, Theo Pavlakou, and Iain Murray (2017). "Masked autoregressive flow for density estimation." In: *Advances in Neural Information Processing Systems (NeurIPS)* 30.

Parisi, German I, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter (2019). "Continual lifelong learning with neural networks: A review." In: *Neural Networks* 113, pp. 54–71.

Park, Jeong Joon, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove (2019). "Deepsdf: Learning continuous signed distance functions for shape representation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174.

Paschalidou, Despoina, Luc Van Gool, and Andreas Geiger (2020). "Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1060–1070.

Peng, Songyou, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger (2020). "Convolutional occupancy networks." In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 523–540.

Pons-Moll, Gerard, Sergi Pujades, Sonny Hu, and Michael J Black (2017). "ClothCap: Seamless 4D clothing capture and retargeting." In: *ACM Transactions on Graphics (ToG)* 36.4, pp. 1–15.

Popov, Stefan, Pablo Bauszat, and Vittorio Ferrari (2020). "Corenet: Coherent 3d scene reconstruction from a single rgb image." In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 366–383.

Pumarola, Albert, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari (2020). "C-flow: Conditional generative flow models for images and 3d point clouds." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7949–7958.

Qi, Charles R, Or Litany, Kaiming He, and Leonidas J Guibas (2019). "Deep hough voting for 3d object detection in point clouds." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 9277–9286.

Qi, Charles R, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas (2018). "Frustum pointnets for 3d object detection from rgb-d data." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 918–927.

Qi, Charles R, Hao Su, Kaichun Mo, and Leonidas J Guibas (2017a). "Pointnet: Deep learning on point sets for 3d classification and segmentation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 1.2, p. 4.

Qi, Charles R, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas (2016). "Volumetric and multi-view cnns for object classification on 3d data." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5648–5656.

Qi, Charles R, Li Yi, Hao Su, and Leonidas J Guibas (2017b). "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5099–5108.

Qin, Can, Haoxuan You, Lichen Wang, C-C Jay Kuo, and Yun Fu (2019). "Pointdan: A multi-scale 3d domain adaption network for point cloud representation." In: *Advances in Neural Information Processing Systems (NeurIPS)* 32.

Ranftl, René, Alexey Bochkovskiy, and Vladlen Koltun (2021). "Vision transformers for dense prediction." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 12179–12188.

Ranftl, René, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun (2020). "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.

Rebuffi, Sylvestre-Alvise, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert (2017). "icarl: Incremental classifier and representation learning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2001–2010.

Rezende, Danilo and Shakir Mohamed (2015). "Variational inference with normalizing flows." In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 1530–1538.

Roweis, Sam (1997). "EM algorithms for PCA and SPCA." In: *Advances in Neural Information Processing Systems (NeurIPS)* 10.

Rubner, Yossi, Carlo Tomasi, and Leonidas J Guibas (2000). "The earth mover's distance as a metric for image retrieval." In: *International Journal of Computer Vision (IJCV)* 40.2, pp. 99–121.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Rusu, Andrei A, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell (2016). "Progressive neural networks." In: *Computing Research Repository arXiv:1606.04671*.

Safadoust, Sadra and Fatma Güney (2021). "Self-Supervised Monocular Scene Decomposition and Depth Estimation." In: *International Conference on 3D Vision (3DV)*. IEEE, pp. 627–636.

Schmidhuber, Jürgen (1992). "Learning to control fast-weight memories: An alternative to dynamic recurrent networks." In: *Neural Computation* 4.1, pp. 131–139.

– (2015). "Deep learning in neural networks: An overview." In: *Neural networks* 61, pp. 85–117.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (2015). "Facenet: A unified embedding for face recognition and clustering." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823.

Sedaghat, Nima, Mohammadreza Zolfaghari, Ehsan Amiri, and Thomas Brox (2017). "Orientation-boosted voxel nets for 3D object recognition." In: *British Machine Vision Conference (BMVC)*.

Shin, Hanul, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim (2017). "Continual learning with deep generative replay." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Siddique, Nahian, Sidike Paheding, Colin P Elkin, and Vijay Devabhaktuni (2021). "U-net and its variants for medical image segmentation: A review of theory and applications." In: *IEEE Access*.

Singandhupe, Ashutosh and Hung Manh La (2019). "A review of slam techniques and security in autonomous driving." In: *IEEE international conference on robotic computing (IRC)*. IEEE, pp. 602–607.

Singh, Satya P, Lipo Wang, Sukrit Gupta, Haveesh Goli, Parasuraman Padmanabhan, and Balázs Gulyás (2020). "3D deep learning on medical images: a review." In: *Sensors* 20.18, p. 5097.

Sohl-Dickstein, Jascha, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli (2015). "Deep unsupervised learning using nonequilibrium thermodynamics." In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 2256–2265.

Sohn, Kihyuk (2016). "Improved deep metric learning with multi-class n-pair loss objective." In: *Advances in Neural Information Processing Systems (NeurIPS)* 29.

Song, Yang and Stefano Ermon (2019). "Generative modeling by estimating gradients of the data distribution." In: *Advances in Neural Information Processing Systems (NeurIPS)* 32.

Spurek, Przemysław, Sebastian Winczowski, Jacek Tabor, Maciej Zamorski, Maciej Zięba, and Tomasz Trzciński (2020). "Hypernetwork approach to generating point clouds." In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 9099–9108.

Stypułkowski, Michał, Kacper Kania, Maciej Zamorski, Maciej Zięba, Tomasz Trzciński, and Jan Chorowski (2021). "Representing point clouds with generative conditional invertible flow networks." In: *Pattern Recognition Letters* 150, pp. 26–32.

Su, Hang, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller (2015). "Multi-view convolutional neural networks for 3d shape recognition." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 945–953.

Sun, Xingyuan, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman (2018). "Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Suzuki, Kenji (2017). "Overview of deep learning in medical imaging." In: *Radiological physics and technology* 10.3, pp. 257–273.

Tadeusiewicz, Ryszard and Maciej Szaleniec (2015). *Leksykon sieci neuronowych*. Projekt Nauka. Fundacja na rzecz promocji nauki polskiej.

Tang, Pingbo, Daniel Huber, Burcu Akinci, Robert Lipman, and Alan Lytle (2010). "Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques." In: *Automation in construction* 19.7, pp. 829–843.

Tchapmi, Lyne, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese (2017). "Segcloud: Semantic segmentation of 3d point clouds." In: *International Conference on 3D Vision (3DV)*. IEEE, pp. 537–547.

Tipping, Michael E and Christopher M Bishop (1999). "Probabilistic principal component analysis." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3, pp. 611–622.

Tomczak, Jakub M (2021). *Deep Generative Modeling*. Springer Nature.

Vahdat, Arash, Karsten Kreis, and Jan Kautz (2021). "Score-based generative modeling in latent space." In: *Advances in Neural Information Processing Systems (NeurIPS)* 34.

Van den Oord, Aaron, Yazhe Li, and Oriol Vinyals (2018). "Representation learning with contrastive predictive coding." In: *Computing Research Repository*.

Van der Maaten, Laurens and Geoffrey E Hinton (2008). "Visualizing data using t-SNE." In: *Journal of Machine Learning Research (JMLR)* 9.11.

Van Engelen, Jesper E and Holger H Hoos (2020). "A survey on semi-supervised learning." In: *Machine Learning* 109.2, pp. 373–440.

Ven, Gido M van de and Andreas S Tolias (2019). "Three scenarios for continual learning." In: *Computing Research Repository arXiv:1904.07734*.

Villani, Cédric (2009). *Optimal transport: old and new*. Vol. 338. Springer.

Von Neumann, John and Oskar Morgenstern (2007). "Theory of games and economic behavior." In: *Theory of games and economic behavior*. Princeton university press.

Wattenberg, Martin, Fernanda Viégas, and Ian Johnson (2016). "How to Use t-SNE Effectively." In: *Distill*. URL: http://distill.pub/2016/misread-tsne.

Weese, Jürgen, Graeme P Penney, Paul Desmedt, Thorsten M Buzug, Derek LG Hill, and David J Hawkes (1997). "Voxel-based 2-D/3-D registration of fluoroscopy images and CT scans for image-guided surgery." In: *IEEE Transactions on Information Technology in Biomedicine* 1.4, pp. 284–293.

Wold, Svante, Kim Esbensen, and Paul Geladi (1987). "Principal component analysis." In: *Chemometrics and intelligent laboratory systems* 2.1-3, pp. 37–52.

Wu, Jiajun, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum (2016). "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling." In: *Advances in Neural Information Processing Systems (NeurIPS)* 29.

Wu, Wenxuan, Zhongang Qi, and Li Fuxin (2019). "Pointconv: Deep convolutional networks on 3d point clouds." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9621–9630.

Wu, Zhirong, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao (2015). "3D Shapenets: A deep representation for volumetric shapes." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920.

Xiang, Yu, Wongun Choi, Yuanqing Lin, and Silvio Savarese (2015). "Data-driven 3d voxel patterns for object category recognition." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1903–1911.

Xie, Haozhe, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang (2019). "Pix2vox: Context-aware 3d reconstruction from single and multi-view images." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 2690–2698.

Xie, Yuxing, Jiaojiao Tian, and Xiao Xiang Zhu (2020). "Linking Points With Labels in 3D: A Review of Point Cloud Semantic Segmentation." In: *IEEE Geoscience and Remote Sensing Magazine*.

Xu, Qiantong, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger (2018). "An empirical study on evaluation metrics of generative adversarial networks." In: *Computing Research Repository arXiv:1806.07755*.

Yan, Yajie, David Letscher, and Tao Ju (2018). "Voxel cores: Efficient, robust, and provably good approximation of 3d medial axes." In: *ACM Transactions on Graphics (TOG)* 37.4, pp. 1–13.

Yang, Guandao, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan (2019). "Pointflow: 3d point cloud generation with continuous normalizing flows." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 4541–4550.

Yang, Jingkang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu (2021). "Generalized Out-of-Distribution Detection: A Survey." In: *Computing Research Repository arXiv:2110.11334*.

Yuan, Wentao, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert (2018). "Pcn: Point completion network." In: *International Conference on 3D Vision (3DV)*. IEEE, pp. 728–737.

Zaidi, Julian, Jonathan Boilard, Ghyslain Gagnon, and Marc-André Carbonneau (2020). "Measuring disentanglement: A review of metrics." In: *Computing Research Repository arXiv:2012.09276*.

Zamorski, Maciej, Michał Stypułkowski, Konrad Karanowski, Tomasz Trzciński, and Maciej Zięba (2022). "Continual learning on 3D point clouds with random com-

pressed rehearsal." In: *Under Review for Computer Vision and Image Understanding (CVIU)*.

Zamorski, Maciej, Adrian Zdobylak, Maciej Zięba, and Jerzy Świątek (2019). "Generative adversarial networks: recent developments." In: *International Conference on Artificial Intelligence and Soft Computing (ICAISC)*. Springer, pp. 248–258.

Zamorski, Maciej and Maciej Zięba (2019). "Semi-supervised learning with Bidirectional GANs." In: *Asian Conference on Intelligent Information and Database Systems (ACIIDS)*. Springer, pp. 649–660.

Zamorski, Maciej, Maciej Zięba, Piotr Klukowski, Rafał Nowak, Karol Kurach, Wojciech Stokowiec, and Tomasz Trzciński (2020a). "Adversarial autoencoders for compact representations of 3D point clouds." In: *Computer Vision and Image Understanding (CVIU)* 193, p. 102921.

Zamorski, Maciej, Maciej Zięba, and Jerzy Świątek (2020b). "Comparison of Aggregation Functions for 3D Point Clouds Classification." In: *Asian Conference on Intelligent Information and Database Systems (ACIIDS)*. Springer, pp. 504–513.

– (2020c). "Generative Modeling in Application to Point Cloud Completion." In: *International Conference on Artificial Intelligence and Soft Computing (ICAISC)*. Springer International Publishing, pp. 292–302. ISBN: 978-3-030-61401-0.

Zbontar, Jure, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny (2021). "Barlow twins: Self-supervised learning via redundancy reduction." In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 12310–12320.

Zhang, Yinda and Thomas Funkhouser (2018). "Deep depth completion of a single rgb-d image." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 175–185.

Zheng, Chuanxia, Tat-Jen Cham, and Jianfei Cai (2019). "Pluralistic image completion." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1438–1447.

Zhou, Yin and Oncel Tuzel (2018). "Voxelnet: End-to-end learning for point cloud based 3d object detection." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4490–4499.

List of Figures

List of Tables

List of Algorithms