UNIVERSITY OF BIRMINGHAM

Note: Please sign this form and present it to the Bindery with your thesis, so that it can be bound at the front of the Library deposit copy

RESEARCH THESIS
Author's Declaration

Full name (block capitals, surname first): ……SNEHA SAHA……

Full title of thesis/dissertation (block capitals):……LEARNING-BASED GENERATIVE REPRESENTATIONS FOR AUTOMOTIVE DESIGN OPTIMIZATION…….....

……..…….

College/School/Department (block capitals): …SCHOOL OF COMPUTER SCIENCE……

Date of award of degree (leave blank):

1. I understand that one printed and one electronic copy of my thesis/dissertation (the Work) will be deposited in the University Library (the Library) and in a suitable electronic repository, for permanent retention.
2. Without changing the content, the Library or any third party with whom it has an agreement to do so, may convert either copy into a format or medium for the purpose of long-term retention, accessibility and preservation.
3. The Library will publish, and/or arrange with appropriate third parties for the non-exclusive publication of, a bibliographic description of the thesis/dissertation, and the author's abstract.
4. Unless arrangements are made to the contrary, (see paragraph 6. below), the Library is authorised to make the Work available for consultation in the Library, and via a recognised inter library loans system. The Library is authorised to make the electronic copy of the Work freely accessible to individuals and institutions - including automated agents - via the Internet.
5. Rights granted to the University of Birmingham through this agreement are entirely non-exclusive. I retain all my rights in the Work in its present version or future derivative works.
6. I understand that I may apply to the University to retain the right to withhold access to the content of my thesis/dissertation. Access to the paper version may be withheld for a period which shall not normally exceed four calendar years from the congregation at which the degree is conferred. The actual length of the period will be specified in the application, together with the precise reasons for making that application. The electronic copy may be withheld from dissemination via the web or other networks for any period.
7. I have obtained permission for any use made of substantial amounts of published or unpublished copyright material (text, illustrations, etc) where the rights are owned by a third party, other than as permitted under either The Copyright Designs and Patents Act 1988 (as modified by any related or successor legislation) or the Terms and Conditions of any Licence governing its use.
8. The content of the copies I shall deposit with the Library will be the final version of my thesis, as approved by the Examiners.
9. I understand that the Library and administrators of any electronic theses repository do not hold any obligation to take legal action on behalf of myself, or other rights holders, in the event of a breach of intellectual property rights, or any other right, in the material deposited.
10. I understand that, in the event of my thesis/dissertation being not approved by the Examiners, this declaration will become null and void.

Signature: ……Sneha Saha…… **Date:** ……21.1.23……

For Library use (please leave blank):

Classmark: **Accession number:** **Control number:**

eTheses Repository url:

Learning-Based Generative Representations for Automotive Design Optimization

By

SNEHA SAHA

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

Department of Computer Science
University of Birmingham
December 2022

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186 (ECOLE).

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. Stefan Menzel, Dr. Leandro L. Minku, Prof. Bernhard Sendhoff, and Prof. Xin Yao for their guidance and inspiration. I am thankful for the regular discussions and feedback that I received from Dr. Stefan Menzel and Dr. Leandro L. Minku throughout my Ph.D. journey. A special thanks to Prof. Bernhard Sendhoff for his insightful explanations of difficult concepts in an easy, understandable way. I thank Prof. Yaochu Jin and Prof. Yew Soon Ong for considering to review my thesis.

It has been an amazing experience working at Honda Research Institute Europe (HRI-EU). I would like to thank my colleagues at HRI-EU for the pleasant atmosphere and fruitful discussions. Apart from the technical support for my research, I would also like to thank the administration team of HRI-EU and University of Birmingham for their support throughout my Ph.D. journey.

I would also like to thank my friends for their constant support during the time of the pandemic. Finally, I wholeheartedly thank my parents and sister for always believing in me and encouraging me to do my best.

Abstract

In automotive design optimizations, engineers intuitively look for suitable representations of CAE models that can be used across different optimization problems. Determining a suitable compact representation of 3D CAE models facilitates faster search and optimization of 3D designs. Therefore, to support novice designers in the automotive design process, we envision a cooperative design system (CDS) which learns the experience embedded in past optimization data and is able to provide assistance to the designer while performing an engineering design optimization task. The research in this thesis addresses different aspects that can be combined to form a CDS framework.

First, based on the survey of deep learning techniques, a point cloud variational autoencoder (PC-VAE) is adapted from the literature, extended and evaluated as a shape generative model in design optimizations. The performance of the PC-VAE is verified with respect to state-of-the-art architectures. The PC-VAE is capable of generating a continuous low-dimensional search space for 3D designs, which further supports the generation of novel realistic 3D designs through interpolation and sampling in the latent space. In general, while designing a 3D car design, engineers need to consider multiple structural or functional performance criteria of a 3D design. Hence, in the second step, the latent representations of the PC-VAE are evaluated for generating novel designs satisfying multiple criteria and user preferences. A seeding method is proposed to provide a warm start to the optimization process and improve convergence time. Further, to replace expensive simulations for performance estimation in an optimization task, surrogate models are trained to map each latent representation of an input 3D design to their respective geometric and functional performance measures. However, the performance of the PC-VAE is less consistent due to additional regularization of the latent space.

Thirdly, to better understand which distinct region of the input 3D design is learned by a particular latent variable of the PC-VAE, a new deep generative model is proposed (Split-AE), which is an extension of the existing autoencoder architecture. The Split-AE learns input 3D point cloud representations and generates two sets of latent variables for each 3D design. The first set of latent variables, referred to as *content*, which helps to represent an overall underlying structure of the 3D shape to discriminate across other semantic shape categories. The second set of latent variables refers to the *style*, which represents the unique shape part of the input 3D shape and this allows grouping of shapes

into shape classes. The reconstruction and latent variables disentanglement properties of the Split-AE are compared with other state-of-the-art architectures. In a series of experiments, it is shown that for given input shapes, the Split-AE is capable of generating the content and style variables which gives the flexibility to transfer and combine style features between different shapes. Thus, the Split-AE is able to disentangle features with minimum supervision and helps in generating novel shapes that are modified versions of the existing designs.

Lastly, to demonstrate the application of our initial envisioned CDS, two interactive systems were developed to assist designers in exploring design ideas. In the first CDS framework, the latent variables of the PC-VAE are integrated with a graphical user interface. This framework enables the designer to explore designs taking into account the data-driven knowledge and different performance measures of 3D designs. The second interactive system aims to guide the designers to achieve their design targets, for which past human experiences of performing 3D design modifications are captured and learned using a machine learning model. The trained model is then used to guide the (novice) engineers and designers by predicting the next step of design modification based on the current applied changes.

Table of contents

List of Symbols

Roman Symbols

(s, t, u)	Coordinates of the lattice parametric space
(x, y, z)	Coordinates of the Cartesian space
α	Hyper-parameter to balance the reconstruction loss function of the variational autoencoder
$\bar{\Psi}_i$	Normalized objective function of the multi-objective optimization
β	Hyper-parameter to balance the Kullback-Leibler divergence loss function of the variational autoencoder
ϵ	A random variable sampled from a standard normal distribution
λ	Interpolating factor
$\mathcal{D}_b$	Set of 3D point clouds in a batch
$\mathcal{D}_{test}$	Set of 3D point clouds considered as a test set
$\mathcal{D}_{train}$	Set of 3D point clouds considered as a training set
$\mathcal{F}_a^j$	Feature matrix of the j-th activation layer of the encoder layer
$\mathcal{F}_{norm}^j$	Normalized feature matrix of the j-th activation layer of the encoder layer
$\mathcal{G}$	Set of newly generated shapes using the variational autoencoder
$\mathcal{K}$	Number of domains from which 3D shapes are selected
$\mathcal{L}_{DC}$	Domain classifier loss function
$\mathcal{L}_{KL}$	Kullback-Leibler divergence loss function
$\mathcal{L}_{recon}$	Reconstruction loss function

$\mathcal{L}_{SC}$	Style classifier loss function
$\mathcal{N}$	Guassian Distribution
$\mathcal{S}$	Complete set of 3D point clouds that is considered as the dataset
$\mathcal{X}$	Number of classes from each domain from which 3D shapes are selected
μ	Mean layer of the variational autoencoder
μ_{bn}	Mean value of the batch normalization layer
$\Psi_i(\vec{z})$	i-th objective function of the multi-objective optimization
ρ	Distance ratio function
σ	Standard deviation layer of the variational Autoencoder
σ_{bn}	Standard deviation value of the batch normalization layer
$D(x)$	Discriminator of the generative adversarial network
$G(z)$	Generator of the generative adversarial network
p_i	Probability of the i-the latent vector $\vec{z}_i$
$\vec{\mu}$	Mean vector generated by the variational autoencoder
$\vec{\sigma}$	Standard deviation vector generated by the variational Autoencoder
$\vec{z}$	Latent representation vector a 3D design
$\vec{z}_c$	Latent vector representing content codes of the input 3D shape
$\vec{z}_s$	Latent vector representing style codes of the input 3D shape
$C(\vec{z}; w_1)$	Cost function of the weighted-sum optimization
C_i	Class labels of 3D point cloud shapes
CD_{test}	Reconstruction loss calculated using Chamfer Distance for the test set samples
CD_{train}	Reconstruction loss calculated using Chamfer Distance for the training set samples
cp_i	i-th control point of the FFD lattice
d_ϕ	Decoder of the variational autoencoder parameterized by ϕ

e_θ	Encoder of the variational autoencoder parameterized by θ
F_x	Drag Coefficient
$I^{'}(z_i^{'}, y)$	Mutual information estimator between the latent variable $z_i^{'}$ and output variable y
L_z	Dimension of the latent vector $\vec{z}$
$p(x)$	Probability distribution of x
$p(z)$	Prior distribution of the latent variables
$p_i \in x_{in}$	i-th point of the input shape x_{in}
$p_o \in x_o$	o-th point of the output shape x_o
r	Ratio for augmented data
R^2	R-Square
V	Volume
w	Weight in the weighted-sum optimization
x_{in}	Point cloud representation of an input 3D design
x_o	Point cloud representation of the reconstructed 3D design
x_{T1}, x_{T2}	Target shapes
Z_C	Content latent space of the Split-AE
z_i	i-th latent variable of the latent vector $\vec{z}$
z_{li}	Lower limit of the i-th latent variable
Z_s	Style latent space of the Split-AE
z_{ui}	Upper limit of the i-th latent variable

Acronyms / Abbreviations

AAE	Adversarial Autoencoder
AE+GMM	Gaussian Mixture Model Trained on Latent Space of Autoencoder
AE	Autoencoder

AI	Artifical Intelligence
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BL	Baseline Model
BN	Batch Normalization
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CD	Chamfer Distance
CDS	Cooperative Design System
CFD	Computational Fluid Dynamic
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CNN	Convolutional Neural Network
CS	Computational Synthesis
DBN	Deep Belief Network
DL	Deep Learning
DM	Decision Maker
EA	Evolutinary Algorithm
ECOLE	Experience-based Computation Learning to Optimize
EMD	Earth Mover Distance
FFD	Free Form Deformation
GAN	Generative Adversarial Network
GDL	Geometric Deep Learning
GMM	Guassian Mixture Model
GNN	Gaph Neural Network
GO	Global Optimization
GP	Gaussian Process

GPU	Graphics Processing Units
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
HMM	Hidden Markov Model
HV	Hyper Volume
IGD	Inverted Generational Distance
KL	Kullback-Leibler
LSTM	Long Short Term Memory
LV	Latent Variable
MAE	Mean Absolute Error
MCDM	Multi-Criteria Decision Making
MD	Mean Distance
MI	Mutual Information
MLE	Maximum Log-Likelihood Estimation
MLP-AE	Multi-Layer Perceptron Trained on the Latent Representation of the Autoencoder (AE)
MLP-VAE	Multi-Layer Perceptron Trained on the Latent Representation of the Variational Autoencoder (VAE)
MLP	Multi Layer Perceptron
MMD-CD	Minimum Matching Distance Calculated using Chamfer Distance
MOACO	Multi-Objective Ant Colony Optimization
MOEA	Multi-Objective Evolutionary Algorithm
MOGA	Mutli-Objective Genetic Algorithm
MOO	Multi-Objective Optimization
MSD	Mean Square Distance
MSE	Mean Square Error

NFE	Number of Function Evaluation
NPGA	Niched Pareto Genetic Algorithm
MOEA	Non-dominated Sorting Genetic Algorithm
NSGA	Non-dominated Sorting Genetic Algorithm
OFF	Object File Format
OMOPSO	Optimized Multi-Objective Particle Swarm Optimization
PACF	Partial Auto-Correlation Function
PAES	Pareto Archived Evolution Strategy
PC-AE-CD	Point Cloud Autoencoder Trained with Chamfer Distance
PC-AE-MSD	Point Cloud Autoencoder Trained with Mean Square Distance
PC-AE	Point Cloud Autoencoder
PC-VAE-CD	Point Cloud Variational Autoencoder Trained with Chamfer Distance
PC-VAE-MSD	Point Cloud Variational Autoencoder Trained with Mean Square Distance
PC-VAE	Point Cloud Variational Autoencoder
PCC	Pearson Correlation Coefficient
PI	Problem Instance
RBM	Restricted Botlzmann Machine
ReLU	Rectified Linear Units
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RQ	Research Question
SHGO	Simplicial Homology Global Optimization
SOI	Solution Of Interest
SPEA	Strength Pareto Evolutionary Algorithm
Split-AE	Split Autoencoder

SP	Spacing
SUV	Sports Utility Vehicle
SWM	Shrink-Wrapping Method
TSMO	Target Shape Matching Optimization
UMAP	Uniform Manifold Approaximation and Projection
VAE	Variational Autoencoder
VEGA	Vector Evaluated Genetic Algorithm
WSM	Weighted Sum Method

Chapter 1

Introduction

During the process of digital development of new automotive designs, engineers intuitively exploit their experience across several product generation cycles to address current optimization challenges. However, often synergies between the optimization instances of similar problems are missing, e.g., current designs have different parameterizations compared to past designs, or different performance metrics are used. Additionally, at each product generation cycle, there is an increasing number of multidisciplinary requirements that must be considered before finalizing a car design prototype.

Car prototypes generated by automotive designers at each generation cycle are often modified versions of the previous generations with added functionalities and improved performances (Fig. 1.1). Mostly, automotive designers use computer-aided design (CAD) and engineering (CAE) tools to manually generate virtual prototypes, modify or update existing prototypes according to their design vision, and engineers utilize these prototypes for design optimization tasks. Both engineers and designers intuitively leverage knowledge from past experiences to solve new optimization problems targeting faster convergences. Therefore, in the context of automotive design optimization, it is beneficial to learn the notion of experience from past optimization data to help in improving the solutions in future optimizations. These challenges are addressed in the Experience-based Computation: Learning to Optimize (ECOLE)[1] project, where the target of the project is the research and development of novel methods to integrate experience extracted by machine learning algorithms along a line of similar optimization problems in the automotive domain.

Prior research demonstrates different approaches for learning from past experiences. Most popular approaches involve extracting knowledge or experience from past solutions [1, 2], learning from past search experiences [3], and reuse of past model-based information [4]. Here, we consider the notion of experience can be defined in the following two ways: The first procedure refers to the experience embedded in (engineering) data, which is

[1]This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186 (ECOLE). https://cordis.europa.eu/project/id/766186.

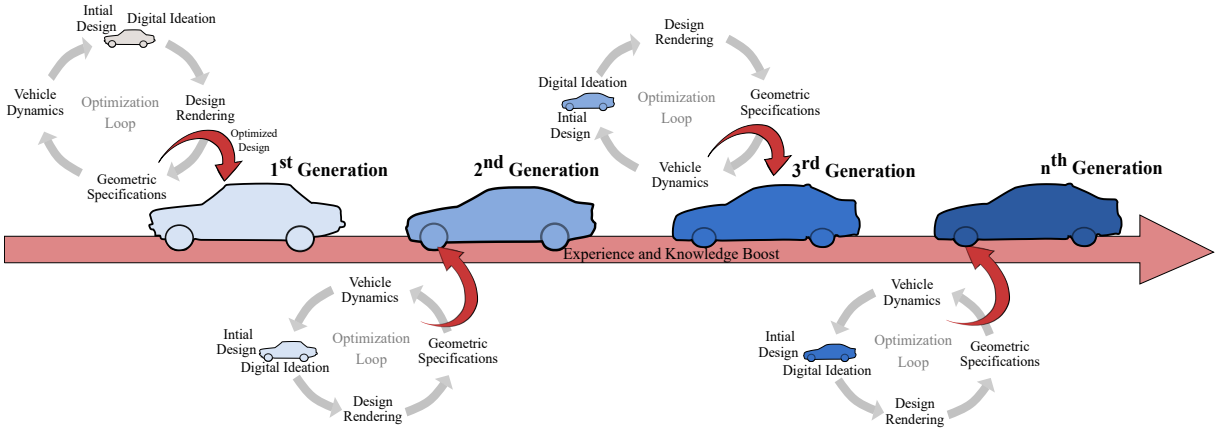


Fig. 1.1 Finalized car designs for each product generation cycle go through several multi-criteria based optimizations. The vision is to transfer the experience of car design optimization from previous generations to future design optimizations in the automotive domain.

collected over the course of optimization runs. A systematic approach to learn, model and transfer past (or benchmark) designs to new optimization problems accelerates the design finalization duration. For example, learning design features and simulated performances from prior optimization runs allows both designers and engineers to sample, combine, and assess the performances of current novel solutions faster and more accurately, which accelerates the design process. The second procedure refers to the human experience, which resembles the engineers' approach of performing past design optimizations and is harder to capture. Hence, in this research, the focus is on the former approach.

The challenging part of exploiting the experience embedded in engineering data is the high dimensionality of existing CAE designs that are collected from prior optimization runs. This increases the complexity of the design domain and hinders the search and analysis of optimization results. Further, engineers usually select design features for optimization based on their domain expertise, and often these design features differ from one optimization to another within the same domain. Even though engineers utilize CAD/E tools for most of the current engineering designs, mining design features from CAE data is challenging due to the sparse, unstructured, high-dimensional nature of the data samples. Therefore, generating an efficient low-dimensional compact representation of 3D designs is a key step for extracting knowledge from prior engineering data and optimization cycles, and transferring it across new optimization problems.

1.1 Research Overview

1.1.1 Problem Formulation

In automotive digital development, simulations are used to evaluate design properties such as aerodynamic drag efficiency, thermodynamic properties, stiffness of structural

components, and crashworthiness. Besides technical aspects, successful and innovative designs rely on human engineers in the loop to ideate and realize conceptual directions. To support designers at each product generation cycle, we envision a cooperative design assistance system (CDS), which will guide the designer by generating potential design alternatives. This provides freedom to the designer to rethink, discuss, and adapt promising product directions. Formulating a CDS framework is related to solving the computational synthesis (CS) problem, which focuses on automatizing methods in engineering design optimizations.

Overview of a Computational Synthesis Approach

Computational synthesis (CS) results in building either a fully automated or an interactive approach to generate a range of design alternatives, which accelerates convergence in optimizations. Ideally, CS [5, 6] is invoked in situations where achieving a solution requires the generation and evaluation of countless alternatives or when human designers are often in doubt or uncertain about the next course of action to perform. Many CS approaches have benefited from the application of optimization algorithms, such as evolutionary algorithms (EAs) [7–9]. A common model for computational design synthesis using EAs is presented in Fig. 1.2, where the flowchart is divided into four main steps: representation, generation, evaluation, and guidance.

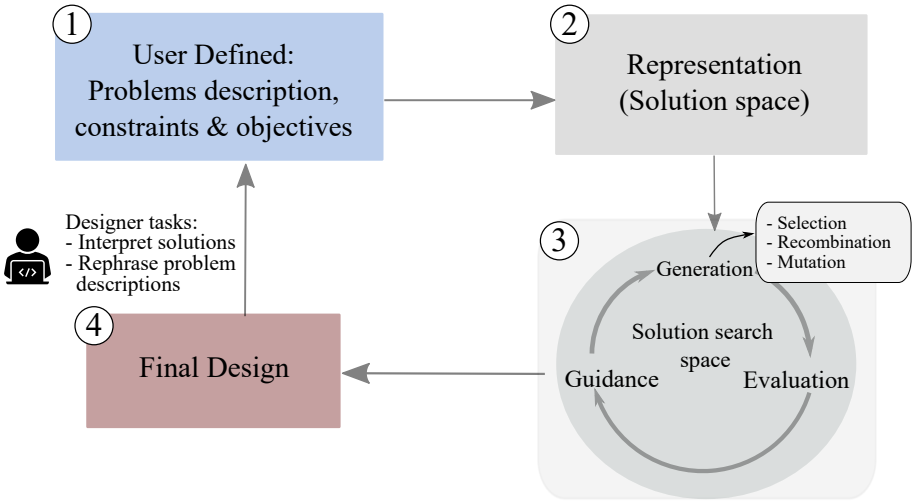


Fig. 1.2 Overview of computational design synthesis for generating automotive designs. Adapted from [6].

The first step (Fig. 1.2) involves defining objective functions (design goals), and constraints. The representation of a design effects significantly on the success of any optimization. Hence, based on the problem descriptions, the second step is to define a suitable *compact representation* that captures the forms or attributes of the 3D designs. These compact representations are also known as decision variables, which form the new solution space for any search and optimization task. After the problem and solution

space are defined, an optimization involving multiple criteria is performed to determine optimal final solutions. For example, a two-objective design optimization can be performed by minimizing or maximizing two objective functions $f_1(\vec{z})$ and $f_2(\vec{z})$, where $\vec{z}$ is a n-dimensional decision variable vector that represents a 3D design. While the solution space is generally restricted by a series of constraints and bounds on the decision variables, the target of the optimization is to find a vector $\vec{z^*}$ that minimizes both objective functions.

Next, in the solution search space, new candidate solutions are generated using selection and crossover of existing solutions. Each generated candidate solution is assessed in the *evaluation* step to determine how well it matches the objectives and constraints of the given optimization problem. Based on the performance of the generated solutions, *guidance* or feedback is implemented to improve the search process and to help the optimization algorithm to find better solutions in the next iteration. Guidance drives the generation process in a given direction to improve the generation of novel, interesting solutions. Lastly, after several iterations, the final solutions are generated, which are shown to the designer. In the final step, the human designer provides feedback on the current optimization approach and based on that, the problem description is reformulated iteratively.

Conceptual Framework

To accomplish a CDS framework following the computational synthesis steps, the first challenge is to determine a compact parameterization of CAE models that is common to a large set of optimizations and retains the knowledge and experience of the designer. Often in engineering applications, the representation of 3D CAE models is not canonical and is often tailored based on the expertise of the engineers. Designers or engineers manually select the design parameters based on their experience, objectives and domain of optimization. However, design parameters often differ from one optimization to another, and even between problems that belong to the same domain. Additionally, the representation changes over time due to higher fidelity of models and adaptation of parameters based on optimization scenarios. Therefore, selecting a handcrafted design parameterization depends on the expertise of engineers, regardless of the selected method. This often leads to the generation of sub-optimal solutions due to limited knowledge about the problem domain and makes it difficult if not impossible to extract the knowledge from a large design data set.

Commonly used design parameterization techniques involve basis function, polynomial spline, computer-aided design (CAD) and free form deformation (FFD)[10]. Direct manipulation of FFD has several advantages when combined with optimizations [11]. Although FFD provides an efficient parameterization approach for shape deformations, knowledge about past experience is not incorporated in this set-up. Hence, the first key element to building a CDS is to identify an efficient design parameterization common to

all design optimizations, that is capable of incorporating knowledge from engineers and represents 3D designs with fewer parameters.

The parameterized designs constitute the optimization search space, where the design candidate solutions are best organized in such a way that solutions with similar configurations are close to each other. Often, for automotive design generation tasks, the final design needs to satisfy multiple constraints. To generate this final design which satisfies user-defined objectives and constraints, an iterative search and evaluation of design solutions needs to be performed in the solution space. Additionally, the generated designs should be novel and realistic shapes that are feasible for engineering applications. Therefore, the research in this thesis addresses these different steps for building a CDS framework.

1.1.2 Research Questions

Apart from the described need in automotive design, the motivation for developing a CDS framework for automotive designers and engineers arises from observing the existing 2D design assistance frameworks, such as ShadowDraw [12] and SketchRNN [13]. These frameworks learn from experience embedded in prior collected drawing data and utilize machine learning models for providing real-time guidance to human users while drawing online. ShadowDraw relies on a database of hand-drawn images of human sketches and provides feedback to novice users while sketching a new drawing by offering opaque visual layers of potentially similar objects (Fig. 1.3a). SketchRNN is a generative, deep neural network trained on a database of hand-drawn sequences by human users (Fig. 1.3b). Training a generative model has several applications, from assisting the creative process of an artist to helping to teach students how to draw by proposing alternative design ideas. Thus, both frameworks use the notion of experience from past 2D hand-drawn data to learn, model and assist with current design optimization tasks.

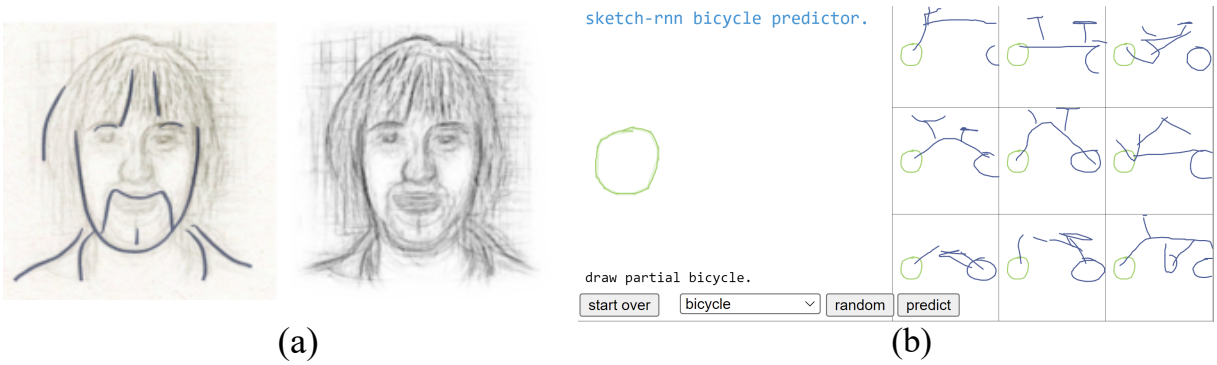


Fig. 1.3 a)ShadowDraw: The left image indicates the user's view of the shadow image, and the right side is the shadow image searched from the data set. b) SketchRNN: The user starts drawing a bicycle on the left side of the framework, and the model predicts the rest of the parts to complete the partial bicycle. Image sources: [12, 13].

With recent advancements in *geometric deep learning* (GDL) methods, deep generative models like (variational) autoencoders simultaneously learn high-dimensional geometric data and extract useful features, which help to represent a 3D design with a compact set of design parameters. When trained on 3D designs, deep generative models learn features that hold information about the structural properties of 3D designs and are transferable between different designs. Hence, these methods have the potential to learn from prior engineering data and represent each 3D design with a compact set of design parameters that will accelerate the search and optimization of 3D designs. Therefore, following the concept of an experience-based design assistance framework in the 2D domain, the driving practical vision for this thesis reads as follows:

> Developing core components of an experience-based *cooperative design system* (CDS), where generative models are used to provide guidance or potential alternatives, giving the designer the freedom to rethink, discuss, and adapt promising product directions.

Different from the 2D assistance frameworks that are used for creative applications, an experience-based design assistance framework for automotive design explorations should support the generation of realistic 3D designs satisfying multi-disciplinary requirements. Existing tools like Maya [14] and CAD support the exploration of 3D design ideas by providing the flexibility to modify 3D shapes easily. However, engineers solely need to depend on their expertise to operate these tools, due to the lack of collaborative assistance. Hence, the aim for developing the CDS framework is to utilize generative models to learn from prior available engineering data and create a common unified search space for different optimization tasks. In general, prior engineering data refers to 3D CAE models and there are multiple forms of representations of 3D geometries for training generative models. Therefore, the first research question answers the type of 3D representation that will be used in the context of this research (Chapter 2).

RQ 1. Which 3D representation is suitable for efficient training of deep learning models?

With recent advancements in deep learning algorithms, generative models learn the patterns of the input data in an unsupervised approach and encode a meaningful low-dimensional latent manifold of the externally observed input data. A generative model creates a probabilistic latent space, such that new examples can be generated by sampling from the latent space. These newly generated 3D designs can be given back as suggestions to provide inspiration for design exploration. Yet, the use of these deep generative models for engineering optimizations is sparsely addressed in the literature. Hence in this research, the second challenge is to investigate which model is the most suitable deep generative model for learning 3D engineering design data (Chapter 3). To utilize the generative model for an engineering task, the reconstruction and shape-generative performance of the generative model needs to be evaluated. This leads us to our next research question:

RQ 2. How well can a generative model learn a compact representation of 3D designs for engineering optimization? And is this model suitable for generating diverse unseen designs that are useful for engineering applications?

Another challenge in an automotive design process is the need to consider multi-disciplinary criteria. These comprise of aesthetic design targets as well as engineering criteria, such as aerodynamic and structural efficiency. Even though CAE/ D tools facilitate performing modifications of 3D designs, the necessity to simultaneously reach various potential targets leads to considerable complexity in the design process. Despite the shape-generative capability of generative models, the appropriateness of the latent manifold for performing multi-criteria-based design optimizations is less explored in the literature (Chapter 4). Moreover, the generative models are trained in an unsupervised fashion. Hence, a detailed analysis of the learned latent variables in terms of representing structural and functional information of 3D designs is currently missing (Chapter 4). The following research questions are formulated to answer whether latent manifolds are fit for multi-criteria-based design optimization.

RQ 3. How can the data-driven latent space be utilized to generate design solutions that satisfy multiple criteria?

RQ 4. Does the latent space of the deep generative models hold relevant information about the structural or functional properties of 3D designs? Can a surrogate model be trained on the latent manifold to replace expensive performance simulation or function evaluations in a design optimization problem?

However, even if generative models learn meaningful features of 3D designs, these learned features lack interpreting ability, i.e., a single feature that does not correspond to a distinctive interpretable characteristic of a 3D design. In the context of novel 3D shape generation, the identification of these interpretable features would enhance the shape-generative capability of the generative model by providing the freedom to transfer or combine different features to generate novel design variations (Chapter 5), which leads to the following research question,

RQ 5. How can the design features learned by a deep generative model represent different interpretable aspects of 3D designs? How can we transfer the unique parts or design features of a 3D shape to another shape, such that the recombination of design features from two distinctive shapes generates a novel shape combining the essence of both 3D designs?

Apart from building a suitable generative model for efficient search and optimization of 3D designs, the research in this dissertation is also motivated by the task of the designer.

This leads us to investigate the use of the generative model for assisting designers in a design concept generation task. Generally, for new design ideation, the designer starts either from scratch or an initial design or may choose some alternative methods to modify designs, such that they match his/her targets. Therefore, to provide collaborative assistance to the designer, an interactive framework is needed to guide the designer for target-oriented design exploration in the latent manifold (Chapter 6). This approach of latent space-based collaborative assistance exploits the experience embedded in the design data collected from past optimization histories. However, an alternate perception of experience in a design task refers to tracking the designers' approach to similar past optimization tasks, which remains mostly untouched in this research. Hence, in addition to the engineering data-driven assistance framework, a practical scenario-driven study is done in Chapter 6 to capture past human-experience data. Further, a deep learning model is used to learn the captured human sequence data and to predict future design steps based on the history of currently performed design modifications. Thus, depending on the type of experience data, we investigate and develop different forms of assistance that can support a designer in a design concept generation task, which eventually leads to the research questions as follows:

RQ 6. How to exploit and learn engineering experience from past design optimization tasks using DL models? And how to utilize the trained DL model for formulating different assistance to collaboratively help designers in a present design ideation task?

1.2 Thesis Outline

The remainder of the thesis is outlined based on the proposed research questions. Towards developing an experience-based design assistance framework for engineering optimization, the scope of the present dissertation is divided into three steps, which are addressed in different chapters in this thesis (Fig. 1.4). The first step involves the pre-processing step, where we determine the representation type of the 3D CAE models. The second step involves the main components of this research, which involves building the generative model for creating the low-dimensional search space of 3D designs. This search space is evaluated for search and optimization of 3D designs. In the third step, an interactive assistance framework is set up for generating 3D designs based on user preferences. Each chapter addresses the following works:

Chapter 2 provides a review of the 3D representations and introduces the background of the generative models from the field of geometric deep learning. Eventually, this section comes to a decision regarding the preferred 3D shape representation for this

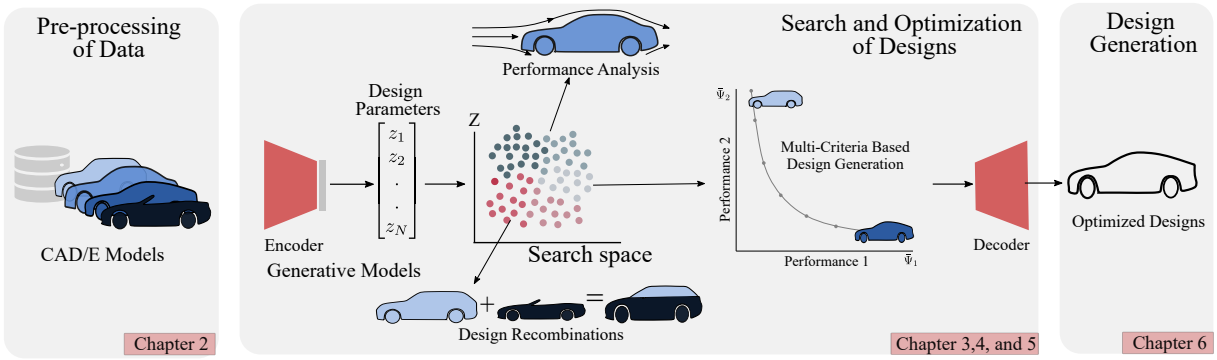


Fig. 1.4 Steps of an experience-based design optimization framework that are addressed in this dissertation.

research ($RQ1$) and deciding the type of generative model suitable for learning the selected 3D representation.

Chapter 3 introduces the proposed generative model, i.e., the point cloud variational autoencoder (PC-VAE) for learning 3D point cloud representations ($RQ2$). The network learns without supervision to create a low-dimensional latent manifold of 3D representations. Lastly, the reconstruction and shape generative capability of the PC-VAE is evaluated for engineering applications.

Chapter 4 investigates the information learned by the PC-VAE from the previous chapter. The first section analyzes the feasibility of the latent manifold for performing multi-objective optimizations of 3D designs, and a method involving latent space knowledge is proposed to improve the convergence of optimization algorithms ($RQ3$). The second section focuses on analyzing the information content in the latent representations and on using these representations to train surrogate models for replacing expensive simulation in optimization tasks ($RQ4$).

Chapter 5 addresses the challenges of controlling interpretable aspects of 3D shapes with latent variables. A novel deep generative architecture is introduced to learn disentangled features, such that latent variables map to the common and unique shape parts of different car designs ($RQ5$).

Chapter 6 combines the proposed methods from prior chapters to set up an interactive framework for providing suggestions to the designer in a design optimization task ($RQ6$). Additional to this 3D design data-driven approach of creating a cooperative framework, a preliminary investigation is done on modeling prior human-experience data to generate an alternative way of suggesting designs.

Chapter 7 concludes the thesis and gives an overview of the outlook and directions for future work.

1.3 Thesis Content Published in Peer Reviewed Papers

Part of the content of this dissertation has been published as follows:

S. Saha, T. Rios, L. L. Minku, X. Yao, Z. Xu, B. Sendhoff, and S. Menzel, "Optimal Evolutionary Optimization Hyper-parameters to Mimic Human User Behavior", in IEEE Symposium Series on Computational Intelligence, SSCI, 2019, pp. 858-866.

S. Saha, T. Rios, S. Menzel, B. Sendhoff, T. Bäck, X. Yao, and P. Wollstadt, "Learning Time-Series Data of Industrial Design Optimization using Recurrent Neural Networks", in International Conference on Data Mining Workshops, ICDMW, 2019, pp. 785-792.

S. Saha, L. L. Minku, X. Yao, Z. Xu, B. Sendhoff, and S. Menzel, "Exploiting Linear Interpolation of Variational Autoencoders for Satisfying Preferences in Evolutionary Design Optimization", in IEEE Congress on Evolutionary Computation, CEC, 2021, pp. 1767-1776.

S. Saha, T. Rios, L. L. Minku, B. Stein. P. Wollstadt, X. Yao, T. Bäck, B. Sendhoff, and S. Menzel, “Exploiting Generative Models for Performance Predictions of 3D Car Designs", in Symposium Series on Computational Intelligence, SSCI, 2021, pp. 1-9.

S. Saha, S. Menzel, L. L. Minku, X. Yao, B. Sendhoff, and P. Wollstadt, "Quantifying The Generative Capabilities Of Variational Autoencoders For 3D Car Point Clouds", in Symposium Series on Computational Intelligence, SSCI, 2020, pp. 1469-1477.

S. Saha, L. L. Minku, X. Yao, B. Sendhoff, and S. Menzel, “Exploiting 3D Variational Autoencoders For Interactive Vehicle Design", in International Design Conference 2022 (pp. 1747-1756).

S. Saha, L. L. Minku, X. Yao, B. Sendhoff, and S. Menzel, “Split-AE: An Autoencoder-based Disentanglement Framework for 3D Shape-to-shape Feature Transfer", in International Joint Conference on Neural Networks, IJCNN, 2022. (Accepted).

Chapter 2

Background on 3D Representations and Geometric Deep Learning Models

As discussed in the previous chapter, learning a compact representation of 3D design features that are suitable and transferable between different design optimization problems is a promising step to accelerate the search and convergence in design optimizations. The first step to learn design features from existing CAE models is to choose a baseline representation of 3D designs, and the second step is to determine the type of deep learning (DL) based generative model that can learn from the selected type of 3D representation. These two pre-processing steps constitute the prerequisite for learning the notion of experience from benchmark CAE data in an offline fashion. Therefore, in this chapter, a survey of the current state-of-the-art in 3D representation is presented. This helps us to determine which representation would be most suitable for research in this thesis ($RQ1$). Further, the details and background of the existing generative models are reviewed that learn on the chosen 3D representation.

The rest of the chapter is outlined as follows: In Section 2.1, commonly used 3D representations are introduced, and their properties are discussed, especially focusing on training deep learning models that takes these 3D representations as input. In addition to this, the available benchmark data sets of 3D models are introduced (Section 2.2). The next Section 2.3, introduces the state-of-the-art deep learning models for learning 3D representations, mainly using generative models.

2.1 Types of 3D Representations

The difficulty of a learning task using the DL approach can vary significantly depending on the choice of the data representation. Having a representation that is well suited to the

particular task and data domain can significantly improve the performance of the chosen model [15]. Mostly, digital 3D data is created through a manual design process, using Computer Aided Design (CAD) tools, or by reverse engineering of prototypes using modern scanning technology. 3D data can have various representations, where the structure and the geometric properties vary from one representation to another. The representations are either Euclidean or non-Euclidean. The former signifies that the data points are positioned on a fixed grid like structure, e.g., images and parametric surfaces. While non-Euclidean representations mean data points are located at an arbitrary position in the 3D space and do not have a global parameterization. Each of the representations comes with its own advantages and disadvantages. The most used 3D geometric representations that are utilized in engineering applications are categorized into polygonal mesh, voxel, and point clouds (Fig. 2.1).

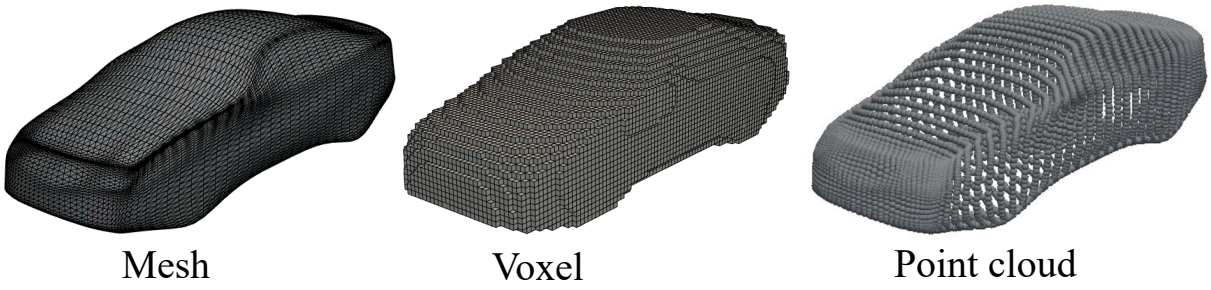


Fig. 2.1 Example of three types of non-Euclidean representations.

Polygonal Meshes: Polygonal meshes are the most efficient 3D representation in terms of capturing object details and memory storage. A polygonal mesh is a collection of vertices, edges, and faces that define the shape of a 3D object with polygonal surfaces and sharp corners. The mesh contains a connectivity list that describes how each vertex is connected to one another, and a normal vector is assigned to every vertex.

In the engineering domain, meshes are mainly utilized for simulation [16] and manufacturing tasks [17]. Ideally, a mesh is watertight; i.e., the mesh consists of one whole surface with no disconnected edges. Unfortunately, this is not the case for all existing CAE models and standard benchmark data sets [18]. Further, learning from mesh structure needs necessary pre-computation of the neighboring vertex correspondence information. Also, adapting a mesh to multiple geometries is a challenging task due to the lack of correspondence between the underlying mesh structures of different geometries. Hence, learning 3D meshes is a challenging task because it needs expensive pre-processing and quality checking on the 3D mesh geometries.

Voxels and Octrees: Voxel grids represent an object as a uniform 3D grid, where a non-zero value is assigned to the cells that belong to the occupied and non-occupied parts

of the shape. Therefore, learning from the voxel representation involves learning the volumetric grids, which partition the input space into regular cubes. This introduces extra computation costs due to the sparsity of high-dimensional 3D data and requires enormous memory storage. Additionally, considering a low-voxel resolution hides fine details of the input geometries, which is also currently an issue for applicability of the voxels in combination with DL methods that rely on fine-grained details of the input geometries. Although encouraging progress has been achieved using voxel-based representations for object detection [19] and style transfer [20]. However, voxel-based methods are unable to scale well to dense 3D data, since the computation and memory requirements grow cubically with higher resolution.

To overcome the insufficient memory usage of the voxel representation, octree graphs are introduced for the storage of voxel information [21, 22]. Octree is a hierarchical graph-based representation, which is formed by recursively partitioning the entire 3D space into sub-spaces, known as octant. The disadvantage of this representation is the need for expensive neighborhood data-preparation to hold information about the octant belonging to different leaves. However, with voxel or octree representation, an object or a scene can only be approximated, but not fully represented [23]. This is because a voxel or an octree decomposes 3D objects into smaller blocks, but 3D properties like surface curvature are lost.

Point-Clouds: A point cloud is an unordered list of points that approximate 3D geometries and each point in the point cloud is represented by three coordinates (x, y, z) in a Cartesian coordinate system. Point clouds are a sparse, memory-efficient data format used in several DL applications like classification [24], object recognition [25, 19], segmentation [26, 27], and generative tasks in autonomous driving community [28, 29]. However, for direct application of point clouds in engineering domains, it needs a post-processing step to convert point clouds to polygonal meshes [30]. Nevertheless, the point cloud representation is easily scalable and preserves the original geometric information in 3D space without any discretization. Further, 3D point clouds can be sampled from the CAE models, where the user has better control of defining the number of sampling points, which makes the point cloud representation one of the popular choices for DL research.

2.2 Benchmark Dataset

A key assumption for training a deep neural network model is the availability of an existing dataset of 3D shapes for training the network, regardless of the type of 3D representations processed by the DL model. A dataset consists of a set of 3D objects that are available on the internet with free or commercial licenses. Public datasets consist of different objects

like cars, airplanes, chairs etc. and are typically stored in a triangular mesh format. In the context of 3D supervised and unsupervised learning, two 3D datasets have become popular for scientific research. The first is the ModelNet [31] curated by Princeton University and the second is the ShapeNet [18] curated by Princeton University and Stanford University. The 3D shapes in these datasets often originate from online object repositories. However, some shapes are manually selected, tagged, and reviewed by crowd sourcing or by the researcher involved. These refined 3D shapes are typically published in subsets of ModelNet and ShapeNet, named as ModelNet40 and ShapeNetCore.

ModelNet40 [31] is a data set of 3D CAD models of the most common object categories in the world, consisting of over 12,311 pre-aligned shapes from 40 categories. The CAD models are stored in object file format (OFF) and have been extensively used in shape retrieval and classification tasks [31]. ShapeNetCore [18] is a subset of the ShapeNet dataset with clean 3D models that are manually verified. The 3D shapes in this dataset are aligned, scaled, and stored as *.obj* format with the class annotation and original dimensions. It consists of 51,300 unique 3D shapes that cover over 55 common object categories and are used extensively in research related to shape generative task, classification and segmentation. Since the research in this dissertation focuses on automotive designs, we rely on the shapes from the car class of the ShapeNetCore dataset.

2.3 Geometric Deep Learning

The essence of DL focuses on extracting hierarchical features of the input data using neural network models and training the models with local-gradient descent, known as back-propagation. Within DL, geometric deep learning (GDL) [32] is a subclass which emphasizes learning on 3D geometric representations. GDL aims to bring geometric unification to DL, where commonly used neural network architectures, such as convolutional (CNNs), graph (GNN), and recurrent neural networks (RNNs) are used for learning non-Euclidean 3D data. However, the performance of any DL model is heavily dependent on the choice of representation of the 3D geometry on which it is applied [15]. Therefore, feature engineering of the input data is important prior to training a DL model, even though it is a labor-intensive task.

With recent advancements in computational power, a novel technique known as representation learning (RL) [15] has been introduced in machine learning community. RL focuses on extracting features automatically and learning patterns from the input data for classification and prediction tasks. Popular deep neural-network architectures for unsupervised RL learning are restricted Boltzmann machines (RBMs) [33], deep belief networks (DBNs)[34, 35], and autoencoders (AEs) [36].

The RBM [33] is a shallow neural network to learn patterns in the input data. RBM architecture consists of input and hidden nodes interconnected with each other for information sharing. The network does not generate any outputs since there is no output node in this model. A DBM is a stacked layer of RBM, which has hidden layers to compute the probability distribution of the input data. However, RBMs and DBMs are less frequently used compared to AEs, which are probably the most versatile unsupervised learning models [37]. The AE network has a bottleneck layer (latent representation) and it learns the data representation with the aim of recovering maximum information of the original input. Training the AE network by minimizing reconstruction objective, enforces to retain all invariant feature information in the encoded latent representations.

Similar to the latent variable models like AE, another popular unsupervised representation learning model is a generative model [38], which is the popular choice in many computer-vision applications in the absence of labeled data. Generative models are probabilistic models, which capture the underlying explanatory factors of the input data in a posterior distribution and make it possible to generate new samples that seem to belong to the original data distribution. Due to their capability to generate new examples, generative models have been used for numerous successful applications, such as data augmentation[39], generation realistic images[40], and RL [41].

2.3.1 Generative Models

The goal of the generative model is to learn the probability density function $p(x)$, which describes the behavior of the input 3D data x. Modeling 3D data in a generative way offers a distinctive advantage, as it allows sampling novel 3D geometries from the probabilistic latent space, which are either explained implicitly or explicitly using a Gaussian or uniform distribution. The implicit models do not compute $p(x)$ but learn a mapping that can be used to generate samples by transforming a simple random variable [42–44], and explicit models define a density function of the distribution or approximate it [45, 46]. Implicit models, like Generative Adversarial Networks (GANs), need to only specify a stochastic procedure to generate data [38]. While explicit models such as, Hidden Markov Models (HMMs) [47], Belief Nets [34] and Variational Autoencoders [48], can either compute the exact density function or approximate using latent variable models.

Latent Variable Models

A latent variable (LV) model is a statistical model that maps a set of input variables x to a set of latent variables z and is trained by maximizing the log-likelihood estimation (MLE) [49] of the training data x, via a neural network model parameterized by θ. The

MLE is represented as,

$$\begin{aligned} \theta^{ML} &= \arg\max_{\theta} \sum_{j=1}^{N} \log p_\theta(x_j) \\ \text{where } p_\theta(x) &= \int p_\theta(x|z)p(z)dz \end{aligned} \tag{2.1}$$

The MLE tries to map the probability distribution $p(x)$ into latent variables z that follow a prior distribution $p(z)$. In this general framework, the prior $p(z)$ is usually a standard Gaussian distribution, and the likelihood $p_\theta(x|z)$ is a distribution, whose parameters are computed using a neural network, known as generators. The complexity of this generative neural network and intractability of the log-likelihood (Eq. 2.1) motivates the introduction of two main generative models: Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs).

Variational Autoencoders (VAEs)

VAEs [48, 41, 50] are generative models, built on the principle of latent variable models. The difference in the VAE from neural network-based autoencoder is that the VAE estimates the latent variable from a probabilistic perspective. The VAE assumes a joint distribution over the latent variables and approximates the intractable posterior density over the latent variables with variational inference, using an inference network. The VAE architecture consist of a probabilistic encoder (e_θ) and a decoder (d_ϕ), where θ and ϕ are the neural network parameters of the encoder and decoder respectively. The loss function depends on individual data points of the input x. Therefore, the total loss is summation over all N data points, i.e., $\sum_{j=1}^{N} l_j$.

$$l_j(\theta, \phi, x_j) = \underbrace{-\mathbb{E}_{z\sim e_\theta(z|x)} \log p_\theta(x|z)}_{\mathcal{L}_{recon}} + \underbrace{\frac{1}{2}\sum_{k=1}^{L_z}(1 + \log \sigma_k^2(x) - \mu_k^2(x) - \sigma_k^2(x))}_{\mathcal{L}_{KL}} \tag{2.2}$$

where μ_k and σ_k are the k-th component of output vector $\mu(x)$ and $\sigma(x)$. The first term is the reconstruction loss ($\mathcal{L}_{recon}$) or the expected negative log-likelihood of the j-th data point, and the second term denotes the Kullback-Leibler (KL) divergence loss function ($\mathcal{L}_{KL}$). The VAE is trained by balancing these two components of the loss functions, where KL-divergence acts as a regularizer of the latent space.

Generative Adversarial Networks (GANs)

GANs [38, 28, 42, 40] follow an opposite view of VAEs and do not optimize the log-likelihood of the data. Instead GANs rely on adversarial training, using a generator and

a discriminator. The generator aims to synthesize samples that look indistinguishable from the input data by passing a randomly drawn sample from a simple distribution $p(z)$ through a generator function (G). Next, the discriminator (D) distinguish synthesized samples from real samples. The generator loss $\mathcal{L}_G$ and the discriminator loss $\mathcal{L}_D$ adopt opposite objectives (Eq. 2.3), which are represented as,

$$\min_G \max_D = \underbrace{\mathbb{E}_{x\sim p(x)}[logD(x)]}_{\mathcal{L}_D} + \underbrace{\mathbb{E}_{z\sim p(z)}[log(1-D(G(z)))]}_{\mathcal{L}_G} \tag{2.3}$$

Even though GANs provide a good generative framework to produce realistic samples, they suffer from non-convergence, mode-collapse and vanishing gradients [51]. Many researchers have been devoted to this problem theoretically [42, 52], or empirically [44, 53]. Consequently, researchers use various forms of regularizations to improve the training convergence of a GAN [54, 55], but still the hyper-parameter tuning remains challenging. Additionally, GANs do not explicitly describe a probability distribution over data and hence are difficult to evaluate. Therefore, in this research, a VAE is used as a generative model for learning on point cloud representations of 3D CAE models.

2.3.2 Generative Models for Learning 3D Representations

Most of the generative models for learning on point-cloud representations follow the PointNet architecture [24], which address the invariance of unorganized point clouds with respect to the ordering of the points in point cloud data. Both the PointNet [24] and PointNet++ [56] addresses the permutation invariance of points in point clouds by handling the input data with point-wise operators followed by a global operator, e.g., max-pooling. Even though such a network is not a generative model, PointNet has been effective in several tasks, such as semantic segmentation [57, 58] and object classification tasks [59]. Therefore, many DL models adapt the architecture of the PointNet to learn from unstructured point clouds for its simplicity and strong representation capability.

The popular autoencoder architecture proposed by Achlioptas et al. [60], is built on the PointNet, where the encoder is like the PointNet and the decoder includes fully connected layers. The encoder is comprised of five 1D convolutional layers followed by a max-pooling layer, which aggregates the learned features into a latent representation. The shapes are reconstructed by passing the samples from the latent space through the decoder, which predicts the Cartesian coordinates of the points. The autoencoder architecture is trained on point cloud representations of ShapeNetCore [18], and the author trained an additional Gaussian mixture model (GMM) on the latent space of AE (AE+GMM) to overcome the limited generative capabilities of the AE. Hence, the network outperforms co-existing models in shape-generative and classification tasks. Other research [61] on autoencoder trained on point cloud representations shows that transferring features using

an AE between two different geometric shapes results in a fuzzy point cloud, which may be a consequence of the irregularity in the latent space of the AE in the absence of any regularization technique.

Variational autoencoders (VAEs) are probabilistic versions of the autoencoder architecture, which address the issue of non-regularized latent space by enforcing additional constraints on the distribution of the latent variables. The original VAE [48] is trained with the combination of reconstruction error and KL-divergence and achieved limited disentanglement performance on the MNIST dataset. To improve latent variable disentanglement, Gonen et al. [62] proposed β-VAE, where the coefficient β regulates the strength of the KL divergence, and hence, imposes a limit on the capacity of the encoded latent information. With this modification, VAE has been successfully used to learn geometric data for various tasks, such as 3D shape reconstruction from 2D images [63], 3D shape segmentation [64], shape generation from segmented objects [65], as well as data-augmentation [66–68]. Most of these works trained the VAE on manually labeled segmented 3D shapes or on 2D images of 3D shapes. Further, VAEs introduced in these works are not evaluated for engineering design optimizations; i.e., the VAEs ability to generate realistic, novel 3D shapes.

The two networks closest to our goal of generating novel and diverse shapes are CompoNet [69] and adversarial autoencoder (AAE) [70]. CompoNet is a generative neural network for 2D or 3D shapes based on part-based prior, which relies on a VAE for 2D shape synthesis. However, for 3D shape generation, Schor et al. [69] used the above-mentioned AE+GMM. Zamorski et al. [70] proposed an end-to-end solution to generate 3D shapes with an adversarial autoencoder (AAE) for 3D point clouds using binary representation in the latent space. The AAE differs from a VAE in the loss computed on the latent space, where the AAE uses the adversarial loss like a GAN, and the VAE uses the KL divergence to enforce regularization of the latent space. Zamorski et al. [70] trained the adversarial model with the Earth-Mover distance (EMD) and analyzed the model with various objectives, such as 3D point cloud clustering and object retrieval. Hence, a VAE tailored to engineering design optimization is still missing in the literature.

2.4 Conclusion

This chapter presents a survey of geometric representations and deep learning models for generating a common design space for automotive optimizations. The main objective is to answer $RQ1$, which focuses on analyzing the suitable 3D representations for training a DL model.

RQ 1. Which 3D representation is suitable for efficient training of deep learning models?

Existing CAE models hold handcrafted features by the designer, providing an effective way to learn from the models. Therefore, to exploit the notion of experience hidden

in 3D CAE models collected over past product design cycles, 3D point clouds provide the best trade-off between the quality of the representations and the computational effort for training a DL model. Additionally, the point-cloud representations are easily scalable to higher dimensions to capture more details of the 3D shapes.

With respect to the type of 3D representation selection and based on the presented review on the state-of-the-art deep learning models, the variational autoencoder (VAE) proves to be the most suitable generative architecture for three main reasons. First, a VAE learns the existing 3D data distribution using unsupervised learning. Second, the network creates a compact low-dimensional solution space for search and optimization of 3D designs. Lastly, the VAE trained on 3D point cloud representations provides the flexibility to generate new designs from the probabilistic latent space, which is key for using a VAE in creative design exploration tasks.

Compared to prior introduced VAEs in the literature that are utilized for different applications, this research focuses on introducing a point cloud variational autoencoder (PC-VAE) tailored for engineering design optimization tasks. Hence, the PC-VAE is used to learn point cloud representations of car shapes available in the ShapeNetCore repository [18]. In the following chapters, the architecture of the PC-VAE is detailed along with a set of experiments to evaluate the performance of the model for multi-criteria-based design optimization tasks.

Chapter 3

Generative Variational Autoencoder for Learning on 3D Point Clouds

Based on the reviewed literature in Chapter 2 and in the context of our envision design optimization framework (Fig. 1.4), the point cloud variational autoencoder (PC-VAE) indicates a promising data-driven approach for learning on 3D CAE models and generating a low-dimensional representations of these designs. Therefore in this chapter, the architecture of the utilized PC-VAE is discussed in detail, along with a series of experiments to verify the shape-generative performance of the model for engineering applications. The objective of this chapter is to answer $RQ2$ by modifying the existing autoencoder architecture [60] to the PC-VAE and utilizing a series of methods to quantify the novelty and diversity of the 3D shapes generated using the PC-VAE.

Following the statistical background of the variational autoencoder (Section 2.3.1), the PC-VAE consists of an encoder that maps an input 3D shape x_{in} into a probabilistic low-dimensional latent representation $\vec{z}$ and a decoder that reconstruct 3D space from $\vec{z}$. The neural network parameters of the encoder and decoder are learned concurrently during training the PC-VAE with a combination of loss functions. Through this training, the PC-VAE creates an explicit distribution in the latent space, instead of exactly learning a simple one-to-one mapping of an input CAE model to a latent representation. Further, this probabilistic latent space allows sampling of new designs, which helps the PC-VAE in shape-generative tasks.

The remainder of this chapter is outlined as follows: In Section 3.1, a literature review is presented focusing on the metrics to evaluate the quality of the 3D shapes generated by VAEs. Section 3.2 describes the architecture of the proposed PC-VAE and the shape generation techniques. Also, quantitative metrics are detailed in this section, which are used for evaluating the reconstruction and generative capability of the PC-VAE. In Section 3.3, the details of the network training are presented, followed by a series of experiments to compare the reconstruction quality of the PC-VAE with other models from the literature

(Section 3.4). Next, the PC-VAE is utilized for generating 3D shapes, and the qualities of the generated shapes are evaluated for engineering applications based on pre-defined metrics (Section 3.5). Lastly, the PC-VAE is exploited to generate unseen shapes to verify the shape generative capability of the model (Section 3.6). Finally, the chapter is concluded in Section 3.7, with a summary and outlook.

3.1 Prior Art

As discussed in the review of generative model (Section 2.3.1), a PC-VAE tailored to engineering design optimization is introduce in this chapter. The aim of introducing the PC-VAE is to generate a continuous latent space, which helps in generating novel realistic car designs, fitted for engineering applications. Prior research on generative models introduced methods for 3D shape generation from latent representations and proposed metrics to evaluate the quality of the generated shapes [60, 71, 69], which are discussed below.

Evaluation of 3D Shape Generation

There are two procedures to generate shapes from a trained generative model. The first procedure is to utilize linear interpolation between two shapes in the latent manifold. Research on generative models typically uses this procedure as a way of demonstrating that a generative model has not simply memorized the training examples [72]. The second procedure involves random sampling on the latent manifold to evaluate the explorative capability of the generative models.

Generating realistic and unseen novel shapes is central to the intended application domain of engineering design. In other words, a model should generate plausible realistic shapes, i.e., shapes that in some aspects resemble the samples of the data set, yet should be diverse, i.e., shapes should be sufficiently different from the training samples that are used to train the PC-VAE. To quantify the *realism* of a generated shape, Berthelot et al. [71] proposed a mean distance (MD) metric for 2D shapes that compares the minimum cosine distance of the interpolated data points with the original data points by finding the nearest neighbor of each interpolation step in the data set. Achlioptas et al.[60] proposed a similar metric named minimum matching distance (MMD) metric to measure the closeness of two-point cloud sets using the Chamfer distance (CD), where closeness between sets is assumed to indicate a higher level of *realism*. To evaluate the *diversity* (or *novelty*) of generated shapes, Schor et al. [69] rely on a classifier to evaluate the diversity of samples produced by a generative model. The generative diversity of the model is calculated using the latent representations and is estimated as the percentage of the generated shapes classified as belonging to an unseen test set.

In sum, quantitative approaches for evaluating both realism and diversity of 3D shapes generated from (variational) autoencoders have been proposed. Both approaches were combined to assess whether the proposed PC-VAE can generate shapes that are both realistic and novel. These criteria for evaluating the PC-VAE based on its ability to generate realistic and novel shapes have been little explored, despite the relative simplicity and higher efficiency of VAEs in comparison to other (3D) generative models.

Additionally, utilizing latent space sampling for generating new shapes adds randomness in selecting shapes from unexplored regions in the latent space. To overcome randomness and generate shapes with specific properties, prior research [73] used a generative model for mapping complex input space into a latent space and conducted optimization to achieve a balance between desirable solution accuracy and convergence rate. This optimization approach in the low-dimensional latent space has been utilized for generation of chemical designs with desired properties [74]. Therefore, to achieve novel shapes using the PC-VAE, an optimization method is proposed to enforce the generation of unseen diverse shapes from unexplored regions in the latent manifold.

3.2 Methodology

3.2.1 3D Point Cloud Variational Autoencoder Architecture

The implementation of the PC-VAE (Fig. 3.1) utilized in the present research follows the mathematical background of a variational autoencoder model described in the previous chapter (Section 2.3.1). The encoder-decoder structure used in PC-VAE corresponds to the architecture proposed in [60], only the last layer of the decoder is replaced with a sigmoid activation function [61], since the coordinates of all points are normalized to the range $[0.1, 0.9]$. The PC-VAE's architecture includes an encoder, followed by a max-pooling layer, which generates two separate layers: a mean (μ) layer and a standard deviation (σ) layer. The bottleneck latent layer z is sampled from the μ and σ layers. After the latent layer, a decoder made of three fully connected layers recovers the Cartesian coordinates of the points based on the values of the latent variables. In the PC-VAE architecture, the encoder is referred to as the recognition model, and the decoder is referred to as the generative model.

Encoder

The encoder comprises five 1D-convolutional layers, where the first four layers are activated by rectified linear units (ReLU) and a batch normalization (BN) layer [75], and the fifth layer by a hyperbolic tangent function (*tanh*). The advantage of utilizing 1D convolutions is that the operator requires fewer parameters than fully connected layers, and it processes

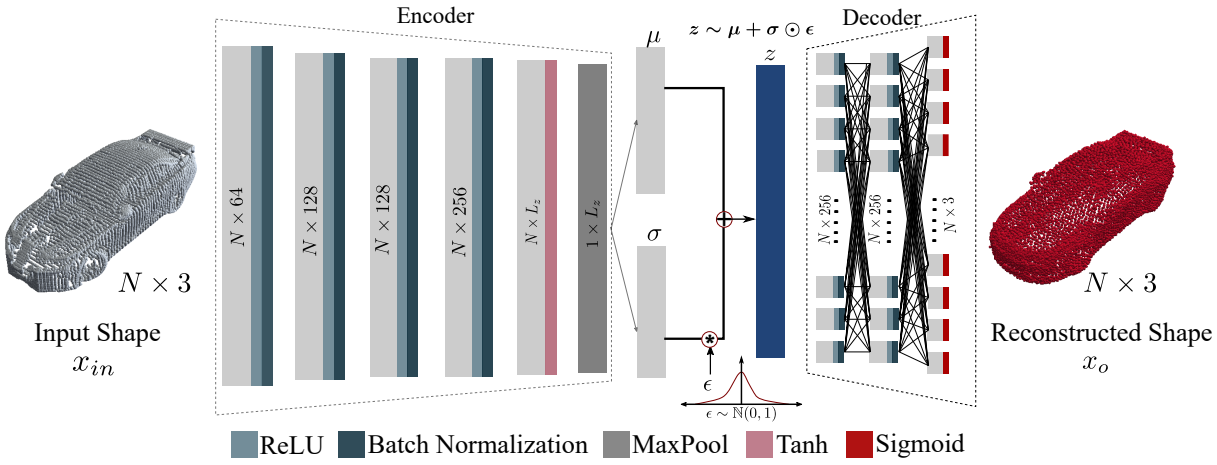


Fig. 3.1 PC-VAE Architecture. Input and output are represented by a 3D point cloud composed of N points.

each point in the input point cloud individually, regardless of the ordering of the points. In an N-dimensional input point cloud, the convolution is applied to one point at a time and yields a feature metric $\mathcal{F}^j$ of dimension $[N, f_j]$ at the j-th layer of the encoder, where f_j corresponds to the number of features of each layers. This feature metric is passed through an activation layer, generating the output metric $\mathcal{F}_a^j$. The batch normalization (BN) in each layer normalizes the output of each activation layer $\mathcal{F}_a^j$ using mean μ_{bn} and variance σ_{bn}^2 values calculated over each batch (Eq. 3.1),

$$\mu_{bn} = \frac{1}{N}\sum_j \mathcal{F}_a^j \ , \ \sigma_{bn}^2 = \frac{1}{N}\sum_j (\mathcal{F}_a^j - \mu)^2$$
$$\mathcal{F}_{norm}^j = \gamma * \frac{(\mathcal{F}_a^j - \mu)}{\sqrt{\sigma^2}} + \delta \tag{3.1}$$

where, $\mathcal{F}_{norm}^j$ is the output of the BN layer. The μ_{bn} and σ_{bn} are estimated within a batch of samples for each dimension independently and γ and δ are the scale and shift parameters. The output of the fifth convolutional layer in the encoder is passed to a max-pooling layer that produces a vector of dimension L_z that forms the bottleneck for two separate L_z-D vectors: a mean vector $\vec{\mu}$ and a standard deviation vector $\vec{\sigma}$. The L_z-dimensional vector after the max-pooling layer corresponds to the maximum activation of each output neuron for a point cloud, and therefore, contains global shape information and is invariant with respect to the ordering of the points. The mean layer $\vec{\mu}$ has no activation function, while the deviation layer $\vec{\sigma}$ uses a sigmoid activation function.

Latent Space and Decoder

Following the μ and σ layers, a vector $\vec{z}$ of dimension L_z sampled from a multi-variate Gaussian with mean $\vec{\mu}$ and standard deviation $\vec{\sigma}$ (Eq. 3.2), where $\vec{\mu}$ and $\vec{\sigma}$ are mean and variances vectors respectively, and x_{in} is the $N \times 3$ input point cloud.

$$\vec{z} \approx \vec{\mu} + \vec{\sigma} \odot \epsilon \tag{3.2}$$

From the latent space, a decoder with three fully connected layers generates 3D point clouds by predicting the Cartesian coordinates of the points. The first two layers are activated by ReLUs and the third layer by a sigmoid function. Hence, the reconstructed point cloud data at the output of the decoder network has normalized coordinates $(x, y, z) \in [0, 1]^3$.

The weights of the neural networks that represent the encoder and decoder are initialized with a value drawn from a zero centered Gaussian with a standard deviation 1.0E−01.

3.2.2 Loss Functions

For training the PC-VAE, a combination of two loss functions (Eq. 3.3) is formulated, following the background of VAE in the previous chapter (Eq. 2.2). Two hyper-parameters α and β are introduced to balance these two loss functions.

$$\mathcal{L}_{VAE}\left(x_{in}; \theta, \phi\right) = \alpha\, \mathcal{L}_{recon} + \beta\, \mathcal{L}_{KL} \tag{3.3}$$

The first term is the reconstruction loss $\mathcal{L}_{recon}$, which measures the difference between the input and reconstructed point cloud. Prior research [60, 61] commonly used Chamfer distance (CD) to measure the reconstruction error between point clouds and the equation to calculate CD is given by:

$$\mathcal{L}_{recon} = CD(x_{in}, x_o) = \sum_{p_i \in x_{in}} \min_{p_o \in x_o} \|p_i - p_o\|_2^2 + \sum_{p_o \in x_o} \min_{p_i \in x_{in}} \|p_i - p_o\|_2^2 \tag{3.4}$$

where $x_{in}, x_o \subset \mathbb{R}^3$ are the input and output point clouds respectively. The second term is the KL-divergence loss ($\mathcal{L}_{KL}$) that measures the difference between the learned latent distribution and an assumed prior normal distribution $p(\vec{z}) = \mathcal{N}(0, 1)$. The KL divergence is implemented following the Eq. 3.5.

$$\mathcal{L}_{KL} = \frac{1}{2} \sum_{k=1}^{L_z} (1 + \log \sigma_k^2(x) - \mu_k^2(x) - \sigma_k^2(x)) \tag{3.5}$$

First, an input shape $x_{in} \subset \mathbb{R}^3$ is passed through an encoder, which is parameterized by θ to obtain a latent code $\vec{z} \approx e_\theta(x_{in})$. The L_z-D latent vector $\vec{z}$ is then passed

through a decoder, which is parameterized by ϕ to generate an approximate reconstruction $x_o = d_\psi(\vec{z}) \subset \mathbb{R}^3$ of the input x_{in}. The encoder and decoder are multi-layer neural networks that are trained simultaneously with respect to ϕ and θ to minimize the combination of loss functions in Eq. 3.3. For training and exploring the generative capabilities of the PC-VAE, the point clouds of the data set $\mathcal{S}$ are divided into a seen training set, $\mathcal{D}_{train}$, and an unseen test set, $\mathcal{D}_{test}$. Algorithm 1 shows the training process of the PC-VAE with the point clouds from $\mathcal{D}_{train}$. At each iteration, based on a pre-selected batch-size b, $\mathcal{D}_b$ number of samples from the $\mathcal{D}_{train}$ is utilized for training the network.

Algorithm 1 PC-VAE training

1: Initialize ϕ and θ
2: **for** $i = 1, 2, ...$until convergence **do**
3: Sample a mini-batch $\mathcal{D}_b$ from $\mathcal{D}_{train}$
4: $\vec{\mu}$, $\log \vec{\sigma^2} = e_\theta(x_{in})$
5: $\vec{z} \sim \mathcal{N}(\vec{\mu}, \vec{\sigma^2})$ and reconstruct $x_o \sim e_\theta(\vec{z})$
6: Computer gradient $g_{\theta,\phi} \leftarrow \bigtriangledown_{\theta,\phi}\mathcal{L}_{\theta,\phi}(x_{in}; \theta, \phi)$
7: Update θ, ϕ using $g_{\theta,\phi}$
8: **end for**

In the next section, the metrics are defined that are used to evaluate the reconstruction and shape-generative performance of the PC-VAE.

3.2.3 Shape Generation Techniques

The decoder (Fig. 3.1) performs as the generative model. To generate shapes from a trained model, research on generative models typically uses two approaches: First, linear interpolation between two shapes as a way of demonstrating that a generative model has not simply memorized the training examples [72]. Second, to limit the generation of feasible shapes, instead of random sampling latent variables from a normal distribution, new latent vectors are sampled from the encoding distributions of the shapes from the training set.

Interpolation between 3D Designs: Interpolating between an initial design x_{in} and a target design x_{target} on the latent space of the PC-VAE generates a mixture of two latent codes, which are reconstructed back into 3D space using the decoder p_θ. The latent codes are combined to generate intermediate designs $x_i^{'}$, where $x_i^{'} = d_\phi(\lambda\vec{z}_{in} + (1-\lambda)\vec{z}_{target})$. The interpolating factor λ varies between $[0, 1]$ and $\vec{z}_{in} = e_\theta(x_{in})$ and $\vec{z}_{target} = e_\theta(x_{target})$ are the latent codes corresponding to the reference shapes x_{in} and x_{target}, respectively.

Sampling Designs: For a given input shape x_{in} the encoder e_θ output two vectors ($\vec{\mu}$ and $\vec{\sigma}$) of dimension $[1 \times L_z]$. Each of the latent variables of $\vec{z}$ is sampled from a Gaussian

distribution using Eq. 3.2. All the sampled latent variables of the input shape x_{in} are combined to form the latent vector $\vec{z}$ of dimension $[1 \times L_z]$.

Each random sampling (Eq. 3.2) generates a slight variation of latent vector $\vec{z}$. For example, for a given input shape x_{in} (Fig. 3.2), sampling the latent distributions twice, generates two slightly different latent vectors $\vec{z}_1$ and $\vec{z}_2$. The decoder network is able to accurately reconstruct the input (x_{in}) into x_{o_1} and x_{o_2}, where there is a slight variation between x_{o_1} and x_{o_2}. Hence, sampling the latent space generates slightly different geometries at each turn.

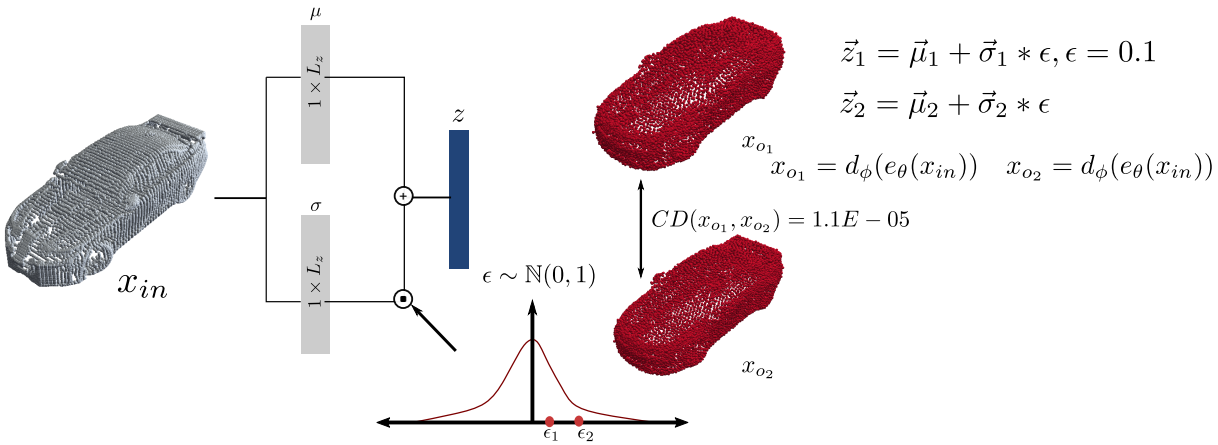


Fig. 3.2 For a given shape x_{in}, two latent vector $\vec{z}_1$ and $\vec{z}_2$ are sampled from the latent distributions. x_{o_1} and x_{o_2} are the reconstructed shapes generated by decoding the latent codes $\vec{z}_1$ and $\vec{z}_2$, respectively. The difference between the reconstructed shapes x_{o_1} and x_{o_2} are given by CD(x_{o_1},x_{o_2}).

The shapes generated through interpolation or sampling the latent space are represented as set $\mathcal{G}$. Next, to evaluate the quality of the generated shapes in $\mathcal{G}$, quantitative metrics are introduced in the next section.

3.2.4 Metrics for Evaluating the PC-VAE

An important component of this research is to compare the degree to which point clouds, synthesized or reconstructed, represent the shapes in the data set $\mathcal{S}$. The realism of the generated shapes in $\mathcal{G}$ is evaluated qualitatively through visual inspection and quantitatively by measuring the closeness of the generated shapes to the shapes in the data set $\mathcal{S}$ [71].

As a second measure to evaluate, the novelty or diversity of the generated shapes in $\mathcal{G}$ is measured by quantifying their variations from shapes in $\mathcal{D}_{train}$, following the approach in [69]. Therefore, the quality of the new shapes generated with PC-VAE has two objectives: create shapes close enough to the data set to be considered realistic, but far enough from the training set to be considered novel.

Minimum Matching Distance using Chamfer Distance (MMD-CD): To quantify the realism or closeness of generated shapes in $\mathcal{G}$ with respect to existing shapes in the

data set $\mathcal{S}$, the mean distance (MMD) [60] is used. MMD-CD is calculated as the average minimum CD between the shapes at $\mathcal{G}$ and their respective neighbor at $\mathcal{S}$. Therefore, MMD measures the similarity between point clouds in $\mathcal{G}$ to those in $\mathcal{S}$.

Generative Diversity: The measure of diversity presented in [69] is utilized to quantify the diversity of the generated shapes, $\mathcal{G}$. Similar to the approach outlined in [69], a binary classifier is trained on the latent representations of the whole data set $\mathcal{S}$, where the information whether a sample belonged to either the training set, $\mathcal{D}_{train}$, or test set, $\mathcal{D}_{test}$, was used as the label. The diversity is then calculated as the percentage of the generated samples, $\mathcal{G}$, that are classified as belonging to the test set. In other words, the classifier quantifies the percentage of samples generated by the model that are closer to the unseen test set $\mathcal{D}_{test}$ than to the training set $\mathcal{D}_{train}$. The higher the number of generated samples classified into $\mathcal{D}_{test}$, the higher the diversity of the model. A multi-layer perceptron (MLP) is used as a classifier, where the output layer of the MLP consists of 2 units to predict the labels of the given latent representation.

Coverage: The novelty of the generated shapes in $\mathcal{G}$ is measured with coverage, which quantifies the fraction of point clouds in $\mathcal{G}$ that can be matched to the test set, $\mathcal{D}_{test}$. For each point-cloud in $\mathcal{G}$, the nearest neighbor in $\mathcal{D}_{test}$ is computed with the CD. The coverage is estimated as the percentage of shapes in $\mathcal{D}_{test}$ that are nearest neighbors to $\mathcal{G}$. Small coverage signifies the generated shapes resemble only a small fraction of $\mathcal{D}_{test}$, and higher coverage indicates the generated samples represent a large fraction of $\mathcal{D}_{test}$.

3.3 Network Training and Shape Reconstruction Evaluation

3.3.1 Data Set

For different experiments in this thesis, geometric models are sampled from the ShapeNetCore repository [18]. The positions of the geometric models are adjusted to fit the models in the space $[0, 1]^3$. The orientation of the geometries is also changed such that the direction of the height is aligned to the global z-axis, and the x-axis represents the length of the models (Fig. 3.3).

The data set $\mathcal{S}$ comprises 3D shapes samples from the car class of the ShapeNetCore repository. Each 3D shape consists of $N = 2048$ points, and the points are sampled directly from the vertices of the pre-processed meshes of the repository, assuming a random uniform probability (Fig. 3.4). For training the network, the data set is split into 80% training ($\mathcal{D}_{train}$) and 20% test set ($\mathcal{D}_{test}$). For hyper-parameter tuning of the network such

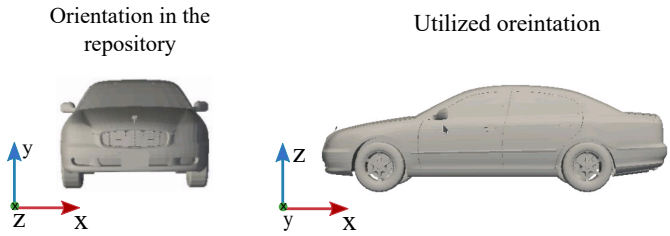


Fig. 3.3 Example of a car geometry from the ShapeNetCore repository with the original orientation and utilized orientation in the experiments.

as learning rate, batch size and epochs, a 20% of the training set is used as the validation set in the initial run. After parameters are finalized, the shapes in the validation set are added back to the training set.

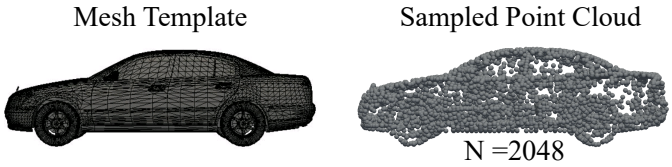


Fig. 3.4 Template mesh and corresponding 3D point cloud sampled using random sampling of N points from the template mesh, here $N = 2048$.

Hardware Settings

Unless specified otherwise, all the architectures in this dissertation are implemented using Python with TensorFlow® for computation on Graphics Processing Units (GPUs). The networks were trained on a single NVIDIA RTX 2080 Ti GPU.

3.3.2 Training a Baseline Model - Point Cloud Autoencoder

Since the proposed PC-VAE is based on the architecture of point cloud autoencoder (PC-AE) [60, 61], this model was chosen as the baseline model for all the experiments. The PC-AE was trained with latent dimensions of $L_z = 128$ using the Adam optimizer [76] and a learning rate of 5E−04. The batch size is set to 50 and each time the network is trained for 750 epochs.

3.3.3 Training the PC-VAE

The proposed PC-VAE architecture (Fig. 3.1) was trained with latent dimension $L_z = 128$, using the Adam optimizer and a learning rate of 5E−03. The network is trained with a batch size of 50 and trained for 750 epochs.

To optimize the hyper-parameters α and β in the loss function (Eq. 3.3), a grid search is performed to determine the values of α and β. Since the scale of the reconstruction loss

and KL-divergence are different, the PC-VAE is trained several times with a list of values of α and β to determine the optimal α and β.

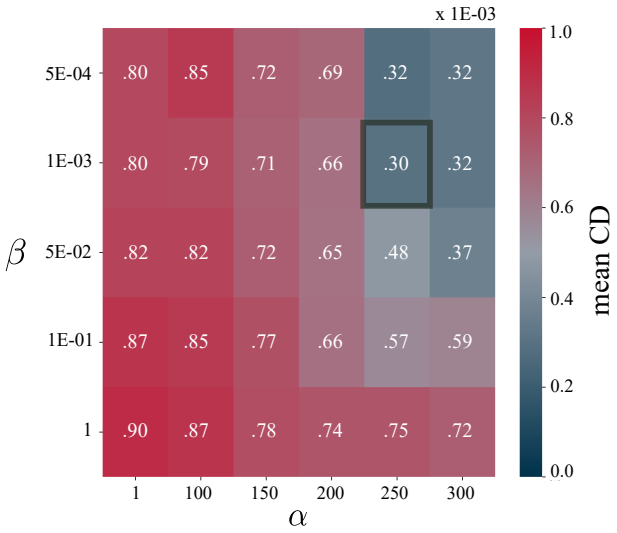


Fig. 3.5 The heat-map represents mean reconstruction error (CD) of the validation set shapes, when trained with different combinations of α and β. The combination of optimal α and β generating the least CD is marked in black square.

The values used for the grid search are, $\alpha = (1, 100, 150, 200, 250, 300)$ and $\beta = (1, 1\text{E}{-}01, 5\text{E}{-}02, 5\text{E}{-}03, 5\text{E}{-}04)$. Since training the network for each combination of α and β needs almost 6-hours, the grid search is performed for the selected set of α and β. Fig. 3.5, shows the mean values of the reconstruction error on the validation set for different combinations of hyper-parameters α and β. For $\alpha = 250$, and $\beta = 1\text{E}{-}03$, the average of the reconstruction error (CD) of the shapes in the validation set is minimum. Therefore, comparing the trade-off between the reconstruction accuracy and divergence of the latent space, the optimal values of the parameters are set to $\alpha = 250$ and $\beta = 1\text{E}{-}03$ for training the PC-VAE.

3.4 Evaluation of the Reconstruction Performance

To validate the implementation of the proposed PC-VAE, the first experiment is designed to compare the reconstruction quality of the PC-VAE with existing models as reported in the literature, specifically a regular AE [60] and a 3D-Adversarial AE (3dAAE-G) [70]. Since both prior works were not based on 3D car shapes, the PC-VAE is trained on the chair class of the ShapeNet data set with 90% of the chair shape and tested the performance of the model using MMD-CD on 10% of the chair shapes. Both the AE and PC-VAE were trained with CD as the reconstruction loss. But for 3dAAE-G [70], the model was trained using Earth-Mover distance (EMD) as a reconstruction loss.

As a second experiment to evaluate the reconstruction quality of the PC-VAE with respect to varied sizes of the latent representation, the PC-VAE is trained considering similar conditions but varying the latent representation size to $L_z = (2, 5, 10, 20, 32, 64, 128, 256)$.

Result and Discussion

The performance of the PC-VAE is compared with a regular AE [60] and a 3dAAE-G [70] by calculating the MMD-CD measure on the test set. The MMD-CD measure is utilized to estimate the fidelity in terms of distance measures of the reconstructed test set shapes with respect to the ground truth shapes in the test set. The MMD-CD between the reconstructed point clouds and their corresponding ground truth in the test data set of the chair object class is shown in Table 3.1. The resulting MMD-CD of PC-VAE is comparable to existing models in terms of its ability to encode and reconstruct 3D shapes.

Table 3.1 Comparison of the reconstruction capability between our PC-VAE and reference models.

Methods	**MMD-CD**
PC-VAE (ours)	.0008
AE [60]	.0011
3dAAE-G [70]	.0008

In the second experiment, the PC-VAE's reconstruction quality is evaluated based on the different size of the latent representations. The PC-VAE is trained on point clouds from the car class of the ShapeNetCore repository and with different latent dimensions. The reconstruction performances of the shapes in the dataset are calculated using each trained PC-VAE with different latent dimensions. Fig. 3.6 shows the reconstruction error of different PC-VAE trained with latent dimensions $L_z = (2, 5, 10, 20, 32, 64, 128, 256)$. From 2D to 128D of the latent representation, the mean CD improves, and the standard deviation reduces with increasing latent dimensions. This indicates that increasing the size of the latent representation encodes more information about the input data, which subsequently improves the quality of the reconstructed shapes.

The reconstructed point clouds using PC-VAE are visually compared to the input point clouds (Fig. 3.7). However, there is no one-to-one correspondence between the points in the reconstructed and input point clouds as the network is trained with CD, which enforces the network to learn point clouds without a specific ordering of points.

Summary

In this section, a set of experiments is utilized to verify the reconstruction quality of the PC-VAE. By comparing the commonly used MMD-CD metric, PC-VAE shows comparable performance against regular AE [60] and 3dAAE-G [70]. Furthermore, by varying the size of the latent dimension in the second set of experiments, the PC-VAE's reconstruction

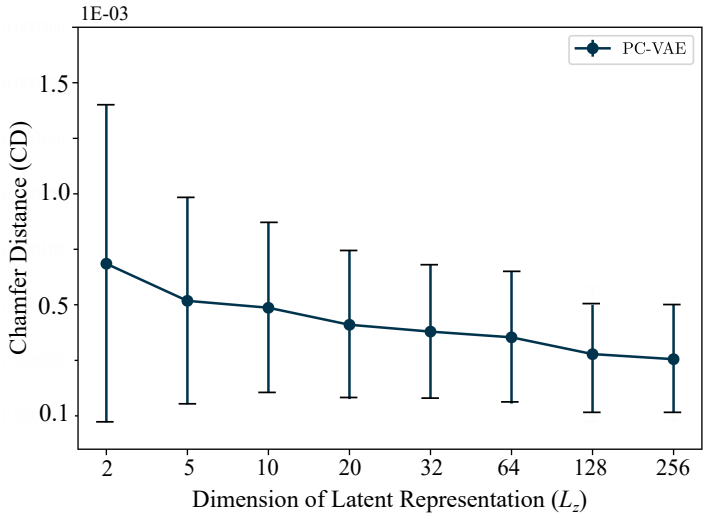


Fig. 3.6 Means and standard deviations of the reconstruction errors calculated using PC-VAEs trained on different dimensions of latent representation (L_z). Lower the CD better the reconstruction.

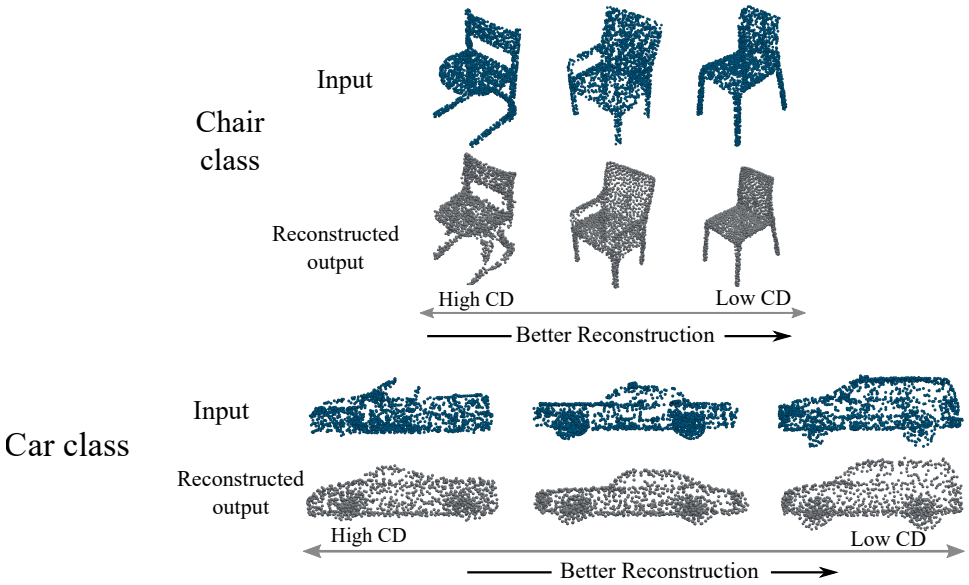


Fig. 3.7 Visualization of the input point cloud and their corresponding reconstructed output point cloud using the PC-VAE trained separately on the chair and car class of the ShapeNetCore models.

quality improves. Therefore, the proposed architecture is considered verified and feasible for shape-generative experiments.

3.5 Evaluation of the Shape Generative Capability of the Models

In previous analyses, PC-VAE showed comparable reconstruction ability to encode and reconstruct the shapes in the data set. However, for application of the PC-VAE for engineering design tasks, the generative capability of the PC-VAE needs to be evaluated

with respect to the realism and novelty of the generated shapes. Both aspects are central to the intended application domain of engineering design, such that shapes in some aspects closely resemble the data set, but also generate novel or diverse shapes that are sufficiently far away from the training data. Note that this evaluation aims at judging the model's ability to reconstruct unseen inputs and to generate completely different unique shapes.

The shape generative techniques (Section 3.2.3) are utilized to generate new shapes using the trained PC-VAE with $L_z = 128$. These newly generated shapes are referred to as set $\mathcal{G}$, and are evaluated based on three central aspects: (1) the realism of the generated shapes (Section 3.5.1), (2) the capability of the model to generate diverse shapes from the ones seen during the training phase (Section 3.5.2), and (3) the capability of the model to generate unseen novel shapes (Section 3.5.3).

3.5.1 Realism of Generated Shapes

We made an assumption that the latent representation learned by the PC-VAE is more continuous than the space learned by the PC-AE due to the additional regularization on the latent space. This should lead to the generation of more realistic car shapes when reconstructing interpolated samples from the latent space of the PC-VAE compared to the PC-AE. To verify our assumption, a 10-step linear interpolation is performed between randomly selected shapes in the latent space of the PC-VAE and PC-AE. These interpolated latent codes are reconstructed, and the reconstructions of the PC-VAE are compared with the PC-AE (Fig. 3.8). Each interpolation pair consists of an initial shape (x_{in}) and a target shape (x_{target}) from the data set $\mathcal{S}$, and the intermediate shapes (x') are reconstructed from the interpolated latent vectors using the decoder.

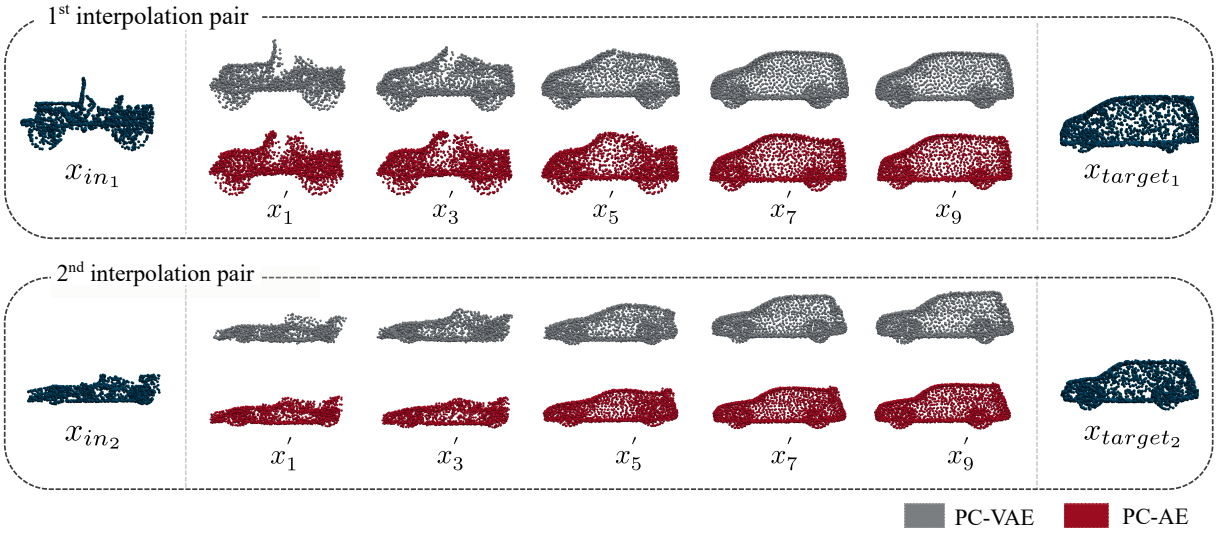


Fig. 3.8 3D shapes x_{in_i} and x_{target_i} represent reconstruction of the initial and target shapes. 3D shapes x'_1 - x'_9: Interpolation between x_{in_i} and x_{target_i} in 10-steps and reconstruction at step 1, 3, 5, 7 and 9. In each interpolation pair, the bottom row represents reconstructions of the interpolations using the PC-AE.

Results and Discussion

The reconstructions of the interpolated latent vectors are evaluated both qualitatively and quantitatively. By qualitative evaluation of the first interpolation pairs (Fig. 3.8), a major difference in the roof region is observed in the intermediate interpolations performed in the latent space of PC-AE that lead to a scattered points in the ($x_3^{'}, x_5^{'}$ and $x_7^{'}$) in the roof region of the car shape. On the other hand, the interpolation on the latent space of PC-VAE shows a more gradual change in the roof region and mostly maintained the curvature of the roof, in total leading to more realistic interpolated car shapes. Similarly, for the second interpolation pair (Fig. 3.8), the transition from a sports car (x_{in_2}) to a utility vehicle (x_{target_2}) is smoother in the roof region for shapes generated by interpolation on the latent space of PC-VAE.

Further in order to quantitatively assess the difference in the generative capabilities between PC-AE and PC-VAE, the closeness is calculated using the MMD-CD between the samples generated from each interpolation and the samples in the complete data set $\mathcal{S}$ (Section 3.2.4). Figure 3.9 shows the MMD-CD over all 50 interpolations for both PC-AE and PC-VAE. Overall, the PC-VAE showed a lower MMD-CD compared to the PC-AE, indicating that the interpolation led to shapes that resembled the shapes in the data set and hence the PC-VAE produced more realistic car shapes.

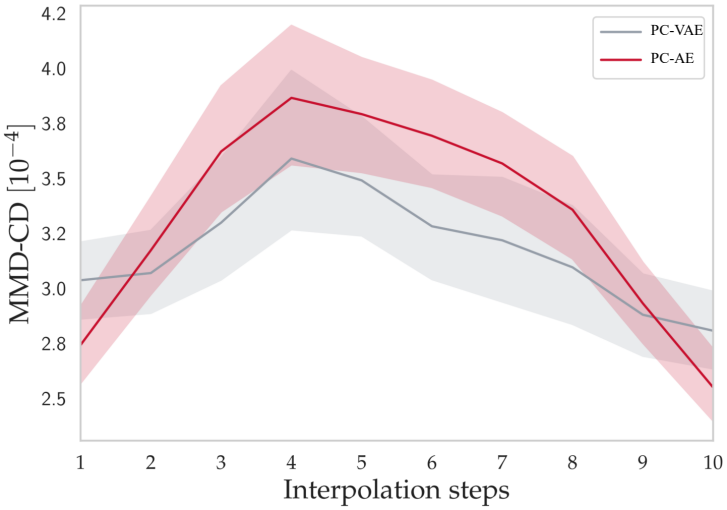


Fig. 3.9 MMD-CD measures for 50 randomly selected car pairs (each with 10 interpolation steps) using PC-AE and PC-VAE (shaded area indicates the standard deviation).

Lastly, to calculate the statistical significance of the difference in MMD-CD values between the PC-AE and PC-VAE, a non-parametric one-sided Wilcoxon signed-rank test is used. A one-sided test was performed based on the null hypothesis that the MMD-CD calculated from the 50 interpolations in the latent space of the PC-VAE (MMD_{PC-VAE}) was greater than or equal to the MMD-CD calculated from the PC-AE interpolations ($MMD - CD_{AE}$). The MMD-CD from the PC-VAE was significantly

smaller, $MMD - CD_{PC-VAE} < MMD - CD_{AE}$, for the tested sample ($z = -4.1449$, $p < 0.001$). Thus, the shapes generated from interpolation in the latent space of the PC-VAE were more realistic car shapes as indicated by a lower distance to the shapes in the data set $\mathcal{S}$.

3.5.2 Diversity of Generated Shapes

As a second measure to compare the generative capability of both PC-AE and PC-VAE, the diversity of the shapes generated by these models was evaluated. The generated shapes are assumed to be diverse when they are different from the shapes that are used for training the model. The diversity of the generated shapes is evaluated using a classification technique [69]. To compare the generative performance of PC-VAE against the baseline PC-AE, a Gaussian Mixture Model (AE+GMM) is trained on the learned latent space of the PC-AE to improve the generative capabilities, following the approach proposed in [60].

In this set of experiments, the shape-generative performance of PC-VAE is evaluated by training the model three times with different combinations of training and test splits (50/50, 70/30 and 90/10), as proposed in [69]. After training each model, 5000 latent representations are generated by random sampling from the distributions over the latent space (Section 3.2.3). The set of newly generated 5000 latent representations are referred to as $\mathcal{G}$.

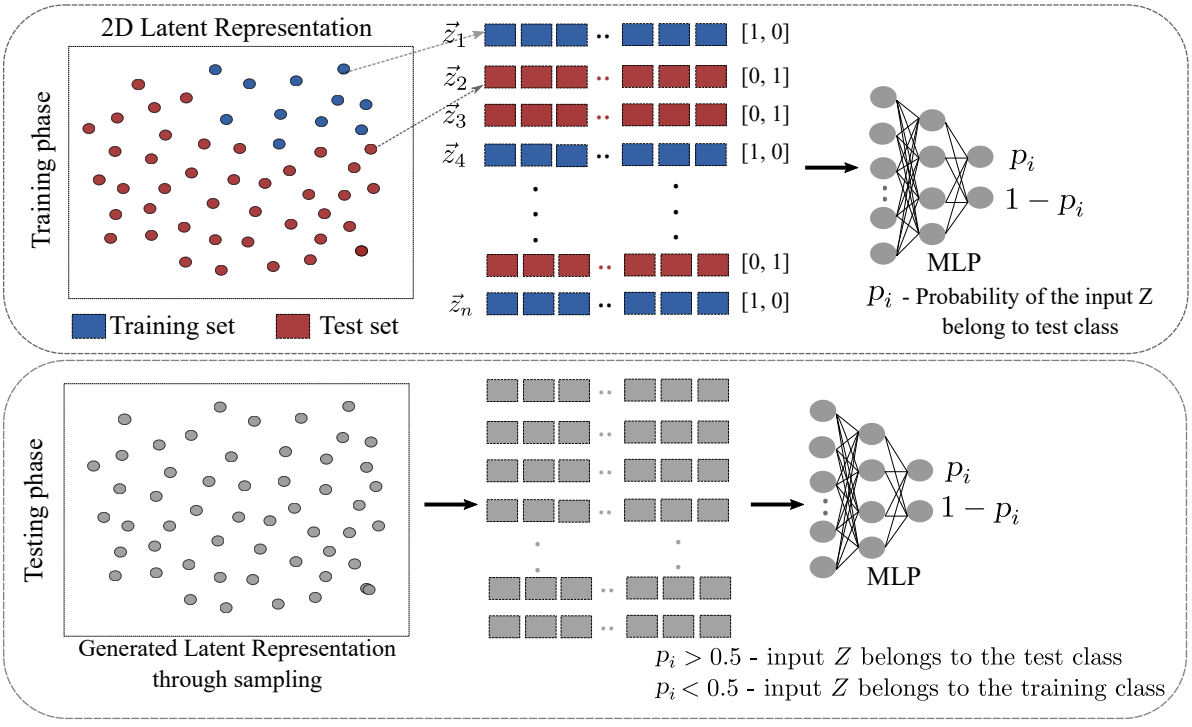


Fig. 3.10 Schematic overview of the input representations and labels of the MLP classifier in the training phase. In the testing phase, the trained MLP classifier is used to predict the labels of the newly sampled latent representations.

Results and Discussion

For each train-test-split, the diversity of the generated set $\mathcal{G}$ is calculated using the generative diversity metric (Section 3.2.4), which measures the percentage of the generated shapes that are classified as like the shapes in the test set based on their latent representations. To calculate the *diversity* of the generated shapes, an MLP-based classifier is trained on the latent representations of the whole data set, $\mathcal{S}$, with the information whether a sample belongs to the training set, $\mathcal{D}_{train}$, or test set, $\mathcal{D}_{test}$, as labels. Next, the latent samples of shapes at $\mathcal{G}$ are classified into either training,$\mathcal{D}_{train}$, or an unseen test set, $\mathcal{D}_{test}$ using the trained MLP classifier. Fig. 3.10, shows a schematic overview of the classifier training phase and the trained classifier is used in the testing phase to predict the train or test class labels of a newly generated sample.

The MLP classifier consists of two hidden layers $[100, 60]$ with a *tanh* activation function for the hidden layers, and the output layer has 2 units to predict the probabilities of the input samples belonging to the training and test set. The MLP architecture was optimized through a grid search. This experiment is repeated 10 times. In each turn, 5000 latent samples are sampled from the latent distribution and the MLP classifier is utilized to calculate the percentage of the 5000 generated samples that belong to the test set. The average percentage of generated samples classified as belonging to the test set is shown in Table 3.2. For all train-test-splits, the PC-VAE generated shapes with a higher diversity than the baseline model.

Table 3.2 Comparing generative diversity of the PC-VAE and the AE+GMM (baseline). Best generative diversity for each train-test-split shown in bold.

Train-test-split	PC-AE+GMM	PC-VAE Encoder
50/50	40.3 ± 0.80	**58.19 ± 0.19**
70/30	21.7 ± 0.5	**26.96 ± 0.8**
90/10	4.0 ± 0.3	**6.5 ± 0.08**

In addition to quantitative evaluation of the generative diversity of the models, three generated samples are randomly selected from each model and three nearest neighbors corresponding to the selected shapes are searched from the training set, $\mathcal{D}_{train}$ (Fig. 3.11). Both models generated realistic shapes; however, the PC-VAE generated shapes that were different to its nearest neighbors at $\mathcal{D}_{train}$ compared to the baseline PC-AE+GMM.

3.5.3 Novelty of Generated Shapes

A third quantitative evaluation is performed to compare the generative capability of both PC-AE and PC-VAE, based on their ability to generate *novel types* of car shapes. A generated car shape is considered novel if no shape from that car shape category is present in the training set that is used for training the models. For example, the pickup truck

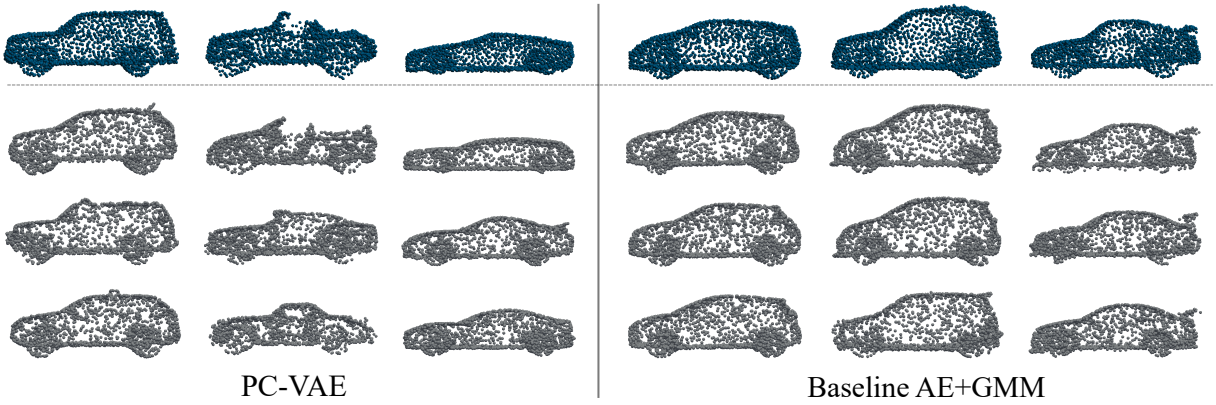


Fig. 3.11 Qualitative comparison of generative diversity of PC-VAE and baseline PC-AE+ GMM. **Row 1**: Three randomly selected shapes from the generated samples. **Row 2-4**: Nearest neighbors of the generated shapes in the training set (measured by CD).

shapes present in the data set are a combination of convertibles and coupe-like car designs. Therefore, if there are no shapes from the pickup truck category present in the training set and the trained model is able to generate pickup truck designs, then the generated shapes are considered as novel.

The PC-VAE and the baseline PC-AE are re-trained on the car class shapes after manually excluding all the pickup truck designs from the data. To measure the novelty of the generated shapes, the binary-classification technique is used (Section. 3.5.2). The *novelty* of the generated samples is calculated based on the percentage of the generated shapes that are classified as belonging to the test class; i.e., in this case the pickup truck designs. All the 630 separated pickup truck shapes are used as the test set, $\mathcal{D}_{test}$ for this experiment (Fig. 3.12a).

Results and Discussion

From each of the trained PC-VAE and baseline PC-AE+GMM, 1800 latent representations (three times the size of $\mathcal{D}_{test}$) are sampled, which are referred to as $\mathcal{G}$. The binary classifier is re-trained with the latent representations of training shapes grouped as one class and the latent representations of the test shapes grouped as the second class. The diversity, coverage, and MMD-CD metrics (Section 3.2.4) are calculated to estimate the closeness of the 1800 newly generated shapes with the pickup truck shapes in the test set $\mathcal{D}_{test}$. The diversity is measured based on sampled latent representations, and the coverage and MMD-CD are measured in the 3D domain. Therefore, the sampled vectors are reconstructed into 3D point clouds and their closeness is measured with respect to the point clouds in the test set (Fig. 3.12). Results are shown as averages over 3 repetitions in Table 3.3.

All three measures show that the PC-VAE is able to generate more truck-like designs and, hence, more novel shapes. Even if the generative diversity of the PC-VAE is only 3% i.e., 55 out of 1800 generated shapes represents pickup truck design, which is higher

Table 3.3 Comparing generative diversity, coverage, and minimum matching distance (MMD-CD) for a subclass of car shapes on test split (best performance for each measures shown in bold).

Models	Generative diversity	Coverage	MMD-CD
PC-AE+GMM	0.3 ± .13	7.3	.00048
PC-VAE	**3.1 ± 2.2**	**9.7**	**.00047**

than the baseline PC-AE+GMM. This provides significant evidence that the PC-VAE can generate unseen novel shapes. Figure 3.12b, c, shows examples of some of the generated shapes that were classified as belonging to $\mathcal{D}_{test}$ i.e., pickup truck designs. Note that the trucks generated by the PC-VAE show a slightly higher quality.

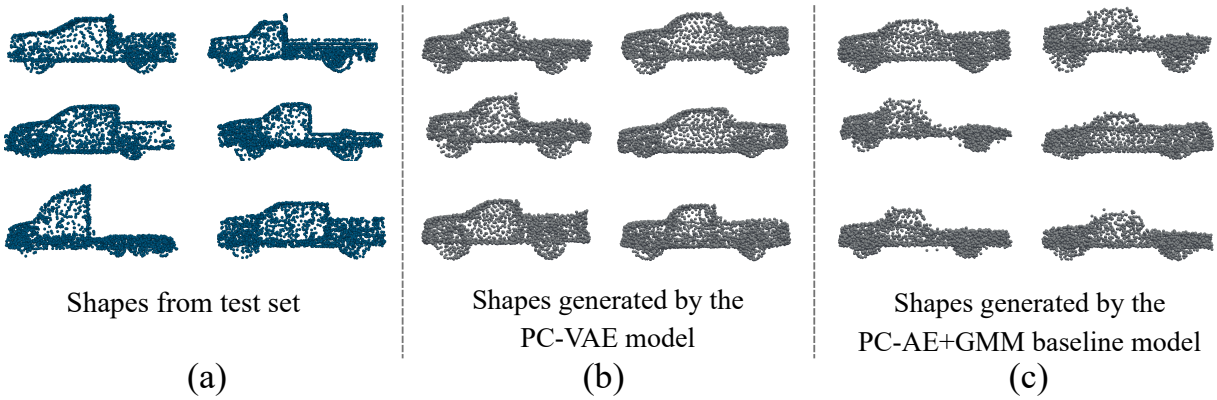


Fig. 3.12 **a**: Randomly sampled geometries from the test set consisting of pickup truck shapes. **b**: Shapes generated by the PC-AE+GMM baseline model that were classified as belong to the test set. **c**: Shapes generated by the PC-VAE classified as belonging to the test set.

Summary

In this section, PC-VAE is evaluated as a shape-generative model for generating new shapes that are suitable for engineering applications. Quantitative and qualitative measures have been used to evaluate the generated shapes in terms of realism, diversity and novelty of the shapes. The shapes generated by the PC-VAE outperform the shapes generated by the baseline autoencoder. The realism of the generated shapes is estimated by measuring the closeness of the shapes with the existing shapes from the data set. The PC-VAE is also capable of generating unseen diverse shapes by sampling the distributions over the latent space. This signifies the continuity of the probabilistic latent space of the PC-VAE, which overcomes the drawback of fuzzy point clouds generation at certain regions in the latent space of that PC-AE [61]. However, even if both interpolation and sampling in the latent manifold help to generate new designs, a guided search-based optimization is needed to reach any unexplored extreme regions in the latent manifold.

3.6 Guiding Novel Shapes Generation

As discussed in the earlier section, the continuity of the latent space of PC-VAE provides a benefit over other models for generating new designs through sampling and interpolation on the latent manifold. However, sampling from the latent distribution does not give control over the generation of specific target designs, which is of interest to the designer. To use the PC-VAE as the shape-generative model, the model should be able to generate target-oriented novel designs, i.e., to find regions in the latent space with unseen shape properties. Hence, developing a differential model that maps latent representations to desirable shape properties enables the use of gradient-based optimization to search for unseen shapes in the latent space.

To demonstrate the application of the PC-VAE for guiding unseen shape generations, an optimization task is set up in the continuous latent space of the PC-VAE to search for target car shapes from unexplored regions in the latent space. The pre-trained PC-VAE excluding pickup truck shapes (Section 3.5.3) is used for this optimization task, since the region in the latent space representing pickup truck designs is unexplored by the trained PC-VAE. For example, given a latent representation $\vec{z}_1$ of any random input shape, the optimization should be able to guide and modify the input $\vec{z}_1$ to the unexplored region in the latent space, generating the final optimized latent representation $\vec{z}_f$ (Fig. 3.13). Reconstructing the final latent representation $\vec{z}_f$ leads to the generation of an unseen novel pickup truck shape in targeted fashion.

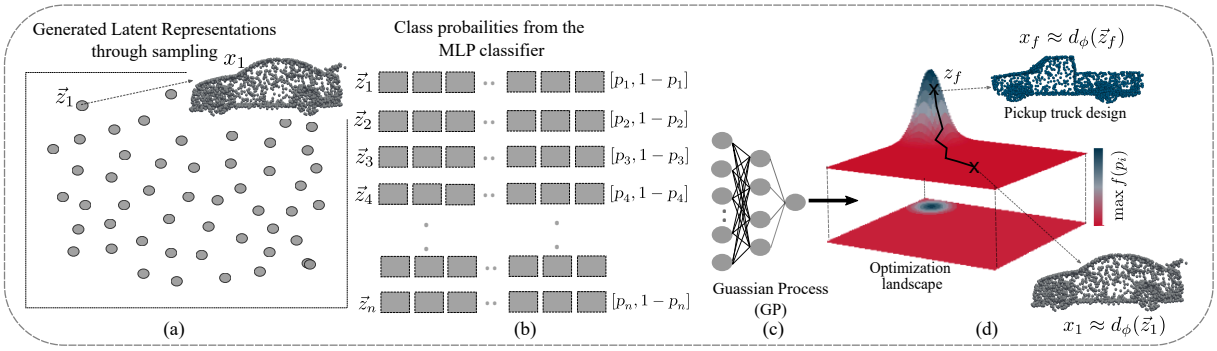


Fig. 3.13 Schematic overview of the gradient-based optimization task. **a**: Randomly generated samples from the latent distribution. **b**: The probability of the generated samples representing pickup truck designs (p_i) is calculated using the pre-trained MLP. **c**: A Gaussian process model is trained on the latent representations and their corresponding probabilities. **d**: Gradient-based optimization landscape on the latent manifold.

The previously trained MLP classifier is used is this experiment to represent the unseen shape properties (Section 3.5.3). Given an input shape to the trained MLP classifier, it predicts the probabilities of the given shape representing unseen pickup truck designs (p_i) and seen shapes from the training set ($1 - p_i$). Next, to create a smoother landscape to perform gradient-based optimization, a Gaussian process model (GP) is used as a property

predictor model. A GP model is trained on the latent representations as input and their corresponding probabilities (p_i) as output. The GP model makes use of gradient-based methods that are typically faster and, thus, using this differentiable model helps to map the latent representations to their unseen shape class probabilities p_i. The GP model trained with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm to generate a shape that maximizes the GP output (p_i), since the BFGS uses an iterative method to solve unconstrained non-linear optimization problems. Here, the higher the probability p_i, more the chance of the optimization algorithm reaching the region in the latent space with a higher gradient (Fig. 3.13c). A schematic overview of the optimization landscape is shown in Fig. 3.13, where the optimization starts with an initial random shape x_1 and the probability objective p_i is maximized to generate the final shape x_f, which represents a pickup truck design. For experimental verification of the final shapes proposed by the optimization approach, an MLP classifier is used to check the percentage of the final car designs representing pickup truck designs based on their latent representations.

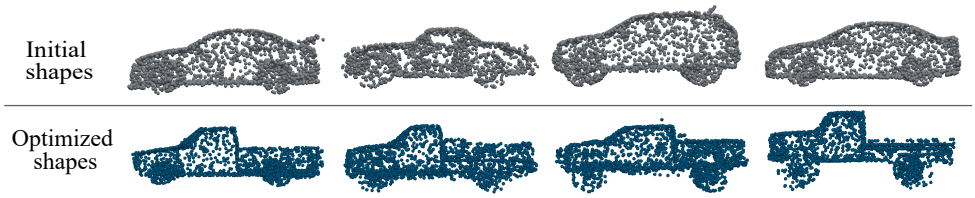


Fig. 3.14 **Top row**: Initial shapes chosen for optimization in the latent space of the PC-VAE. **Bottom row**: Reconstruction of the final optimized latent representations.

Results and Discussion

1800 shapes are sampled from the trained PC-VAE (Section 3.5.3), out of which 1744 shapes are selected for this optimization task as they don't represent pickup truck designs. Fig. 3.14 shows 4 examples of the initial shapes and their corresponding final pickup truck shapes that have been generated using the proposed optimization method. Out of 1744 final latent representations, 1307 are classified as representing pickup truck designs using the trained MLP classifier that differentiates between other designs and pickup truck designs. Hence, in 75% of the investigated cases, the PC-VAE could guide towards generating a novel design.

Summary

The proposed gradient-based optimization approach verified the exploratory ability of the PC-VAE and explained the use of the PC-VAE to generate target-oriented novel designs using gradient-based optimization. The optimization is performed on a GP model trained on latent representations and their corresponding probability of representing pickup truck

designs. However, since the training set of the GP model consists of 1744 samples, the predictive power of the GP model is lower in some cases, leading to several local minima, instead of a global optima. For this reason, 25% of the initial designs fail to converge and generate final shapes that do not represent pickup truck designs. However, for 75% of the investigated cases, the proposed optimization approach generates realistic unseen pickup truck designs. Therefore, gradient-based optimizations on the latent representations helps to generate a larger diversity of unseen designs using the PC-VAE.

3.7 Conclusion

This chapter introduced the architecture of the proposed 3D point cloud variational autoencoder (PC-VAE), as well as a set of experiments to evaluate its performance as a shape-generative model for engineering applications. The code for the proposed PC-VAE is available in the GitHub repository (https://github.com/HRI-EU/GDL4DesignApps). The main goal of this chapter is to answer $RQ2$.

RQ 2. How well can a generative model learn a compact representation of 3D designs for engineering optimization? And is this model suitable for generating diverse unseen designs that are useful for engineering applications?

In the context of building an experience-based CDS framework (Fig. 1.4), the first key challenge is to learn on existing 3D CAE models and generate a compact low-dimensional representation to facilitate faster search and optimization of 3D designs. The variational autoencoder serve as a promising approach to address this challenge.

First, to learn point cloud representations of 3D CAE models, the PC-VAE is proposed [77], which is built upon the networks in [60, 61]. The reference architecture is changed by adding two more convolutional layers after the max-pooling layers that stand for the mean and standard deviations of the input shapes. These allow the PC-VAE to create a probabilistic latent space, from where the 1D latent representation of the input shape is sampled. Additionally, batch normalization layers are added after the activation function of each layer in the encoder to normalize the input data. Normalizing is likely to produce activation with a stable distribution. The PC-VAE is optimized to maximize the combined loss function which involves two terms: First, the reconstruction loss that enforces the encoder to generate meaningful latent representations, so that the decoder can reconstruct the input shape from the latent representations. The second term in the loss function is the KL regularization that minimizes the distance between the observed data distribution and an assumed prior Gaussian distribution, which makes the latent space continuous.

Second, the proposed PC-VAE is evaluated based on its reconstruction and shape generation capabilities. The PC-VAE performed better than other models [60, 70] proposed in the literature. The PC-VAE can generate shapes that are both realistic, yet novel; i.e., the model is able to generate a higher percentage of diverse unseen shapes compared to the baseline PC-AE [61] model. In particular, the PC-VAE is capable of extrapolating; i.e., it can generate shapes that are fundamentally different from the shapes that are used for training the model.

Lastly, to allow the PC-VAE to generate new 3D designs for engineering applications, an optimization approach is used for an open-ended search of 3D designs. The utilized gradient-based optimization in the latent space is a versatile approach with respect to the optimization objectives. In other words, instead of guiding the model toward generating novel shapes, the PC-VAE can be used to guide the generation of shapes with different target properties, such as the aerodynamic performance of the generated car shapes.

Therefore, the research in the present chapter offers a promising step toward using the PC-VAE for generating a continuous low-dimensional search space for 3D CAE models. However, the generation of car shapes ideally involves both aesthetic design targets and engineering criteria, such as aerodynamic and structural efficiency. Therefore, in the following chapter, the focus is to use the latent space of PC-VAE as a common design space for search and optimization of designs based on multiple criteria.

Chapter 4

Exploiting a Generative Model for Guidance in 3D Designs Optimization

In the context of automotive design optimization (Fig. 1.4), the first key step is to represent the CAE models in a low-dimensional search space. Next, for automotive design generations, an increasing number of multi-disciplinary requirements have to be considered in the design process. These requirements comprise both aesthetic design targets and engineering aspects, such as aerodynamic drag and crash worthiness, which further require excessive computational effort. Therefore, setting up a multi-objective optimization for generating 3D designs involving both geometric and aerodynamic constraints is a challenging task. Further, estimating the aerodynamic performance of CAE models in an optimization process is a time-consuming process. Hence, in the context of our envisioned experience-based design optimization framework (Fig. 1.4), the steps involving multi-criteria based design generations and faster performance analysis of 3D designs are addressed in this chapter.

The experiments in the last chapter show that our PC-VAE can generate probabilistic low-dimensional latent representations of the 3D data using the PC-VAE's encoder. The decoder of the PC-VAE serves as a shape generative model for generating novel car designs. However, the feasibility of these latent representations to generate solutions satisfying multiple criteria remained unexplored in the previous chapter. Further, the PC-VAE is trained on geometric data in an unsupervised fashion and the information about geometries performances is neglected during training. This makes it unclear whether the latent representations hold information about geometric performance measures of the learned 3D designs. Hence, the motivation for this chapter is to utilize the low-dimensional latent manifold of the PC-VAE as the optimization search space and to analyze latent variables of PC-VAE in terms of two factors: First, as decision variables

for the multi-objective optimization problem to generate feasible solutions based on user preferences. Second, to evaluate whether these latent representations contain relevant structural and functional information about the input data, such that it can be used as input representations for training surrogate models to predict performances of 3D designs. The methods and experiments in this chapter address the proposed research questions *RQ*3 and *RQ*4.

The remainder of this chapter is outlined as follows: A 3D multi-objective design optimization problem is formulated from two perspectives, where the designer does and doesn't have prior information about their design preferences. The latent variables of the PC-VAE are used as the decision variables for a multi-objective optimization problem and the feasibility of these variables are analyzed for the engineering optimization tasks (Section 4.1). Next, information-theoretic measures are utilized to qualitatively evaluate the information content in the learned latent variables of the PC-VAE, such that these variables are able to represent engineering performance metrics. This helps to provide hints at the quality of the regression before training a surrogate model (Section 4.2). Lastly, the accuracy of the surrogate model is evaluated for predicting performance metrics of 3D car designs from the learned latent representation.

4.1 Exploiting Latent Representation for Preference Guided Design Optimization

In the automotive design optimization process, often designers have to face multi-criteria decisions (MCDM) at various stages of the design generation. Providing an efficient optimization method at an earlier design stage would allow higher design innovation potential. Often, in optimizations related to MCDM analysis, there are two scenarios [78–80]: The first scenario refers to a *posterior* approach, where the designer or the decision maker (DM) has no prior information about their design preferences and aims to generate a diverse set of design solutions. The second scenario refers to a *prior* or *interactive* approach, where the designer has knowledge about the design preferences and aims to find the best design solution for the optimization task. To support decision-making in both scenarios, a multi-objective optimization (MOO) problem is formulated in this research and the trained PC-VAE latent manifold (Section 3.2.1) is used as an optimization landscape to perform the MOO.

In the following sections, a brief survey of the literature is provided related to preference incorporation approaches in multi-objective optimizations (Section 4.1.1). In Section 4.1.2, a two-objective optimization problem is formulated for generating 3D designs satisfying these two criteria and the experimental setup for the optimization problem is detailed (Section. 4.1.3). Next, to provide a "warm start" to the optimization algorithm, a seeding

strategy is proposed to initialize the algorithm by exploiting the learned latent space knowledge of the PC-VAE (Section 4.1.4). Lastly, in the context of the optimization scenario considering prior user preference (Section 4.1.5), a combinative strategy is proposed that utilized the prior proposed seeding method to generate a final solution that reflects the DM's design preference. Finally, the conclusions of this set of experiments are detailed (Section. 4.1.6).

4.1.1 Prior Art

The goal of a multi-objective optimization is to help a DM identify solution(s) of interest (SOI) achieving satisfactory trade-offs among conflicting criteria. Both the *posterior* and *prior* approach for preference incorporation in an optimization task, requires the DM to fix the preference in terms of fixing the relation between the objectives. The *prior* approach relates to fixing relations between the two objectives before the start of the optimization search process, which eventually leads to the generation of a single solution that matches the DM's preference. While in the *posterior* approach, relations between the objectives are not fixed prior to the optimization search process. This leads to the generation of a set of solutions that represents an approximation of the Pareto set. The DM can choose a solution from the Pareto set that provides the most appropriate trade-off among many. Therefore, the later approach can be solved with multi-objective optimization, where the DM takes the decision at a later stage of the optimization process. While the former approach can be solved with a single objective optimization, where the user needs to decide at first of the optimization process.

Prior research [81–83] utilized the low-dimensional latent representation of an AE or a VAE for single and multi-objective optimizations. This overcomes a major factor contributing to the complexity of the optimization problem by reducing the number of decision variables. Therefore, in our design optimization task, the latent representation of the PC-VAE [77] is used as the decision variable for the optimization task.

Posterior Incorporation of Preference in Multi-Objective Optimization

In the *posterior* preference incorporation scenario, the aim of the multi-objective optimization is to generate a diverse set of solutions, which can be given back as a suggestion to the DM. To perform MOO, a wide range of evolutionary algorithms (MOEA) are developed over the decades, which are classified into two categories. The first category corresponds to those algorithms that include the use of selection mechanisms based on fitness sharing but does not include mechanisms for the preservation of good solutions (elitism), e.g., NPGA, NSGA[84], VEGA [85], MOGA [7]. The second category of algorithms is characterized by the use of the elitism strategy, e.g., SPEA [86], PAES [87], SPEA2 [88], NSGA-II[89]. Besides the above-mentioned evolutionary algorithms, several meta-heuristics like Ant

Colony Optimization (MOACO) [90] and Multi-Objective Particle Swarm Optimization (OMOPSO)[91] have been developed to solve MOO problems. The OMOPSO extends the Particle Swarm Optimizer (PSO), which mainly works for the single objective optimizations. At each generation, a list of leaders is selected based on crowding distances and each population is subdivided into three different subsets where a different mutation operator is applied to each subset. Hence, OMOPSO helps in generating a list of final solutions based on Pareto dominance.

Seeding the Initial Population of MOEA: A typical MOEA starts from a set of initial solutions and iteratively improves the solutions during the optimization process. Here, MOEA relies on random initialization of the initial populations. But often in large and complex search spaces, this random method leads to an initial population which consists of infeasible solutions [92]. Previous research on modifying the initial population of a MOEA [93–95] showed that constructing a well-suited initial population can speed up evolutionary algorithms (EAs) and reduce the convergence time to achieve acceptable results. This method of injecting knowledge about the problem into the initial population of the MOEA is known as *seeding*. Fraser et. al [96] proposed that leveraging target-oriented knowledge about the problem and incorporating it into the initial population can considerably improve the algorithm's performance. Opposed to prior research, the data-driven latent manifold of the PC-VAE is exploited to generate a target-oriented seeding population for MOEAs, such that the optimization maintains a good diversity of the final non-dominated solution set.

Prior Incorporation of Preference in the Optimization Task

Often, experienced decision makers (DMs) have prior design preference information that can be incorporated into the optimization algorithm to generate an optimal solution. Hence, in the *prior* preference incorporation scenario, there are different methods for DMs to specify preferences [79]. The most common methods include dominance relationship, fitness evaluation, termination criteria, constraint limits, etc. Another popular approach is the weighted-sum method (WSM) [97] that relies on eliciting a designer's preference through weighing the criteria. WSM generates a single final solution point by presumably incorporating a single set of weights based on DM preferences. The two most popular approaches for weighing multiple criteria are the weighted sum [98] and Tchebycheff approach [99]. However, a weighted sum is capable of the linear approximation of a preference function only when the feasible design space is convex [100].

The weights in WSM need to be set according to the relative magnitude of the objective functions rather than the relative importance of the objectives [101–103]. Arbitrary specification of these weight values can lead to an undesired solution. Often, even with

full knowledge of the objectives and satisfactory selection of weights, the final solutions may not necessarily reflect the intended preferences that are supposedly incorporated in the weights [100]. Hence, a strategy needs to be adapted to set the prior weights in WSM, such that the final solution fits the designer's preference criteria. In this research, an adaptive strategy is proposed to adjust the weights in WSM, such that the final solution of the convex optimization matches the DM' preference.

4.1.2 MCDM Problem Formulation and Proposed Approaches

To consider a multi-criteria problem for automotive designs and to keep the computational cost low, a multi-objective design optimization task is formulated using two 3D target shape matching optimizations (TSMO). TSMO is used as a benchmark problem for design optimization frameworks [104, 105]. A TSMO is a scenario, where an initial shape x_i, represented by a set of parameters $\vec{z}$, is approximated to a target shape by minimizing a metric that measures the difference between the shapes. Here, two reference (target) shapes are considered which reflect two targets, e.g., an aesthetic design target (x_1) and an aerodynamic target (x_2). The multi-objective design problem can be formulated as,

$$
\begin{aligned}
&\min_{x \in D} \Psi_1(\vec{z}), \Psi_2(\vec{z}) \\
&\Psi_1(\vec{z}) = CD(d_\phi(\vec{z}), x_1) = CD(x_i, x_1) \\
&\qquad = \sum_{p_i \in x_i} \min_{p_o \in x_1} \|p_i - p_o\|_2^2 + \sum_{p_o \in x_1} \min_{p_i \in x_i} \|p_i - p_o\|_2^2 \\
&\Psi_2(\vec{z}) = CD(d_\phi(\vec{z}), x_2) = CD(x_i, x_2) \\
&\qquad = \sum_{p_i \in x_i} \min_{p_o \in x_2} \|p_i - p_o\|_2^2 + \sum_{p_o \in x_2} \min_{p_i \in x_i} \|p_i - p_o\|_2^2
\end{aligned}
\tag{4.1}
$$

where Ψ_1, Ψ_2 are the two objective functions and $\vec{z}$ is an n-dimensional latent vector of a PC-VAE, which are used as decision variables for the optimization. The optimization objective functions (Ψ_1 and Ψ_2) receive an n-dimensional latent vector $\vec{z}$ of each 3D shape from the latent space of the trained PC-VAE. Next, the decoder of the PC-VAE reconstructs the 3D shape from the latent vector $\vec{z}$, where the objective function computes the difference between the reference shapes to the current deformed shape using the Chamfer Distance (CD).

In a real-world scenario, the objectives are typically of different magnitudes. To normalize the objective function values, the upper bound (Nadir point z_i^N) of the Pareto optimal set needs to be determined. The upper bound is obtained by separating i objectives into i sub-problems, and these sub-problems are minimized independently using a single-objective optimization strategy. Each objective function ($\bar{\Psi}_i$) is normalized with the upper

bound value of the opposite objective function, and the normalized objective functions ($\bar{\Psi}_i$) are defined as,

$$z_i^N = \min\left(\Psi_i\left(\vec{z}\right)\right), i = 1, 2$$
$$\bar{\Psi}_i = \frac{\Psi_i}{z_i^N} \text{ and } 0 \leq \bar{\Psi}_i \leq 1 \tag{4.2}$$

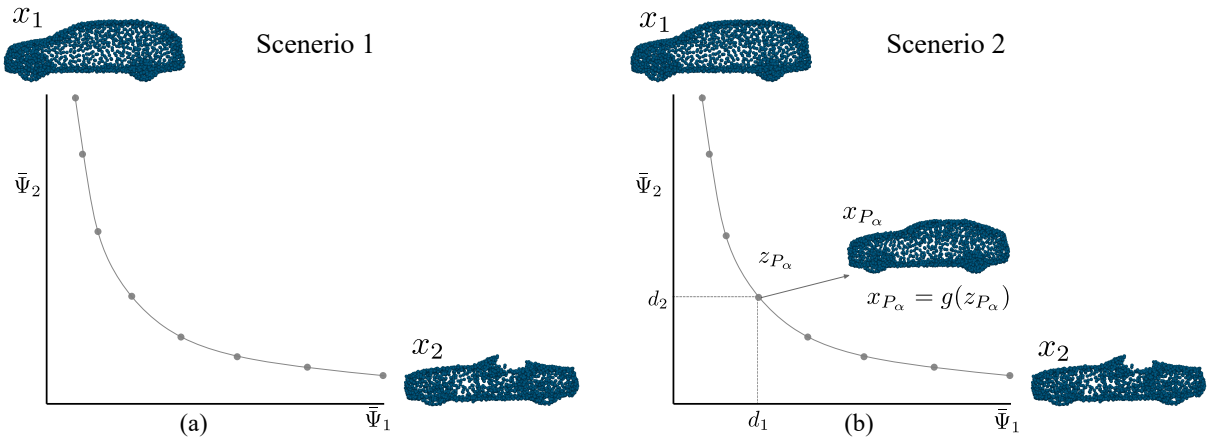


Fig. 4.1 Two scenarios for multi-criteria decision analysis where x_1 and x_2 are two reference shapes for the optimization task. a) Scenario 1 for generating a range of diverse solutions. b) Scenario 2 for generating solution(s) of interest based on DM's design preference.

This multi-objective optimization problem leads to several scenarios for multi-criteria decision analysis. Based on the objective of this research, two optimization scenarios are considered here: First, when the DM has no prior preference, the target is to generate a wide range of design possibilities between two reference shapes x_1 and x_2 (Figure 4.1a). This relates to *posterior* incorporation of preference in the optimization task. The second scenario relates to the *prior* incorporation of DM's preference to generate a solution of interest (SOI), x_{P_ρ}, that is an intermediate design between the two reference shapes x_1 and x_2 (Figure 4.1b). As a preference information, the DM needs to define the distances of d_1 and d_2 from the reference designs x_1 and x_2, respectively.

Note that prior to all the following experiments, we generated a set of non-dominated solution points as a first approximation of the Pareto front. This set is required to evaluate the performance of the different algorithms, e.g., using the IGD measure (Section 4.1.4). This solution set (best-NDS) is generated as follows. First, the multi-objective optimization problem is converted into a single-objective function using a weighted-sum method. Then, we systematically sample weights between [0, 1] to generate solutions for best-NDS. Of course, this is only possible here to benchmark the algorithms using target shape matching, which has a comparably low function computation time. In practical use cases like car aerodynamic optimization, the computational time for a systematic weighted-sum method is too high.

Generate Diverse Solutions

In the first scenario (Fig. 4.1a), the target is to obtain a diverse range of design solutions that approximate the Pareto front of the multi-objective optimization. The objective functions in Eq. 4.1 are normalized, and the final normalized optimization functions are minimized (Eq. 4.3).

$$\min \quad \bar{\Psi}_1(\vec{z}), \bar{\Psi}_2(\vec{z}) \tag{4.3}$$

Methods to Generate Target-Oriented Seeding of a MOEA: The MOEAs used to solve MOO are generally initialized with a random initial population. To improve the diversity within the non-dominated set, a method is proposed (Fig. 4.2) to generate a target-oriented initial population for MOEAs.

In the proposed method, the normalized optimization functions in the MOO (Eq. 4.3) are decomposed into 2 single-objective sub-problems. Each single-objective optimization is performed using the covariance matrix adaptation evolution strategy (CMA-ES) [106]. The CMA-ES method was selected due to its suitability for small populations [106], high convergence ratio, and a low number of hyper-parameters. Next, the final solutions of each single-objective optimization are obtained, and a 100-step interpolation is performed between the two final vectors in the latent manifold of the PC-VAE. The interpolation is performed by calculating ratios of the contribution from two final solutions, and then enumerating these ratios to construct a vector for each ratio (Section 3.2.3). The 100 latent vectors generated through interpolation (*Lerp-seed*) are used for the initialization of the initial population of an MOEA.

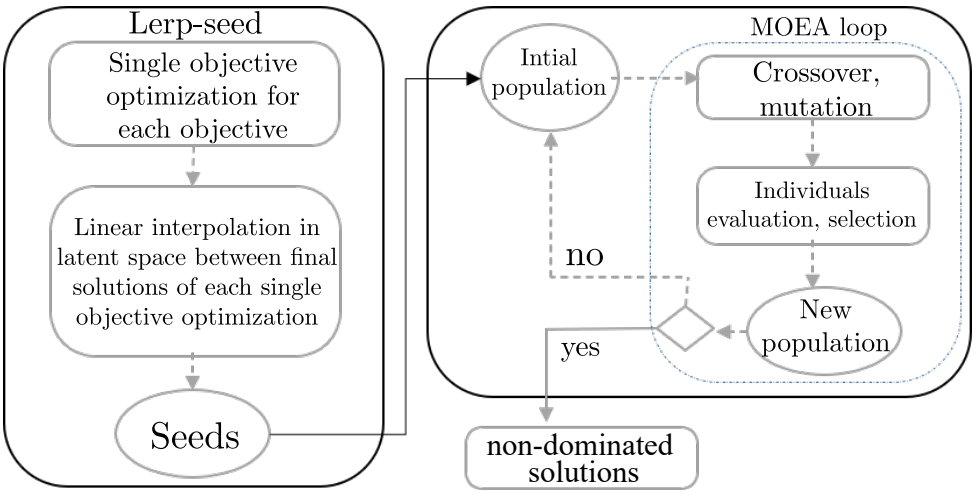


Fig. 4.2 Proposed seeding mechanism in an MOEA.

Generate Single Solution of Interest

For the second scenario in Fig. 4.1b, to generate a single solution point that reflects preferences, the DM needs to specify comparative preference states for each objective function value. However, here the objective functions are distance functions, so an assumption is made that the DM has knowledge about the few best Pareto optimal points (best-NDS), and the DM can specify the desired distance values of d_1 and d_2 for each objective. Hence, the DM's preference can be defined as a distance ratio (ρ) function:

$$\begin{gathered} \rho = \frac{d_1}{d_2} \\ \text{the normalized preference ratio}(\bar{\rho}) \\ \text{with } \bar{d}_1 = \frac{d_1}{z_1^N} \text{ and } \bar{d}_2 = \frac{d_2}{z_2^N} \\ \bar{\rho} = \frac{\bar{d}_1}{\bar{d}_2} \end{gathered} \tag{4.4}$$

The preference relation describes the relative importance over a set of objective functions. Three scenarios for 3 different preference relations are considered here (Fig. 4.3):

1. when the quality of importance of objective $\bar{\Psi}_1$ is higher compared to $\bar{\Psi}_2$ ($d_1 > d_2$), $\rho = 2$
2. when objectives are of equal importance ($d_1 = d_2$), $\rho = 1$
3. when the quality of importance of objective $\bar{\Psi}_2$ is higher compared to $\bar{\Psi}_1$ ($d_1 < d_2$), $\rho = 0.5$

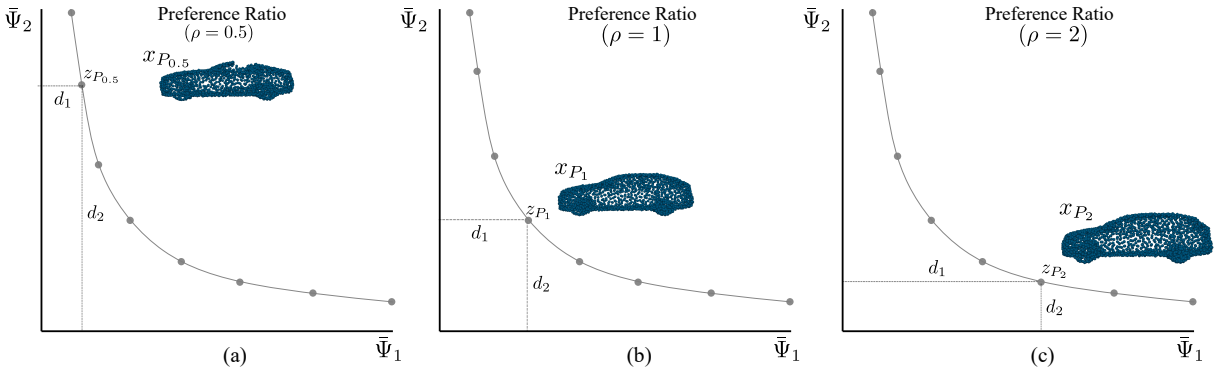


Fig. 4.3 Three preference scenarios based on selection of d_1 and d_2 by the DM for each set of reference shapes.

To incorporate the preference information into the optimization problem, two different approaches are considered. First, the WSM, where the objective functions (Eq. 4.3) are transformed into a single-objective (Eq. 4.5) and the preference information is incorporated

by adjusting the weights of the WSM. Generating a final solution by selecting a set of weights in WSM may not necessarily reflect the intended preferences, hence the research on adapting the weights in WSM is under-explored in the literature [107, 108]. Therefore, the aim is to propose an adaptive strategy to identify the best weight values in WSM to match the DM's design preference in the multi-objective optimization task.

$$F\left(\vec{z}; w_1\right) = w_1 \bar{\Psi}_1\left(\vec{z}\right) + w_2 \bar{\Psi}_2\left(\vec{z}\right) \tag{4.5}$$

As a baseline method to compare with the WSM, a second optimization approach is utilized where the preference information ($\bar{\rho}$) is incorporated using an additional objective in MOO (Eq.4.3).

Determining the Weights of WSM: The normalized objective functions $\bar{\Psi}_1$ and $\bar{\Psi}_2$ are summed in WSM and the aggregated objective function $F\left(\vec{z}; w_1\right)$ (Eq. 4.5) is minimized. The final solution in the WSM depends on the values of the weights of w_1 and w_2, which are in the range $0 \leq w_{1,2} \leq 1$. It is common practice to choose weights in a way that their sum equals to 1, i.e., $w_2 = 1 - w_1$.

The proposed strategy to determine the optimal weight in WSM involves a two-step strategy, where the weighted-sum optimization is performed first with a random weight w_1. Next, a global optimization is performed to reduce the distance between the WSM output and the DMs preference ($\vec{\rho}$) (Eq. 4.6). The proposed combinative method (Fig. 4.4) helps to determine a global optimum weight (w_1) in WSM, such that the final solution of the optimization with the selected weights is closer to the DM's design preference.

$$C(\vec{z}; w_1) = (\bar{\rho} - \frac{\bar{\Psi}_1(\vec{z})}{\bar{\Psi}_2(\vec{z})})^2 \tag{4.6}$$

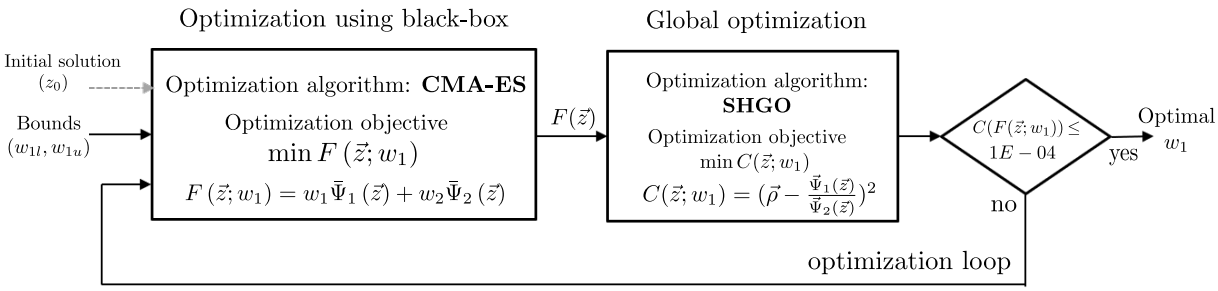


Fig. 4.4 Optimization strategy to determine weight w_1 in a weighted-sum method to match the DM's design preference.

In the first step (Fig. 4.4), a CMA-ES algorithm is considered for the WSM. CMA-ES often requires providing a feasible solution as a starting point for the initial candidate solution. To select the initial candidate solution in the CMA-ES algorithm, the initial population vectors generated from *Lerp-seed* are considered (Fig. 4.2) and their normalized

distance ($\bar{d}_1$ and $\bar{d}_2$) are calculated from the respective reference shapes (x_1, x_2) in the optimization task. The solution vector in *Lerp-seed* with normalized distances closest to the preferred distance function $\bar{d}_1$ and $\bar{d}_2$ is chosen as the initial solution vector ($\vec{z_0}$) for the CMA-ES. This approach of selecting the initial solution vector ($\vec{z_0}$) is intended to reduce the number of generations needed for convergence to the optimum. Next, the objective function of the WSM (Eq. 4.5) is optimized using CMA-ES.

In the second step, a global optimization algorithm is utilized that demonstrates a promising global search capability. A global minimum is a point where the function's value is the minimum of all possible points in the function's domain. There are numerous global optimization techniques, yet only a few of them are derivative-free. The simplicial homology global optimization (SHGO) [109] algorithm is a promising, derivative-free global optimization (GO) algorithm, and it also returns all other local and global minima. It follows an iterative clustering mechanism that maps the hyper-surface of the objective function into topological matrix and determine the local minima of the topography. Hence, the SHGO optimizer determines the global minimum weight w_1 by minimizing the cost function in Eq. 4.6. The final optimal weight w_1 helps the DM to understand the relation between the weights in WSM and the final solution.

4.1.3 Experimental Setup

To perform the above discussed optimization experiments (Fig. 4.5), the PC-VAE [77] from the previous chapter is used to generate latent representations of the input shapes, which form the decision variables of the optimization. The PC-VAE is trained with a latent dimension L_z=128, i.e., each decision variable is a 128-dimensional $\vec{z}$ vector. All remaining parameters and hardware settings for training the PC-VAE are the same as utilized in the verification experiments (Section 3.3.1). The network is trained on point cloud representations of car shapes from the ShapeNetCore repository [18].

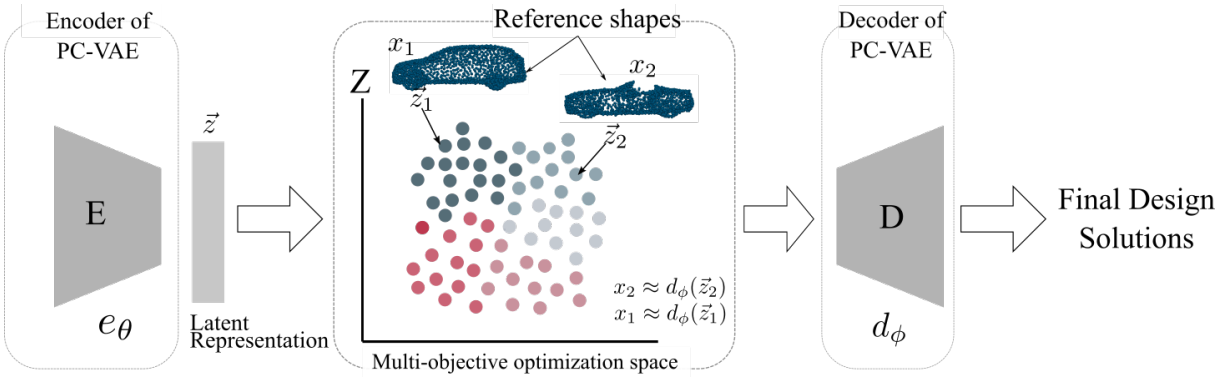


Fig. 4.5 Optimization workflow with different sets of reference shapes.

A schematic workflow of the optimization task is shown in Fig. 4.5, where a pre-trained PC-VAE encoder is considered to encode input shapes in the latent space Z. The optimization is performed in the latent space, while the objective function in Eq. 4.1 is calculated in the 3D space by reconstructing the latent representations at each generation. In the end, the final solution is reconstructed using the decoder.

Problem Instances: Different combinations of reference shapes (x_1, x_2) are selected to verify the proposed method in different multi-objective optimization scenarios. To select reference shapes, the latent representations of the shapes from the training set are clustered using k-means clustering. The clustering organizes the data according to geometric similarity in the latent space of the trained PC-VAE.

Three problem instances (PIs) are selected (Fig 4.6) based on 3 scenarios. Each PI consists of two reference shapes, that are selected from the different clusters in the latent representation.

- PI-1 refers to Fig. 4.6a, where selected reference shapes are from two difference clusters.
- PI-2 refers to Fig. 4.6b, where selected reference shapes are from two nearby clusters.
- PI-3 refers to Fig. 4.6c, where selected reference shapes are from similar clusters.

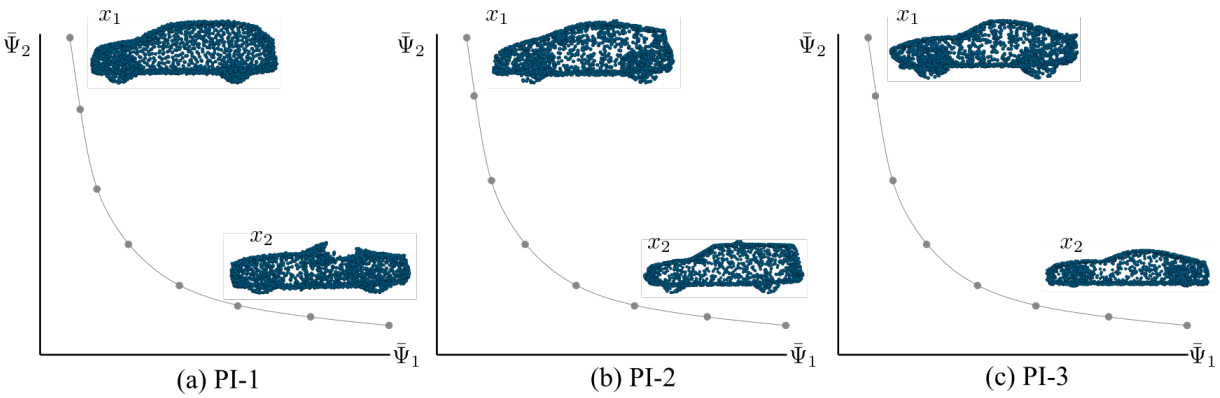


Fig. 4.6 Reference shapes in each of the three problem instances (PIs).

4.1.4 Optimization for generation of diverse design proposals

The multi-objective optimization (Eq. 4.1) is performed using reference shapes from the 3 PIs. The multi-objective evolutionary algorithms (MOEAs) chosen for the optimization tasks are NSGA-II, OMOPSO (both with random seeds), and *Lerp-seed*-OMOPSO (with the proposed approach of seeding from Fig. 4.2).

Parameterization of MOEA: The parameters of MOEAs are set in such a way to have a fair comparison among the algorithms (Table 4.1). The genetic algorithm NSGA-II uses an internal population equal to 100, and OMOPSO has been configured with 100 particles. For assessing the search capabilities of the algorithms, 10 independent runs of each experiment are performed for 100 generations, and the results are compared using different metrics.

Table 4.1 Parameterization of MOEA algorithms.

Parameterization used in NSGA-II	
Population size	100 individuals
Selection of parents	binary tournament
Parameterization used in OMOPSO	
Swarn size	100 particles
Mutation	uniform+non-uniform
Leader size	100

***Lerp*-seeding of MOEA:** To generate the seeds for the MOEA with the proposed approach (Fig. 4.2), the two objectives (Eq. 4.3) are converted into two single objective sub-problems. Next, each sub-problem is optimized using CMA-ES. CMA-ES algorithm samples solutions from a multivariate Gaussian distribution and all solutions are ranked and the distribution parameters are updated. The CMA-ES[110] utilizes a (μ, λ) strategy, where the chosen population size λ is set to 10, and the number of parents μ is set to 3. The algorithm is set to an initial step size of 0.01 and 100 generations. The objective functions $\bar{\Psi}_1$ and $\bar{\Psi}_2$ (Eq. 4.3) are minimized separately, where $\vec{z}$ is a 128-D latent vector of the trained PC-VAE. Even though there is a large number of decision variables, the convergence of CMA-ES is observed through monitoring the fitness function over 100 generations (Figure not shown here). The final solution of each single objective optimization is a 128-D latent vector that represents a car design. The solutions for *Lerp-seed* is generated by a 100-step linear interpolation in the latent space of the PC-VAE between two final solution vectors, which are added as an individual to the initial population of MOEA. The effect of the *Lerp-seed* with OMOPSO is analyzed and compared with no seed MOEAs using the following performance metrics.

Performance Metrics: For comparing the behavior of the MOEAs, four metrics are considered:

- *Hyper-volume indicator (HV)*[86] measures the volume enclosed by a solution set and a specified reference point. Since the objective functions are normalized, the reference point is chosen as $(1, 1)$. It indicates a quality of convergence and diversity. A high HV value is preferable, reflecting the set having good diversity.

- *Inverted generational distance (IGD)*[111] indicates how far are the non-dominated solutions produced by the algorithm from the reference points in the true Pareto front (best-NDS) of the problem. A smaller IGD indicates that all the elements generated are in the true Pareto front.
- *Spacing (SP)* as suggested in [112] measures the distance variance of neighboring vectors in the Pareto front. Lower SP shows that all the non-dominated solutions found are equidistantly spaced.
- *Number of function evaluation (NFEs)* is calculated by multiplying the population size (or the number of particles) and the number of generations, i.e., the total number of experimental runs.

Results and Discussion

The performance of different MOEAs (NSGA-II, OMOPSO with random seeds and *Lerp-seed* in OMOPSO) are assessed in terms of generating diverse design solutions for the multi-objective optimization problem (Eq. 4.3). Table 4.2 presents mean values of HV, IGD, and SP of the optimization task using NSGA-II, OMOPSO and *Lerp-seed*-OMOPSO. Out of the three MOEAs, *Lerp-seed*-OMOPSO achieved the best results with respect to mean HV and mean IGD values for the multi-objective optimization task.

Table 4.2 Results obtained from optimization of problem instance 1 for NSGA-II, OMOPSO, and *Lerp-seed*-OMOPSO. Best performance metrics (HV, SP and IGD) among all methods are shown in bold.

Methods	**NFEs**	**HV** (mean)	**SP** (mean)	**IGD** (mean)
NSGA-II	10000	0.473	**0.186**	0.060
OMOPSO	10000	0.484	0.229	0.059
Lerp-seed OMOPSO	10000	**0.490**	0.211	**0.055**

Additionally, to compare the statistical significance of the HV difference, the Mann-Whitney test is performed to prove the difference between *Lerp-seed*-OMOPSO and other MOEAs (NSGA-II and OMOPSO initialized with random seeds). The analysis reveals significant statistical differences ($p < 0.05$) between *Lerp-seed*-OMOPSO and other MOEAs. Hence, *Lerp-seed*-OMOPSO with high HV and low IGD values provides better distribution of the generated non-dominated solutions.

Nevertheless, the NSGA-II returns a higher number of non-dominated solutions and has a lower spacing (SP) between the solutions. Yet, OMOPSO (with or without seeds) provides a better spread of solutions than NSGA-II. Besides the quantitative comparisons, the non-dominated solutions plot obtained from one optimization run with shapes from

PI-1 (Fig. 4.7), shows that the *Lerp-seed*-OMOPSO has a better spread of solutions than NSGA-II and OMOPSO with random seeds.

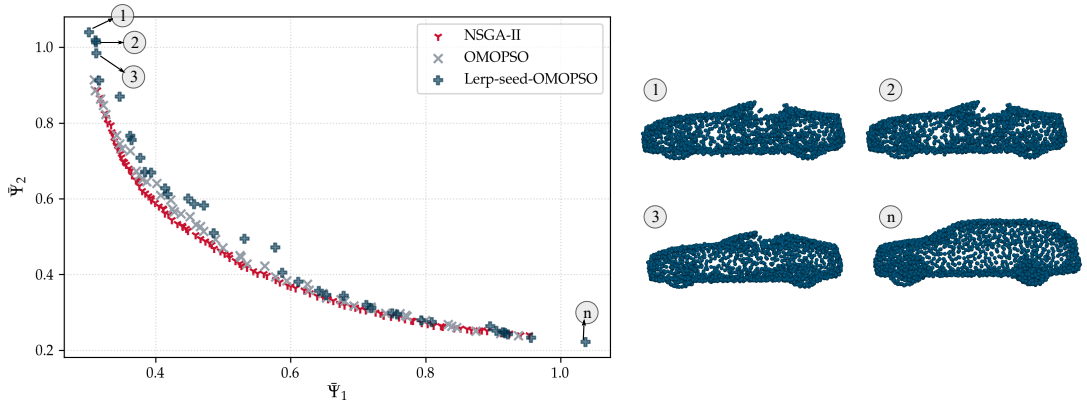


Fig. 4.7 Pareto fronts obtained by the NSGA-II, OMOPSO, and *Lerp-seed* OMOPSO for optimization with reference shapes from PI-1 (shown for 1 run).

The optimizations are also performed for the other two other problem instances (PI-2 and 3), to take a closer look at how the mean values of HV and IGD change with respect to NFEs as the search evolves. Figure 4.8 and 4.9 show the mean HV and mean IGD of the non-dominated solution set for 10,000 NFEs. The optimizations are for each of the 3 problem instances and the mean HV and IGD are calculated as an average over 10 runs. The proposed *Lerp-seed* OMOPSO requires initial 2000 function evaluations to run the two single objective optimization using CMA-ES, so the mean HV and mean IGD plot for *Lerp-seed* OMOPSO starts after 2000 NFEs to make fair comparisons.

For all 3 PIs, the OMOPSO algorithm is initialized with *Lerp-seed* as the initial population outperforms other algorithms that initialized with random seeds. This leads to the conclusion that spending some additional computational effort in constructing a target-oriented initial population of an MOEA helps to improve the convergence in the optimization problem. However, the exploitation of the low-dimensional latent domain knowledge comes with an additional cost of training the PC-VAE with 3D shapes.

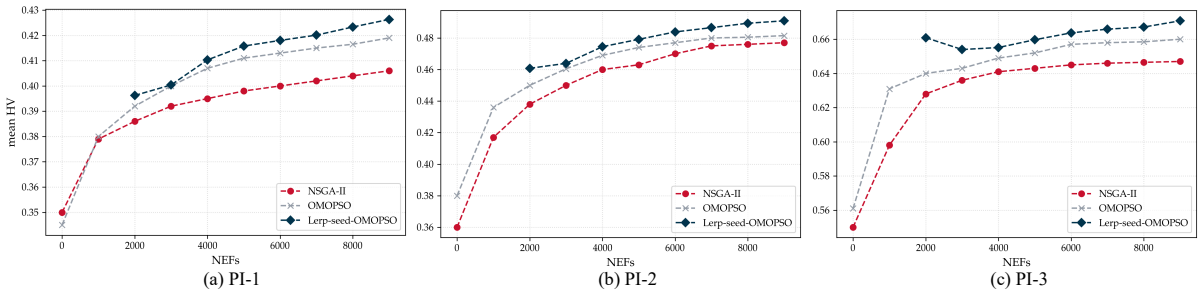
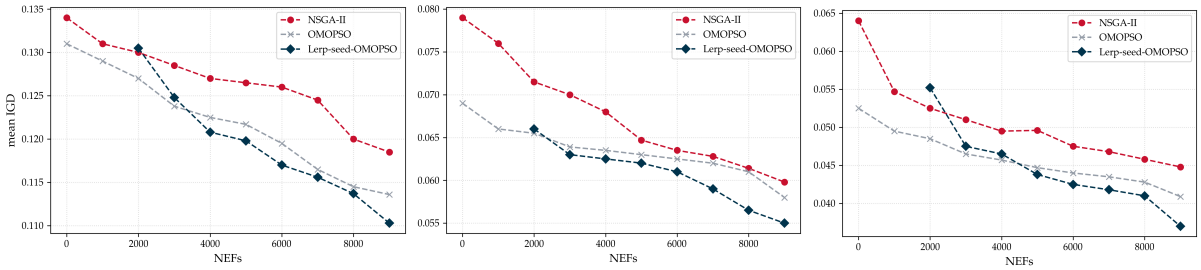


Fig. 4.8 The change of mean HV (10 runs) with respect to the number of function evaluations (NFEs) on 3 problem instances (PIs).



(a) PI-1 (b) PI-2 (c) PI-3

Fig. 4.9 The change of mean IGD (10 runs) with respect to the number of function evaluations (NFEs) on 3 problem instances (PIs).

Further, the scale of mean HV and mean IGD for PI-3 is lower compared to the scale observed in PI-1 (Fig. 4.8 and 4.9). This is because the reference shapes in PI-3 are from similar clusters, signifying less Euclidean distance between the reference shapes in the latent subspace of the trained PC-VAE. Hence, the optimization algorithms generated a better set of diverse non-dominated solutions for PI-3 compared to PI-1, where the reference shapes are from two different clusters signifying a high Euclidean distance between the reference shapes. However, for all the PIs, the optimization algorithms can achieve feasible car shapes. Therefore, according to the findings, the experiments show that exploiting the data-driven knowledge of latent space improves the convergence of the multi-objective optimization, leading to the generation of diverse 3D shapes.

4.1.5 Optimization for generating solutions based on DM preference

To generate solution(s) of interest that satisfy DM's design preference in the second optimization scenario (Fig. 4.1b), three preference scenarios (ρ=2, 1 and 0.5) are considered based on the distance (d_1 and d_2) values provided by the DM. The normalized preference $\bar{\rho}$ is calculated from the distance values (Eq. 4.4). Next, the normalized ratio is incorporated in both the WSM and the baseline approach. The ideal weight in the WSM needs to be estimated (Eq. 4.6) based on each normalized preference ($\bar{\rho}$). In the baseline approach, the DM specified preference ratio $\bar{\rho}$ is considered as an additional objective function in the multi-objective problem (Eq. 4.3).

Parameterization of WSM: The weight in the WSM is estimated using the proposed adaptive weight determination method (Fig. 4.4). First, the composite function (Eq. 4.5) is minimized for a single w_1 to generate a final solution, whose distance to both the objectives is denoted by $\bar{\Psi}_1(\vec{z})$ and $\bar{\Psi}_2(\vec{z})$, respectively. Second, an additional global optimization is performed such that the final solution from the WSM matches the normalized preference criteria ($\bar{\rho}$). These two steps are performed in an iterative manner, where an initial bound

of the weight $w_1 \in [0, 1]$ is provided at the start of each optimization process. Through the two-step iterative process, the global optimizer tries to find the global minimum (optimal w_1) and the process continues until the cost function (Eq. 4.6) converges. The final solution generated with this optimal weight w_1 in WSM is closer or matches the DM's design preference. This adaptive strategy is performed for each preference scenario ($\rho = 2, 1$ and 0.5) and for all the 3 PIs.

Performance Metrics: For the comparative study, three unary metrics are considered:

- *Number of function evaluations (NFEs)* for the proposed weight adaptive strategy depends on the number of calculations of the objective functions. For the WSM (Fig. 4.4), the weight w_1 is searched using a global optimizer. Each iteration of the optimization loop needs 500 function evaluations for performing the CMA-ES optimization. Next, modifying the initial solution of CMA-ES ($\vec{z_0}$) needs 2000 NFEs. Therefore, the total NFEs for the proposed approach of determining weights w_i for WSM is $(2000 + n \times 500)$ NFEs. For the second baseline MOEA algorithm (OMOPSO), the total number of function evaluations exhausted is equal to the product of the number of particles (100) and the number of generations (100).

- *Preference ratio* measures the ratio of the distance in $\bar{\Psi}_1$ to $\bar{\Psi}_2$ of the final generated solution.

- *Quality* of the final generated solutions is measured by the Euclidean distance between the generated solution and the preferred solution (z_{P_ρ} in Fig. 4.3). The lower the Euclidean distance, the closer the final generated solution to the DM's preferred solution.

Results and Discussion

The performance of both the WSM and baseline 3-objective MOEA are assessed (Table 4.3), in terms of the above-mentioned three unary metrics. For each preference scenario (ρ= 2, 1, and 0.5), an optimization task is performed for all the 3 PIs (Sec. 4.1.3), using both WSM and 3-objective MOEA. The optimization using the 3-objective MOEA generates more than one final solution, so one or two final solutions are chosen that are closer to the normalized preference ratio ($\bar{\rho}$). For 3-objective MOEA, the NFEs are constant for each optimization task.

For all PIs and normalized preference ratios ($\bar{\rho}$) (Table 4.3), both optimization methods converge and generate feasible car shapes. However, the WSM method with a global optimizer generates a shape that is closer (lower quality metrics) to the DM-defined preference, along with much fewer NFEs. The WSM generates a different optimal weight w_1 depending on each PI and normalized preference ratio ($\bar{\rho}$).

Table 4.3 Metric comparisons (NFEs, preference ratio, and quality) for generating solution (s) of interest using WSM with the global optimizer and 3-objective MOEA. Each of the problem instances (PI) is evaluated for each of the preference scenarios. For WSM, we reported as NFEs (n), where n is the number of iterations through the optimization loop. Method with best performance metrics for each preference ratio are shown in bold.

Methods	**Metrics**	**Preference ratio (ρ = 2)**			**Preference ratio (ρ = 1)**			**Preference ratio (ρ = 0.5)**		
		PI 1 $\bar{\rho}$ =7.94	PI 2 $\bar{\rho}$ = 3.62	PI 3 $\bar{\rho}$=8.66	PI 1 $\bar{\rho}$=1.6	PI 2 $\bar{\rho}$=0.91	PI 3 $\bar{\rho}$=1.06	PI 1 $\bar{\rho}$=0.56	PI 2 $\bar{\rho}$=0.22	PI 3 $\bar{\rho}$=0.085
Search w_1 with global optimizer in WSM	NFEs (n)	**4000 (4)**	**6000 (8)**	**4000 (4)**	**5000 (6)**	**3500 (3)**	**4000 (4)**	**5500 (7)**	**5000 (6)**	**5000 (6)**
	Normalized preference ratio ($\bar{\rho}$)	7.96	3.48	7.72	1.10	0.92	1.09	0.48	0.20	0.095
	Quality	**0.019**	**0.085**	**0.089**	**0.022**	**0.003**	**0.036**	**0.045**	**0.054**	**0.050**
3-objective MOEA	NFEs	10000	10000	10000	10000	10000	10000	10000	10000	10000
	Normalized preference ratio ($\bar{\rho}$)	6.30	3.92, 3.35	5.30	1.10, 1.03	1.06	1.04, 0.95	0.56, 0.64	0.24	0.64
	Quality	0.114	0.194, 0.139	0.10	0.121, 0.120	0.054	0.24, 0.19	0.28, 0.27	0.16	0.40

This experiment shows that the proposed strategy (Fig. 4.4) adapted the initial bound $[0, 1]$ of weight (w_1) to an optimal weight value (optimal w_1), and the final solution of WSM with this optimal w_1 is the closest to the DM's preference criteria. Hence, it can be concluded that using the latent representation of the PC-VAE as decision variables for the preference-based optimizations helps in faster generation of solution.

4.1.6 Summary

In this section, the feasibility of the latent representation of the point cloud variational autoencoder (PC-VAE) is evaluated as a geometric representation in a set of multi-objective optimization tasks. The PC-VAE trained with car shapes from the ShapeNetCore repository, generates a suitable probabilistic latent manifold, which is used as an optimization landscape for preference-based optimization. Knowledge from the latent space is exploited to generate solutions involving multiple criteria and is incorporated in the optimization algorithm to achieve faster convergence. However, exploiting the knowledge of the learned latent representation requires an existing data set for training the PC-VAE, which is not always available prior to the optimization task. Additionally, defining the data set is already a challenging task, since it should contain the complete set of geometric features that the optimization search should be able to reach during the optimization.

For the multi-objective optimization task, two preference incorporation scenarios are considered in this research. The first scenario resembles designers having no prior information about their design preferences. Therefore, the aim of the optimization task is to generate a diverse range of design solutions that are given back as a suggestion to the designer to select a design of interest. A seeding strategy (*Lerp-seed*) is proposed by exploiting latent space knowledge, to initialize a multi-objective evolutionary algorithm.

This proposed seeding strategy provides a warm start to the optimization process, which in turn improves the performance of the evolutionary algorithm for solving the optimization problem. Through a series of experiments, it has been demonstrated that using the proposed seeding strategy is beneficial not only to improve the convergence time of the algorithm and generate better diverse solutions compared to using algorithms with a random seeding approach.

The second scenario corresponds to designers having prior preference information and therefore the aim to generate a final solution that matches the preference criteria. The WSM method is considered for this approach, where the algorithm is initialized with solutions from the *Lerp-seed*. Next, an adaptive strategy is proposed to determine the weight in WSM, such that the final solution of the optimization matches the designer's preference. Comparing the result with other baseline approaches, WSM can generate a final solution that matches the designer's prior mentioned preference and with less NFEs.

Therefore, with the series of optimization experiments, the continuity of the latent representation of the PC-VAE is verified for multi-criteria-based design generations. However, an ideal automotive design optimization involves the calculation of several design properties such as aerodynamic drag efficiency, crash-worthiness etc. Optimizing each design based on these criteria requires expensive, time-consuming simulation runs. Therefore, to replace costly objective function estimations with surrogate models, the research in the next section focuses on analyzing the suitability of the latent representation for surrogate modeling tasks.

4.2 Exploiting Latent Space for Surrogate Modeling

In automotive digital development, computational fluid dynamics (CFD) analysis helps to optimize car designs for increased efficiency and better aerodynamics, however, each CFD simulation is computationally expensive. A feasible way of replacing expensive simulations or function evaluations in design optimizations is to learn faster-to-evaluate surrogate models from past CFD evaluations and employ them for estimating objective functions in an optimization task [113, 114]. Nevertheless, generating an accurate data-driven surrogate model for optimization requires enough data so that the surrogate model can sufficiently approximate the original fitness landscape of the optimization. Though benchmark data sets of 3D shapes are available, data sets that are geared towards engineering applications and engineering-specific problems are lacking due to generation cost and confidentiality reasons. Therefore, in the engineering domain, it is difficult to apply state-of-the-art machine learning techniques for surrogate modeling. An example of such an application is the prediction of CFD from geometric representations of 3D designs, which poses challenges such as highly non-linear data and relationships between performance and

geometry. Furthermore, a suitable representation of geometry must be found that can be used as input to a predictive surrogate model. One possible idea is to utilize the latent representation of the point cloud (variational) autoencoder (PC-(V)AE) as the input representation for training a surrogate model. However, training a PC-(V)AE neglects any explicit information related to the performance of the designs. Therefore, the relation between the learned latent variables and the design performances are also unknown. Hence, configuring the surrogate models to map highly nonlinear data, such as aerodynamic performances, becomes very challenging.

In this section, information-theoretic measures are utilized to quantify the information stored in the latent space of the PC-(V)AE with respect to different structural and functional properties of the 3D shapes. Next, a surrogate model is trained on the learned latent representation for performance prediction of 3D shapes. Further, to address the data-limitation issue of engineering performance measures, the probabilistic latent space of PC-VAE is exploited for generating additional realistic car shapes to augment the surrogate model's training data.

The remaining sections are organized as follows: In Section 4.2.1, the existing surrogate models built on the latent space of autoencoders are reviewed. Next, Section 4.2.2 provides details of the PC-(V)AE's architecture, surrogate model's architecture and the data augmentation process, along with the detail of the experimental set-up (Section 4.2.3). The information theoretic analysis of the PC-(V)AEs latent variables for regression tasks is presented in Section 4.2.4 and the performance of the surrogate models for the regression task are analyzed (Section 4.2.5). Lastly, in Section 4.2.6, the latent space of the PC-VAE is exploited for data-augmentation task to address data-limitation for surrogate model training.

4.2.1 Prior Art

Unsupervised Learning of 3D Representations: Both in the autoencoder and variational autoencoder, the encoder processes unstructured 3D data through point-wise operations to generate the latent representation, which primarily holds local design features. Even if the latent representations of both the PC-AE and PC-VAE provide a high-level summary of the input representations [115], a quantitative analysis of the information stored in the latent representations of these models is currently missing.

Training Surrogate Models on Latent Representations: Previous work in the chemistry and biology domains shows promising results with surrogate models trained on latent representations to predict the characteristic of chemical compounds before committing to an extensive synthesis process [15, 116]. In the automotive domain, few studies use analogous representations to predict aerodynamic forces for real-world applications.

In 2D laminar flow estimation, Eismann et al. [117] utilized a VAE for learning latent representations of 2D shapes and Gaussian process regression to predict the corresponding drag coefficients. For 3D shapes, Umetani et al. [118] utilized a machine learning-based regression model to predict aerodynamic forces of 3D shapes, along with a time-averaged velocity field around the object. Nevertheless, the proposed model is limited to 3D shapes parameterized using Poly Cube maps, which assume a pre-determined ordering of the points. Hence, as the number of network parameters increases with the size of the input point cloud, the network is prone to scalability issues. Different from prior research, the encoder of the PC-(V)AE processes unstructured 3D data through point-wise operations to generate latent representation and the network is easily scalable to high-dimensional point clouds.

However, in our application, the data available for training the surrogate model is limited to a subset of the available shapes. Previous works address this data limitation problem by utilizing the generative capability of VAEs for synthesizing data to tackle imbalanced data sets [66] and improve classification accuracy [68]. In contrast to classification tasks, the labels in the regression task are in continuous space, so associating labels to each of the generated samples is a challenging task. Therefore, to address the data limitation problem in the surrogate modeling task, the PC-VAE is trained on a data set of benchmark car shapes are utilized to generate additional realistic car shapes to augment the surrogate model's training data. Yet recovering the performance labels from 3D representations in the latent space of the PC-VAE is a more demanding task than in the reviewed cases.

To consider surrogate models for regression tasks, the multi-layer perceptrons (MLPs) are a popular neural network architecture [119] due to its structural flexibility. An MLP maps input to output data through input, one or more hidden, and output layers. The layers comprise artificial neurons activated with nonlinear functions and fully connected to others in successive layers. An MLP learns the mapping by adjusting the connection weights between neurons by using the back-propagation algorithm [120] for minimizing the error between the MLP predictions and expected output values. In this research, a multi-output MLP is utilized to map the latent representations with their corresponding performance metrics.

4.2.2 Methodology

The data utilized in this set of experiments comprise 3D point clouds sampled from polygonal meshes of the car class of the ShapeNetCore [18] repository. Different from prior experiments, a shrink-wrapping algorithm [121] is utilized to sample point clouds from 3D meshes. The shrink-wrapping samples points uniformly from the external surfaces of 3D objects, which are most relevant for our applications, as it generates point clouds organized based on the vertex assignment of the shrink-wrapped mesh. The values of the

algorithm are set to six shrinking steps and a single smoothing step (Fig. 4.10). The car class consisted of 3750 shapes and each shape consists of 24578 points ($N = 24578$).

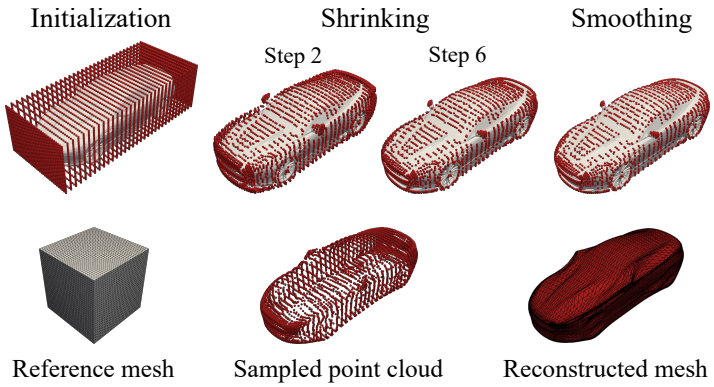


Fig. 4.10 Shrink-wrapping.

Training PC-(V)AEs on organized point clouds has two main advantages. First, since the point correspondence between point clouds is known, it allows us to utilize simpler loss functions, here: mean-squared distance, which are computationally less expensive. Second, by enforcing the autoencoder to learn the ordering of the vertices in the shrink-wrapped mesh, it allows us to utilize the PC-(V)AE predicted point cloud to update the vertices of the mesh and quickly generate water-tight models for engineering simulations.

Design Performance Data: In this research, performances of a 3D design is defined by its volume and aerodynamic drag coefficient. The volume (V) measure is selected due to its low computational effort and simple interpretation. The aerodynamic drag (F_x) requires higher computational effort to compute and is highly nonlinear with respect to the latent representations. Hence, finding suitable surrogate models for aerodynamic data is highly of interest, as it would accelerate design optimization in real-world automotive development [122].

For calculating the aerodynamic drag, CFD simulations are performed on the shrink-wrapped meshes using OpenFOAM®[1]. An assumption is made that the cars drive at a constant velocity of 110 km/h in a straight line and that the cars are symmetric with respect to the vertical-longitudinal (xz) plane. The convergence of each model is verified based on the drag force calculated in the last 20 simulation steps, and the models are selected with a standard deviation within 5% of the mean value and magnitude within $[75, 500]$. Out of 3500 shapes in the original data set, the performance metrics are obtained for 600 shapes only. This simulation database is prepared by co-author Rios [123].

[1] https://www.openfoam.com/

Models for Learning on 3D Point Clouds

To analyze the effect of regularization in the latent space of the variational autoencoder in terms of information content in the latent space, both the autoencoder (PC-AE) and variational autoencoder (PC-VAE) are trained and analyzed in this experiment.

Point Cloud Autoencoder (PC-AE): The PC-AE architecture used in this set of experiments, is similar to the PC-AE proposed in [61].

Point Cloud Variational Autoencoder (PC-VAE): The architecture of the PC-VAE is like the one proposed in the previous chapter (Fig. 3.1). In the previous experiments, the reconstruction loss $\mathcal{L}_{recon}$ in Eq. 3.3 was measured using Chamfer Distance (CD) (Eq. 3.4). Since the CD function is invariant with the permutation of the points, the network trained with CD generates unorganized point clouds. In the present research, the data set consists of organized point clouds. Hence, a second PC-AE and PC-VAE are trained with the mean-squared distance (MSD) as a loss function (Eq. 4.7) to maintain the correspondence between the input and output point clouds and thus generate an organized output point cloud. The MSD between input and output point clouds, defined as,

$$MSD(x_{in} - x_o) = \frac{1}{N} \sum_{j=1}^{N} \|x_{in,j} - x_{o,j}\|_2^2 \tag{4.7}$$

where $x_{in,j}$ and $x_{o,j}$ are the j-th points of the input and output point clouds respectively.

Furthermore, as the VAE learns a stochastic mapping between the input space and the latent space, $\vec{z} = \vec{\mu}$ (Fig. 3.1) is considered as the latent representation of the PC-VAE to get a fixed set of latent variables for the experimental analysis in the following sections.

Information Theoretic Measures to Analyze the Latent Representations

To analyze the relationships between the latent representations learned by the PC-(V)AEs and the design performance metrics, the Pearson correlation coefficient (PCC) and mutual information (MI) are calculated. Both measures quantify the relationship between a pair of variables, X, and Y. The $PCC(X, Y)$ assumes a linear relation between the variables, while the MI measures the amount of information that X holds about Y as the KL divergence between the joint distribution $p(X, Y)$ and the product between the marginal distributions $p(X), p(Y)$. Hence, MI also describes nonlinear relations in the data, which are not captured by the PCC. Kraskov et al. [124] proposed an MI estimator for continuous input variables, which is used here. Both the PCC and MI are used to quantify relationships between variables, as the PCC may miss non-linear relationships, while it is computationally less expensive to estimate from the data.

Note, that MI estimators have known bias when applying them to finite data [124]. A common approach for handling this bias is by performing statistical testing of the MI estimate, to evaluate whether the estimated MI is truly different from zero. For this, a permutation testing is done under the Null hypothesis of zero MI. The Null distribution is generated by repeatedly estimating the MI from shuffled data such as to generate a null distribution, against which the original estimate is compared.

Surrogate Model

For training a surrogate model from the latent representations of 3D designs to their desired properties, a 2-output multi-layer perceptron (MLP) is trained to map a latent representation $\vec{z}_i$ to their corresponding two output variables (Fig. 4.11): normalized volume (V) and normalized drag coefficient (F_x). The multi-output MLP is trained for a regression task, i.e., given a latent vector $\vec{z}_i$ of an unseen 3D shape instance $x_i, i = 1, 2 \subset \mathbb{R}^3$ (Fig. 4.11), the learned multi-output regression function $f(\cdot)$ predicts a two-dimensional vector y as output, predicting V and F_x.

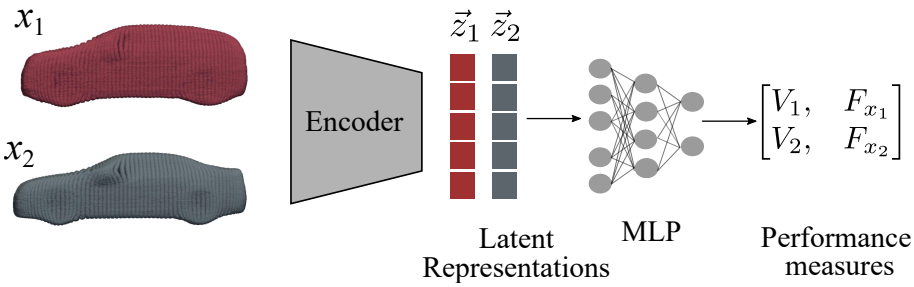


Fig. 4.11 Schematic overview of surrogate model (MLP) training procedure to map latent representations to performance measures (V, F_x).

Here, the MLP comprises a single hidden layer with 10 hidden neurons, which are activated with ReLU. The output layer comprises two neurons that yield the predicted values of volume (V) and drag coefficient (F_x), and is activated with a sigmoid function. The data set is divided into partitions of 85% and 15% for training and testing, respectively. A 5-fold cross-validation strategy is applied on the training set to tune the number of hidden neurons, and the learning rate of the MLP. The weights of the network are optimized by minimizing the mean-squared error for a maximum number of 1000 epochs utilizing the Adam optimizer [76]. The learning rate of the optimization algorithm is set to $\eta = 0.001$ and the data is fed to the network in a batch of 50 samples.

After selecting the hyper-parameters of the MLP, the MLP is re-trained on the whole training set and the performance of the surrogate model is evaluated on the test set using R-square (R^2), root-mean-square error (RMSE), and mean absolute error (MAE). R^2 score measures the squared correlation between the true and predicted values of the surrogate

model, while RMSE and MAE measure the prediction error of the model. Therefore, the higher the R^2 score and lower the RMSE and MAE, the better the surrogate model.

Data Augmentation

The available performance measure data for training the surrogate model is limited in our case. More precisely, performance data is sparse in certain regions of the latent space. This imbalance in the data set potentially leads the MLP to over-fit the data corresponding to the most abundant target values or miss the input-output relations of less observable samples. Therefore, to address this issue, the PC-VAE trained with MSD loss is utilized to extrapolate new designs based on the distribution $(\vec{\mu}, \vec{\sigma})$ learned in the sparse regions.

The advantage of using the PC-VAE is that the stochastic latent space allows us to quickly generate new designs with shared geometric features (Fig. 4.12), which is challenging to perform by directly manipulating the ShapeNetCore models. The sampled latent vectors are fed to the trained decoder of the PC-VAE-MSD. Note, the PC-VAE trained on MSD is used for this shape generation task, as the aim is to generate water-tight meshes from the output point clouds, which is a requirement for CFD simulations.

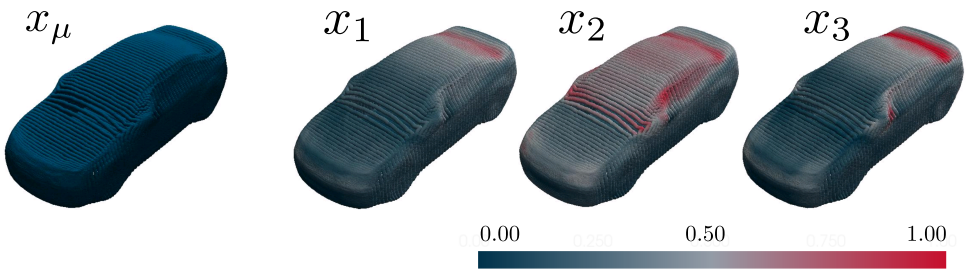


Fig. 4.12 Reconstruction of the sampled shapes from a latent distribution. x_μ is the shape reconstructed by decoding the $\vec{\mu}$ vector of the distribution. The colors of shapes (x_1, x_2, x_3) indicate difference of the shape from x_μ.

4.2.3 Experimental Setup

Considering the discussed methods, an experimental setup is presented for training the PC-AE and PC-VAE architectures with different training loss functions.

Data Set: The data set for this set of experiments comprises 3500 shapes selected from the car class of the ShapeNetCore repository [18] (Fig. 4.10). The networks are trained with randomly selected 90% percent of the shapes in the data set.

Training Settings for PC-AE: Two PC-AE models are considered: The first (PC-AE-CD) was trained utilizing the Chamfer Distance (CD) as reconstruction loss, while the second (PC-AE-MSD) was trained using MSD as loss function. The hyper-parameters of

the networks are set to standard values [61]. The size of the latent representation is set to $L_z = 20$, to balance the trade-off between low reconstruction loss and a limited number of latent variables for qualitative evaluations.

Training Setting for PC-VAE: Two variations of PC-VAE models are considered similar to PC-AEs. Our motivation is to allow the PC-VAE to reconstruct an organized point cloud for easy generation of water-tight meshes. Hence, the hyper-parameters of the PC-VAE-MSD model are optimized using a grid search. The selected values for hyper-parameters α and β are 1000 and 0.001 respectively, as they yield an acceptable trade-off between reconstruction accuracy and divergence in the latent space. For training PC-VAE-CD, the hyper-parameters of the network are set to standard values [77].

All remaining parameters and hardware for training PC-(V)AEs are the same as utilized in the verification experiments (Sec. 3.3.1).

4.2.4 Information-Theoretic Analysis of the Latent Representations

The Pearson correlation coefficient (PCC) and mutual information (MI) are calculated between the latent representations learned by each model (PC-AE-CD, PC-AE-MSD, PC-VAE-CD, and PC-VAE-MSD) and two performance metrics: normalized volume (V) and normalized drag coefficient (F_x) (Figs. 4.13 and 4.14). In this analysis, a sub-set of data comprising 600 shapes is considered for which the CFD calculation converged.

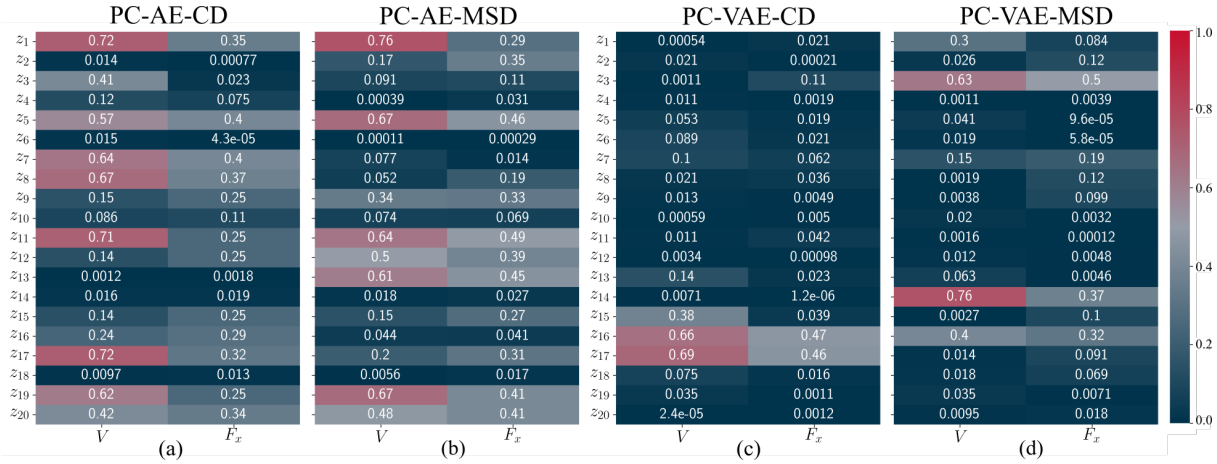


Fig. 4.13 Pearson Correlation between the latent representations of the autoencoders (PC-AE-CD, PC-AE-MSD) and variational autoencoders (PC-VAE-CD and PC-VAE-MSD) and normalized performance metrics calculated from the designs: volume (V) and drag (F_x).

The absolute values of PCC and MI obtained for the PC-AE models were higher than observed for the PC-VAE models. A higher PCC magnitude indicates a lower complexity for surrogate models to learn the data since the relations between input and output have

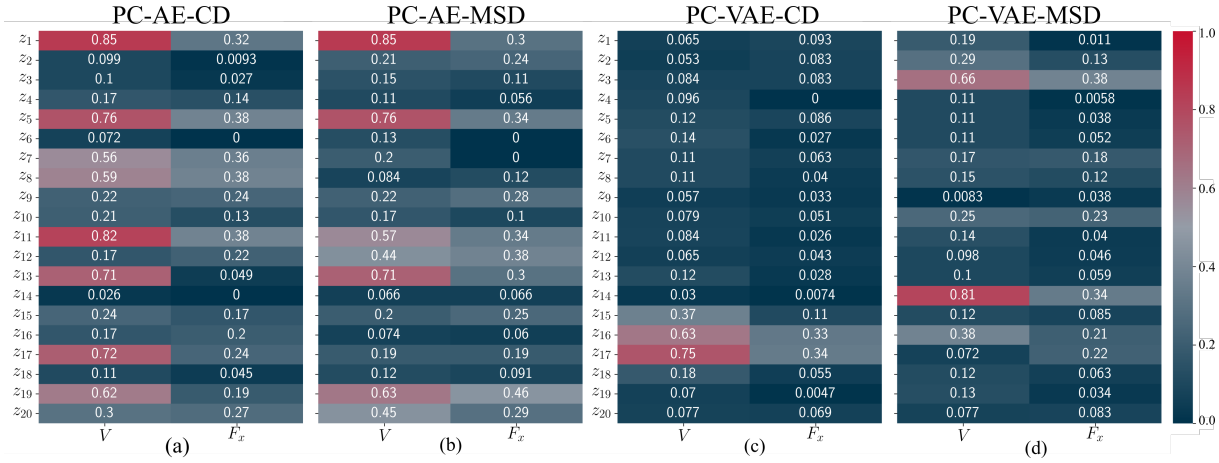


Fig. 4.14 Mutual Information between the latent representations of the autoencoders (PC-AE-CD, PC-AE-MSD) and variational autoencoders (PC-VAE-CD and PC-VAE-MSD) and normalized performance metrics calculated from the designs: volume (V) and drag coefficient (F_x).

a higher degree of linearity. The MI indicates that the latent representations of the PC-AE hold more information about the design performances than the PC-VAE variants. Regularization in the latent space of the PC-VAE reduces the information content with the probabilistic encoding. Hence, regardless of the data linearity, the PC-AE latent space potentially leads to more accurate surrogate models since it allows the surrogate model to leverage more information for learning the performance measures.

Also, the values of PCC and MI were more evenly distributed in the PC-AE latent spaces than obtained with the PC-VAE. This difference indicates that the PC-AEs enable different design degrees of freedom than the PC-VAEs and are in line with the results reported in [121]. Furthermore, comparing the obtained MI between models trained with CD and MSD (Fig. 4.14, columns (b) and (d)), the models trained with MSD yielded higher values. Our interpretation is that the MSD allows the PC-(V)AEs to implicitly learn global shape information by enforcing the organization of the points and, thus, improving the correlation of the latent representations to more complex performance data.

To test the statistical significance of MI estimations in Fig. 4.14, the order of values for each latent variable z_i, is shuffled to obtain permuted values z_i'. Next, a surrogate MI value $I'(z_i', y)$ is calculated. This process was repeated 200 times for each combination of the latent variable z_i and performance value. The p-value for each $I(z_i, y)$ was obtained as a fraction of surrogate values larger than the original estimate, $I'(z_i', y) > I(z_i, y)$. If the p-value was lower than the critical α-level of 0.05, it is considered statistically significant. All MI estimates were statistically significant at the specified α-level. Therefore, MI estimation in the PC-VAE latent space signifies that a small set of variables holds information about the performance, compared to more latent variables with high information in the PC-AE latent space.

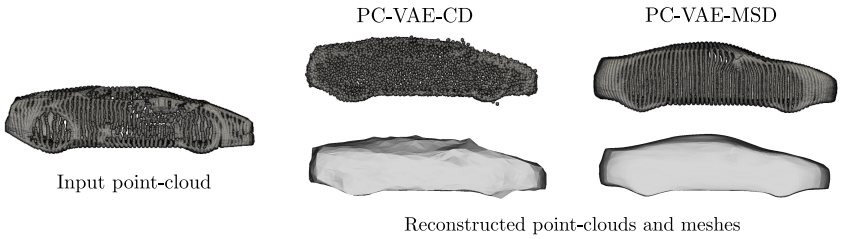


Fig. 4.15 Reconstruction of the input point cloud using the trained PC-VAE-CD and PC-VAE-MSD models.

Nevertheless, from the results, it can be concluded that the baseline MI values (Fig. 4.14) are representative, and our observation that the MSD-based training increases the information contained in the latent space still holds. Furthermore, the PC-VAE trained with MSD loss function shows improved quality of mesh reconstruction, which is verified by visual inspection of the reconstructed shapes in Fig. 4.15.

4.2.5 Surrogate Model

Surrogate models are trained on the latent representations of 3D shapes along with their performance metrics. To build a surrogate model for predicting performance measures, a two-output multi-layer perceptron is considered to map the latent representations with the performance metrics: F_x and V. Two MLPs are trained to predict performance metrics: The first MLP trained on the latent representations of PC-AE-MSD (MLP-AE) and the second MLP trained on the latent representations of PC-VAE-MSD (MLP-VAE). As baselines, two models are trained to predict the performance as the mean performance in the training set for both the PC-AE-MSD (Baseline AE) and the PC-VAE-MSD (Baseline VAE). The data set of 600 shapes along with their performance metrics are divided into 85% and 15% training and test set.

To consider the stochastic nature of the MLPs initialization, the performances on the training and test set are evaluated over 30 runs and the mean and standard deviation of the errors are reported. Table 4.4 shows the prediction performances of MLPs and baseline regression models on the training and held-out test set. Both the MLP-AE and MLP-VAE showed better prediction performances compared to baseline models. The R^2 score, RMSE, and MAE between MLP-AE and MLP-VAE are comparable for the training set. However, in the test set, the MLP-AE predicts the output performances more accurately than the MLP-VAE, which indicates that the latent space learned by the PC-AE is more suitable for surrogate modeling than the space learned by the PC-VAE.

For visual verification of the two performance prediction results, two scatter plots of true versus predicted values for the test samples are shown (Fig. 4.16). Both models predicted the volume with better performance than the aerodynamic drag coefficient, for

Table 4.4

	Training set			Test set		
Models	**R2**	**RMSE**	**MAE**	**R2**	**RMSE**	**MAE**
Baseline AE	-.005±.004	.160±.002	.120±.001	-.007±.000	.119±.000	.156±.000
MLP - AE	**.875±.002**	**.044±.002**	**.035±.004**	**.834±.003**	**.048±.0006**	**.068±.0015**
Baseline VAE	.101±.280	.140±.030	.110±.020	-.008±.000	.110±.000	.157±.000
MLP-VAE	.861±.010	.048±.005	.036±.002	.750±.001	.055±.0023	.072±.0023

which the scatter plot showed higher dispersion of the data with respect to the ideal regression model.

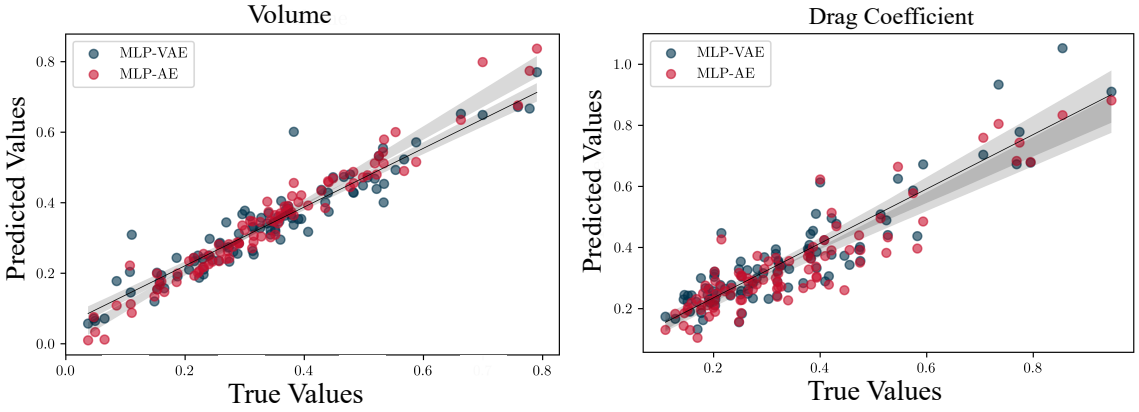


Fig. 4.16 True versus predicted values of drag and volume measures on the test set.

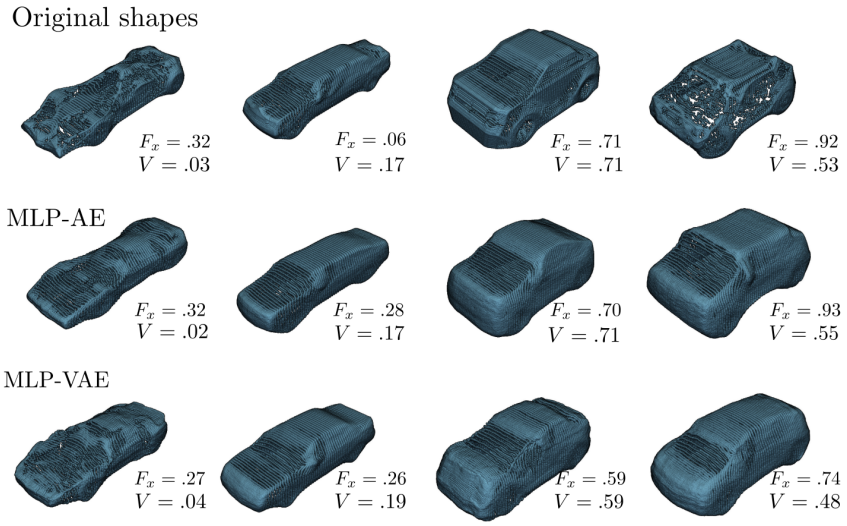


Fig. 4.17 Comparison between the original shapes in the test set with true performance values and reconstruction of the samples using PC-AE-MSD and PC-VAE-MSD with performance values predicted using trained MLP regressors on their latent space.

In the second verification, the reconstructions of 4 samples from the test set along with their predicted performance values are plotted for visual verification (Fig. 4.17). The prediction of the MLP-AE on 4 samples is more accurate compared to MLP-VAE. However,

for the shape with the lowest drag (second column), both models failed to predict the lower drag value (F_x). This could be due to limited data of that particular car class in the training set. Further, analyzing the distribution of the normalized volume and drag of the training samples, it can be concluded that most of the samples yield accurate normalized performance values in the interval $[0.2, 0.6]$ (Fig. 4.17). However, prediction power is poor in the extremes of both performance measures due to data sparsity. Therefore, to improve the distribution of the training data, the shape generative capability of the PC-VAE-MSD is explored to augment the data set and improve the quality of the surrogate models.

4.2.6 Exploiting the Generative Capability of PC-VAE for Data Augmentation

To augment the regression data set, an additional 300 latent vectors are sampled from input shape distributions which has normalized volume and drag values below 0.2 and higher than 0.6. For each vector, the point cloud is reconstructed using the decoder of the PC-VAE-MSD (Section 4.2.2). The reconstructed point clouds are converted into meshes using the shrink-wrapping algorithm, and the volume and drag coefficient of these shapes are calculated using OpenFOAM®.

Table 4.5 RMSE on test set. RMSE marked in bold are statistically significant compared to the baseline RMSE (when $r = 0$).

Ratio of Augmented Data (r)	**MLP-AE**	**MLP-VAE**
0	$0.0480 \pm .0006$	$0.0550 \pm .0023$
0.2	$\mathbf{0.0454 \pm .0011}$	$\mathbf{0.0488 \pm .0014}$
0.4	$\mathbf{0.0447 \pm .0011}$	$\mathbf{0.0492 \pm .0015}$
0.6	$\mathbf{0.0452 \pm .0013}$	$\mathbf{0.0491 \pm .0011}$
0.8	$0.0474 \pm .0012$	$0.0557 \pm .0018$
1	$0.0492 \pm .0017$	$0.0570 \pm .0020$

Starting with the prediction performance on the test set in Table 4.4, the augmented shapes generated by the PC-VAE-MSD are added in an incremental fashion to the existing training set. The ratio r signifies the percentage of generated samples added to the training data of the regression model, which is increased from 0 to 1 by 0.2.

Both MLPs are trained by adding the augmented data set. The testing is performed on the held-out test set for 30 runs and reported the mean and standard deviation of the RMSEs on the test set. This also compared the generalization performance of the surrogate models on the test set by adding the additional augmented data. When the ratio $r = 0$, the regression model is trained only on the original training data, so the RMSE values in Table 4.5 are similar to the ones in Table 4.4.

For ratios 0.2 to 0.6, the added samples increased the diversity of the data, which in general improves the quality of the surrogate models. But for higher $r = 0.8$ or $r = 1$ ratios, the added samples shared a high degree of similarity with the existing data, thus adding redundancies and decreasing the performance of the MLPs. Therefore, from the Table 4.5, it can be concluded that utilizing the augmented dataset for training the surrogate models improved the generalization performance of both the MLP-VAE and MLP-AE since the RMSE values decreased.

Further, to confirm the difference in the RMSE values of each ratio with the RMSE values at $r = 0$, a Mann-Whitney test was performed. The results of the statistically significant analysis are in bold (Table 4.5) (p-values< 0.01). Nevertheless, the experiment shows that the data augmentation using the PC-VAE has the potential to improve the quality of surrogate models, which was our objective.

4.2.7 Summary

In this section, the feasibility of the latent representation is evaluated in terms of information content in the latent variables about structural and functional performance measures of 3D designs. Two challenges for learning latent space-based surrogate models are addressed in this research. First, the autoencoder training data neglects any explicit design performance information. Hence, an initial information theoretic analysis on the latent representations gives a hint of the capability of these models to learn performance-related information. Further, it can be concluded that enforcing the autoencoders to learn organized point clouds increases the amount of information available in the latent space, potentially by providing an implicit description of global design features.

Second, the regularized latent space of the PC-VAE is utilized for fast design prototyping for data augmentation. Since imbalanced training data decreases the quality of surrogate models, the latent distribution learned by the PC-VAE-MSD model is utilized to sample and generate new realistic designs in sparsely sampled regions of the data set. This way of performance-based design generation is faster and more intuitive than direct manipulation of the 3D designs. Lastly, it is shown that within a threshold, the proposed data-augmentation method improved the accuracy and generalization capability of the surrogate models for predicting the aerodynamic performances of car shapes.

4.3 Conclusion

The work in this chapter is based on our previously published research [125, 123]. The objective of these experiments and proposed methods is to improve our understanding of the information learned by the PC-VAE. For that purpose, the latent space is first analyzed as a low-dimensional search space for optimization tasks, where a method is proposed to

provide a "warm start" of the optimization algorithm by exploiting the linear interpolation in the latent space of a trained PC-VAE. Then, in a second series of experiments, the latent features of PC-VAE are evaluated from the data compression and encoding perspective, where information theoretic measures are used to identify the amount of information content in the latent variables of PC-VAE with respect to structural and functional properties of the 3D shapes.

The analyses in this chapter address the research questions *RQ*3 and *RQ*4, which are answered in the following paragraphs.

RQ 3. How can the data-driven latent space be utilized to generate design solutions that satisfy multiple criteria?

To build an experience-based CDS framework (Fig. 1.4), it is beneficial to determine a suitable optimization landscape, which helps in faster search and optimization of 3D designs. In this context, the trained PC-VAE from the previous chapter offers a low-dimensional latent manifold, which can be used as a search space for multi-objective design optimization tasks. Next, to evaluate the suitability of the latent space as an optimization landscape, a set of real-world inspired multi-objective optimization tasks has been set up for incorporating user preferences and generating final design solutions. Two optimization scenarios are considered, where in the first scenario, the DM aims to look at a range of design solutions from the optimization task and then select the final solution. While the second scenario, refers mainly to experienced designers as DM, who have prior design preferences and aim to generate a solution of interest. For the optimization task in both of these scenarios, the latent representations are used as decision variables.

Latent space knowledge is exploited to improve the convergence of optimization algorithms in both the scenarios of the preference-driven optimization tasks. During the *posterior* selection of design solutions, a seeding method is proposed to initialize the optimization algorithm. This provides a faster convergence of the MOEA and improves the diversity of the generated solutions. Hence, the designers can select the solution of interest from the generated diverse design set. The second scenario mainly refers to an experienced designer, who has a design preference prior to the start of the optimization task. The weighted-sum method (WSM) method is suitable in this scenario to generate a particular solution of interest based on the designer's preference. An adaptive strategy is proposed to decide the optimal weights in the WSM to generate a final solution of interest with a lesser number of function evaluations. However, the limitation of the WSM is that this method only works for convex optimization problems.

RQ 4. Does the latent space of the deep generative models hold relevant information about the structural or functional properties of 3D designs? Can a surrogate model be trained on the latent manifold to replace expensive performance simulation or function evaluations in a design optimization problem?

To reduce expensive performance evaluations in design optimizations, one suitable approach is to learn faster to evaluate surrogate models for performance predictions. Further, in the context of the experienced-based CDS framework (Fig. 1.4), training a surrogate model for performance prediction of 3D designs will allow the designers to consider multiple performance measures to finalize a design. To build a surrogate model for performance predictions of 3D designs, the trained PC-(V)AE from the previous experiments offers a low-dimensional latent manifold, which facilitates faster search and optimization tasks. Similarly, the latent representations of the PC-(V)AE are used as input representations for training a surrogate model for this experiment.

The information theoretic measures are utilized to gather hints on the quality of the regressions before training a surrogate model. Since the autoencoder (PC-AE) and the variational autoencoder (PC-VAE) generate latent representations similarly, both the learned latent spaces of these models are evaluated using information theoretic measures. The PC-AE trained with only reconstruction loss shows higher information content compared to the PC-VAE, which is trained with both reconstruction loss and a regularization loss function. Therefore, it can be concluded that even if the regularization loss function helps to generate a probabilistic continuous latent manifold for the PC-VAE, it reduces the performance information content in the latent variables of the PC-VAE. An additional observation is that training both the networks with the MSD loss function enforces them to learn implicit information on the global point cloud structure. This increases the information content in the latent space of both networks (PC-AE-MSD and PC-VAE-MSD).

After the initial evaluation of the latent representation structures with the information theoretic analysis, MLP-based surrogate models are trained on the latent representations of the PC-AE-MSD and PC-VAE-MSD to predict volume and aerodynamic drag coefficients of the 3D shapes. The surrogate model trained on the latent space of PC-AE shows better performance compared to the one trained on the latent space of PC-VAE. This result is in line with the information theoretic analysis, which shows the information content in the latent variables of PC-AE is higher compared to PC-VAE. However, both MLPs performed better in predicting the volume measure of the input shapes compared to the drag coefficient predictions, which could be due to the data-limitation and high non-linearity of the aerodynamic values.

Lastly, to address the data-limitation of sparse designs while training a surrogate model, a method is proposed to address the data limitation in a regression task. The probabilistic latent space of the PC-VAE-MSD is used to generate new realistic designs from sparsely sampled regions of the data set. A set of additional 3D designs are sampled from the latent space of the PC-VAE-MSD model, which are added to the training data of the surrogate model. Samples are selected from the augmented data set on which the aerodynamic performance measures converge. While adding these selected samples to the training set of the surrogate models and retraining the surrogates show improved performance prediction accuracy when there is no redundancy in the augmented data. This also signifies the generalization capability of the surrogate models, which improves with the addition of new, diverse data.

Based on the discussions presented in this chapter, it can be concluded that low-dimensional latent representations of the PC-VAE enable a suitable manifold for training surrogates and optimization of 3D designs. With multi-objective optimization in the latent space of the PC-VAE, it is possible to generate diverse solutions between reference shapes. These shapes provide a wide range of novel solutions, which show gradual change in car designs. Yet, it is hard to generate a novel car shape that combines features from both the reference shapes in a controlled fashion. Therefore, in the following chapter, a novel architecture is proposed, which gives the flexibility to recombine the essence of two designs to generate a novel design.

Chapter 5

A Novel Learning-based Representation for 3D Shape-to-Shape Feature Transfer

In the context of the experience-based design optimization framework (Fig. 1.4), the research in the previous chapter shows that the latent variables of the PC-VAE are suitable for generating diverse car designs considering multiple criteria. However, since the latent variables of PC-(V)AEs are correlated with each other, it is hard to identify which latent variables represent distinct visual features of a 3D design. For example, car designs in the automotive domain can be divided into classes, such as convertibles, utility vehicles, passenger cars, etc. Each of the car design classes posses distinct visual geometric features, such as the shapes from 3 car classes in Fig. 5.1, shows distinct roof structures. While the existing PC-VAE can accurately reconstruct the shapes, yet identifying distinct latent variables that represent unique visual features of 3D shapes from each class are still missing. Therefore, to learn distinct geometric features of 3D shapes from each class, a novel deep learning network is proposed in this chapter: Split-AE. The architecture is build upon PC-(V)AE [77, 61] and is trained to generate two sets of latent variables named as, *content* and *style*, such that each of the disentangled set of latent variables represents different components of the 3D car designs. Each of these two sets of latent variables is shared between shapes from multiple classes and across semantic categories. These disentangled latent variables are beneficial in terms of shape-to-shape feature transfer to generate novel car shapes by combining the essence of existing shapes from different classes.

Prior research on 3D shape-to-shape feature transfer represented each 3D shape with two characteristics [126]: *Content* and *style*. Content refers to the global structure of a 3D shape, and style refers to the distinctive parts of the shape that discriminates it from others with the same content. However, the assumption in this research is that the content estimates whether the underlying structure of the 3D shape is coherent as a whole

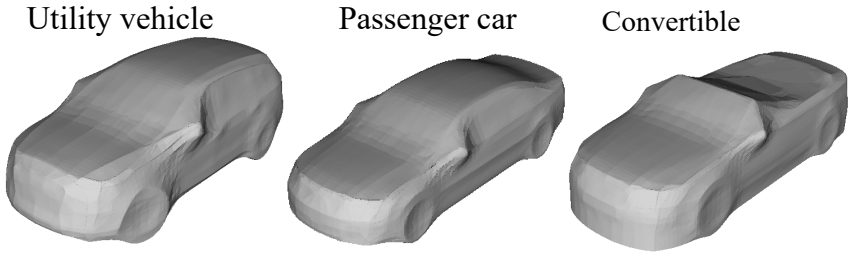


Fig. 5.1 Car shapes from 3 car classes.

and distinguishes across other semantic categories, while the style refers to the localized regions or distinctive parts of the given 3D shape that allows grouping of shapes into shape classes. As an example, in Fig. 5.1, sports utility vehicles (SUV) in the utility vehicle class and sedans in the passenger car class have distinctive shape parts such as unique roof structures, while the convertible car has a removable roof. Hence, this research considers style as the distinct shape parts that differentiate between shapes across shape classes. Further, a 3D shape-to-shape style transfer method is proposed to generate a novel shape by providing a 3D shape as a style reference. For instance, a style transfer from a convertible car to an SUV should generate a novel realistic car shape with a removable roof structure like the convertible car class while having an similar exterior design similar to other parts of the SUV car.

The rest of the chapter is organized as follows: Section 5.1 provides the background of deep generative models for learning 3D shapes and disentangling the latent representation for 3D style transfer. In Section 5.2, the architecture of Split-AE and the pre-processing steps of the data set are detailed. Also, the experimental setup for generating augmented shapes by style transfer and qualitative measures to evaluate the effectiveness of style transfer are described in this section. Section 5.3 describes the experimental settings to train the Split-AE and other baseline networks. Next, the reconstruction and disentanglement quality of the Split-AE is analyzed using quantitative and qualitative measures. These measures analyze whether the content and style features of the trained Split-AE represent different underlying factors of variations in 3D shapes, which hints at the position of style features before performing style transfer (Section 5.4). In Section 5.5, the performance of Split-AE is described with respect to its ability to generate novel augmented shapes by style transfer. Finally, in Section 5.6, a summary and outlook of the chapter are presented.

5.1 Prior Art

Autoencoders for Content and Style Disentanglement of 3D Representations: In the context of unsupervised feature extraction of 3D shapes, state-of-the-art geometric deep learning models such as autoencoders (PC-AEs) [60, 61] and variational autoencoders (PC-VAEs) [77], learn low-dimensional latent representations of 3D shapes, which allow

them to alter 3D shapes by modifying the latent variables. The correlated latent variables of the PC-(V)AE makes it harder for each latent variable to control separate interpretable aspects of a 3D shape. Prior research on disentangling latent variables proposed to divide the latent representation of 3D shapes into two factors [126]: content and style. The authors in [126] directly address the 3D style transfer problem by learning a style-dependent generative model (3DSNet) for shapes from two classes. Given a pair of 3D shapes as input to the 3DSNet, the network can directly generate new shapes by transferring style features between the input shapes, since the network is trained in parallel for reconstruction and shape translation tasks. However, the content space is shared between shapes across two classes, but the style latent space is class-dependent, i.e., for shapes from each class, a separate encoder needs to be trained to generate class-specific style space. Hence, the prior network still lacks a generalized method that addresses style transfer for multiple classes. In contrast, the proposed Split-AE addresses the content-style disentanglement for 3D shapes across multiple classes and domains, by assuming two separate latent spaces [127]: content and style shared between 3D shapes.

Enforcing Disentanglement in Latent Representation: Research on training autoencoder based architectures for multiple tasks [128] relies on a combination of geometric loss and penalties that operate on the latent space to ensure that the space contains the information we wish to encode. Earlier analysis on types of penalties [129] showed that the unsupervised disentanglement of the latent space is fundamentally impossible without minimal supervision or inductive biases on models and data. Hence, some form of implicit supervision in terms of labels of the input data is necessary to achieve disentangled latent variables that represent discriminative features of 3D shapes. To enhance the discrimination of features learned by the latent variables, prior research imposes an additional classification task in terms of cross-entropy loss to train their networks [130, 131].

In this research, to divide and enforce the latent variables for learning the content and style of shapes from different domains (having separate semantic meanings) and multiple classes, two classification tasks are utilized. The first task is to enforce the content space to learn the underlying shared global representations of 3D geometries and distinguish shapes across semantic categories or domains. The second task is to enforce the style space to learn in an unsupervised fashion the distinctive local features of 3D shapes from each shape class, such that it leads to an efficient disentanglement of class-essential and class-redundant information [131]. Further, following the idea of this unsupervised shape-class classification approach to encode distinctive features in style space, helps the Split-AE to learn shapes from multiple classes which is not possible with prior approaches [132, 126].

5.2 Methodology

Given a source shape x_1 and a target shape x_2 as a reference from different classes within the same domain, the goal is to transfer the distinct features or style of x_2 to x_1 ($x_2 \rightarrow x_1$), such that the newly generated shape x_{12} has style features of the target shape and content features similar to the source shape. Therefore, in this section, the proposed approach is described to implicitly learn the separate content and style features of 3D shapes for performing shape-to-shape style transfer.

Section 5.2.1 describes different shape categories and pre-processing steps for sampling point cloud data from surfaces of 3D shapes. Next, the architecture of the proposed Split-AE network is detailed (Section 5.2.2) and the quantitative measures used to evaluate the performance of the network are presented (Section 5.2.3). Lastly, the experimental setup is depicted to generate novel shapes by style transfer using the proposed network (Section 5.2.4).

5.2.1 Pre-Processing of 3D Point Clouds

The shapes are chosen from $\mathcal{K}$ number of domains with different semantic meanings. From each domain, shapes from different classes are considered, such that each class of shapes has distinct geometric features and also possesses common global features between the classes within a domain. The total number of classes is referred to as $\mathcal{X}$ from $\mathcal{K}$ domains. 3D point clouds are sampled from polygonal meshes of $\mathcal{X}$ classes from $\mathcal{K}$ domains, using the shrink-wrapping algorithm utilized in [121]. The shrink-wrapping algorithm is used to sample N points uniformly from the surface of 3D meshes and generate organized point clouds based on the vertex assignment of the shrink-wrapped mesh. Therefore, each shape is a 3D point cloud consisting of N points.

The shapes are chosen from two domains ($\mathcal{K} = 2$), e.g., cars, and airplane domains of the ShapeNetCore data set[18], which is a large data set of 3D shapes (Fig. 5.2). From these two domains, shapes from a total of five classes ($\mathcal{X} = 5$) with their respective class labels $\mathcal{C}_i$ ($i = 1, .., \mathcal{X}$) are considered.

Shapes from three classes in the car domain: SUV, sedan, and convertible, and two classes from the airplane domain: airliners and propellers are selected. The shapes from the above classes are chosen, such that shapes from each class have distinct geometric characteristics that are visually observable, which is referred to as style in this research.

However, the assumptions of content and style for shapes from discrete domains are different. For example, style for shapes from the three classes of the car domain refers to distinctive roof structures, and style for the two shape classes in the airplane domain refers to distinctive wing structures (Fig. 5.2). Therefore, in the next section, the architecture of the proposed Split-AE is introduced to learn the style of these shapes.

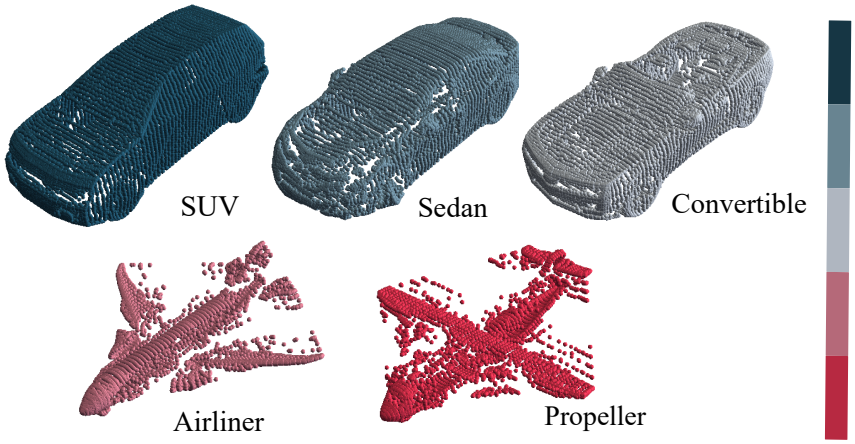


Fig. 5.2 Visualization of one shape from each of the 5 classes from the car and airplane domains. The scale shows distinct colors to represent class labels $\mathcal{C}_i$ $(i = 1, .., 5)$. Each shape is color-coded based on their class label.

5.2.2 Split-AE Architecture

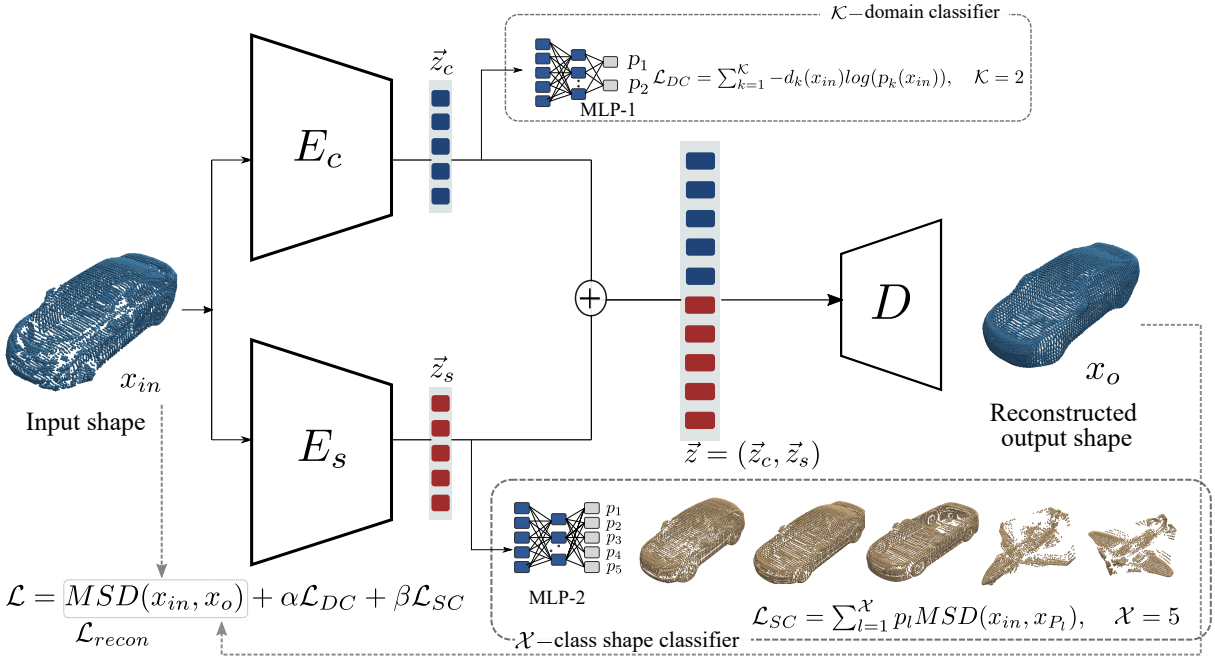


Fig. 5.3 Split-AE architecture and data flow for training the network.

The Split-AE architecture consists of two encoders (E_c and E_s) for mapping input shapes to two latent space branches: the content (Z_c) and style (Z_s) space (Fig. 5.3). Both encoder networks follow the architecture proposed in [61] with five 1D convolutional layers. The first four layers are activated with the rectified linear unit (ReLU) and the last layer with a hyperbolic tangent function. After the fifth convolution layer, a max-pooling operator reduces the dimensionality of the input shape to two L_z-dimensional vectors: Content vector $\vec{z}_c$ and style vector $\vec{z}_s$. Next, these two latent vectors, $\vec{z}_c$ and $\vec{z}_s$, are given

as input to two multi-layer perceptrons (MLPs): MLP-1 and MLP-2, respectively. In the first content space Z_c branch, the MLP-1 classifies the input shape into one of the $\mathcal{K}$ domain categories. The architecture of the MLP-1 consists of two layers, where the first layer with 10 hidden neurons is activated with ReLU, and the last layer with $\mathcal{K}$ hidden neurons is activated with a soft-max function. The index of the selected domain of the input shape corresponds to the index of the neuron with maximum soft-max activation.

In the second style space Z_s branch, the MLP-2 classifies the input shape into one of the $\mathcal{X}$ number of classes. The architecture of the MLP-2 is similar to the MLP-1 in the first branch. But in the output layer of the MLP-2, there are $\mathcal{X}$ number of neurons based on the total number of shape classes, and the index of the selected class corresponds to the index of the neuron with maximum softmax activation. Next, the 2 vectors $\vec{z_c}$ and $\vec{z_s}$ are padded to form a $2 * L_z$ dimensional vector $\vec{z}$, which is given as input to the decoder (D). The decoder comprises three fully connected layers, with only the first two layers activated by ReLU.

Loss Functions: The loss function ($\mathcal{L}$) for training the Split-AE architecture (Fig. 5.3) consists of 3 terms as follows:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{recon} + \alpha \mathcal{L}_{DC} + \beta \mathcal{L}_{SC} \\ \mathcal{L}_{recon} &= MSD(x_{in}, x_o) = \frac{1}{N} \sum_{j=1}^{N} \|x_{in,j} - x_{o,j}\|_2^2 \\ \mathcal{L}_{DC} &= \sum_{k=1}^{\mathcal{K}} -d_k(x_{in}) log(p_k(x_{in})), \quad \mathcal{K} = 2 \\ \mathcal{L}_{SC} &= \sum_{l=1}^{\mathcal{X}} p_l MSD(x_{in}, x_{P_l}), \quad \mathcal{X} = 5 \end{aligned} \tag{5.1}$$

The first term computes the reconstruction error ($\mathcal{L}_{recon}$) between the input point cloud ($x_{i,j}$) and output reconstructed point cloud ($x_{o,j}$), with N number of points in each point-cloud. Since the network is trained on organized point clouds, i.e., the point-correspondence between the point clouds is known, it allows the utilization of a simple loss function, here: mean-square distance (MSD) [121, 123].

The second term ($\mathcal{L}_{DC}$) in Eq. 5.1 obtains a partition between shapes from $\mathcal{K}$ different domains in the content space Z_c using cross-entropy loss (Eq. 5.1), where each input point cloud (x_i) has a ground truth domain label d_k vector. The domain prediction probability vector of the input shape (x_i) is given by p_k.

The third term ($\mathcal{L}_{SC}$) in the loss function in (Eq. 5.1) tries to obtain class-wise partitions of the data into $\mathcal{X}$ number of classes by computing a weighted average of the MSD between the $\mathcal{X}$ chosen prototypes, one from each class and the input shape. Each

prototype defines the most representative 3D point cloud shape from each class, which is selected by calculating the mean point cloud of shapes from each class. In Fig. 5.3, the selected prototypes x_{P_i} $(i = 1, .., \mathcal{X})$ are shown for $\mathcal{X}$ classes, considering $\mathcal{X} = 5$. The weight for each prototype is defined by the corresponding probability output p_i $(i = 1, .., 5)$ of the MLP-2 classifier. Thus, the third term is minimized when the MLP-2 yields the highest selection probability p_i to the prototype with the lowest MSD value (the highest similarity). The shape-classifier tries to obtain a class-wise partition of the data in the style space Z_s.

All terms of the loss function (Eq. 5.1) differ by several orders of magnitude. To balance these differences, two hyper-parameters α and β are introduced to scale the classification losses, respectively.

5.2.3 Metrics for Evaluating the Trained Split-AE

For comparing the performance of the trained Split-AE with the baseline models, two quantitative metrics are considered for estimating the reconstruction capability and style-class classification accuracy of the three latent spaces with respect to the shapes in the unseen test set.

Reconstruction Error: The Chamfer distance (CD) [60] is utilized to calculate the reconstruction error between the input and the generated output point clouds.

Style Class Classification Accuracy: Three random forest-based multi-class classification models are trained to map the latent representations ($\vec{z}_c$, $\vec{z}_s$ and $\vec{z}$) of the training set to their respective class labels $\mathcal{C}_i$ $(i = 1, .., 5)$. The goal of the classification is to take any latent representation of the test set samples as input and assign it to any one of the $\mathcal{X}$ style classes and measure the prediction accuracy of the classifier. This enables us to check for disentanglement of latent variables based on style classes.

Further, for qualitative evaluation of the features learned by the two latent spaces, the feature visualization approach [133] is utilized for visual inspections of regions of the input space mapped by the latent variables.

5.2.4 Augmented Shape Generation by Style Transfer

The trained Split-AE generates a latent vector $\vec{z}_i$ for each input shape x_i, where each $\vec{z}_i$ consists of two parts: Content $\vec{z}_{c_i}$ and style $\vec{z}_{s_i}$ code. Given two shapes, one source shape x_1 from the $\mathcal{X}_1$ class and another target shape x_2 from the $\mathcal{X}_2$ class within the same domain, the aim is to evaluate the effectiveness of the style transfer from shape $x_2 \rightarrow x_1$. Both of the source and target shapes are represented with content and style codes, such as, $\vec{z}_{s1}, \vec{z}_{c1}$ $(x_1 \sim \vec{z}_1 = (\vec{z}_{c1}, \vec{z}_{s1}))$ and $\vec{z}_{s2}$, $\vec{z}_{c2}$ $(x_2 \sim \vec{z}_2 = (\vec{z}_{c2}, \vec{z}_{s2}))$, respectively.

Our goal of transferring the style code $\vec{z}_{s2}$ from $x_2 \rightarrow x_1$ results in generating an augmented shape x_{12} ($\vec{z}_{12} = (\vec{z}_{c1}, \vec{z}_{s2})$), such that x_{12} is perceptually more similar to the target x_2 than the source x_1. Specifically, the augmented shape x_{12} should possess distinct traits of shape x_2 such that it belongs to the target car class $\mathcal{X}_2$.

Therefore, to evaluate the success of style transfer between source-target pairs of each domain, firstly, the distance similarity of the generated shapes with the sour-target pair is measured and secondly, the classification accuracy of how many generated shapes belong to their target shape class using the pre-trained multi-class classifier (Section. 5.2.3) is calculated.

5.3 Network Training

5.3.1 Data Set

The selected data set comprises 1500 shapes out of 3500 shapes in the car domain consisting of shapes from three car classes and 1100 shapes out of 4045 airliner shapes consisting of shapes from airliner and propeller classes from the ShapeNetCore data set [18]. Each shape consists of N points, here: $N = 24578$. In total, 2600 point cloud shapes are considered in the data set and the coordinates of each point cloud were normalized to the range $[0.1, 0.9]^3$, preserving the aspect ratio of the shapes. For training the network, the considered data set is split into a 90% training set and 10% test set.

5.3.2 Training the Split-AE

The network in Fig. 5.3 is trained with a 5D latent vector ($L_z = 5$) for each content and style code using the Adam optimizer [76] with a learning rate $\eta = 5\text{E-}04$. The dimension of L_z is set to 5, after analyzing the trade-off between L_z dimension between 2 to 10 and reconstruction accuracy on the test set. Further, since the objective is to keep the number of latent variables low for visual verification. Therefore, for this experiment the dimension of L_z is fixed to 5. The training data was organized into batches of 50 shapes and the network was trained for 700 epochs.

The hyper-parameters α and β in the loss function (Eq. 5.1) are set to $\alpha = 0.03$ and $\beta = 0.1$, such that the trade-off between two loss functions resulted in an acceptable reconstruction accuracy and provided equal importance between the two classification loss functions. For hyper-parameter tuning, a 20% of the training set is used as the validation set in the initial run. After the parameters are finalized, the shapes in the validation set are added back to the training set.

5.3.3 Training Baseline Models

To compare the performance of Split-AE with existing models, two baseline models are selected. As a first baseline model, PC-AE [134] is selected, since the encoder-decoder architecture of this network is utilized in the Split-AE. The PC-AE network was trained utilizing mean square distance (MSD) as a reconstruction loss function. As a second baseline model, a PC-AE-classifier is considered, where the similar PC-AE network is trained with an additional cross-entropy classification loss function to discriminate all classes equally. Both networks are trained with a 10D latent vector using the Adam optimizer and with a learning rate of $\eta = 5\text{E-}04$ for 700 epochs.

Regarding the hardware set-up, all networks are trained on a machine with two CPUs Intel®Xeon®Silver, clocked at 2.10 GHz, and with two GPUs NVidia®GeForce®RTX 8000 with 48GB each. In all cases, the networks were trained on a single GPU.

5.4 Shape-Generation and Feature Analysis

The performances of the networks are compared based on two criteria: reconstruction quality and style-class classification performance on the test set using the quantitative metrics from Section 5.2.3.

Reconstruction Capability of Models: The reconstruction losses are calculated between the reconstructed output point clouds and their corresponding input point clouds in the training and test sets using Chamfer distance (CD) (Table 5.1).

For a 95% confidence interval, Split-AE achieved lower reconstruction errors (low CD), which are better than the errors of the PC-AE and PC-AE-classifier. The reconstruction quality of Split-AE is closer to the PC-AE-classifier, which indicates that training a network with a classification loss improves the quality of the reconstructed shapes. However, the run-time of Split-AE is much higher compared to other models, and this is due to the extended architecture and multitask loss functions that are used for training the Split-AE.

Table 5.1 Reconstruction performance of the models. Best models with less training set error (CD_{train}), test set error (CD_{test}) and low run-time are marked in bold.

	Split-AE	PC-AE	PC-AE-Classifier
CD_{train}	**(7.54±.77)E-05**	(9.70±.12)E-05	(8.69±.41)E-05
CD_{test}	**(8.00±.31)E-05**	(10.0±.20)E-05	(8.94±.50)E-05
Run-time	4hrs 50mins 20secs	**1hr 40mins 20secs**	2hrs 54mins 32 secs

Style Class Classification Accuracy based on Latent Representation: Since the Split-AE network is designed to enforce latent variables to learn distinct visual geometric features of each class of 3D shapes, a classification accuracy is measured on the latent space of this network to analyze the disentanglement of the style features in the latent space of the models. The classification accuracy of the trained multi-class classifier (Section 5.2.3) is measured based on its ability to predict the class labels of the test samples from their latent representations (Table 5.2). Both the baseline models learn content and style features of 3D shapes in a common latent space (Z). Therefore, the style class classification accuracy of the two baseline models is calculated based on the latent space Z opposed to Split-AE, where we calculated classification accuracy based on content (Z_c), style (Z_s) and combined space (Z). Higher accuracy relates to the model's ability to generate distinctly separable latent representations of the test samples based on style classes.

Table 5.2 Style class classification accuracy of the models. Models with high classification accuracy are marked in bold.

	Split-AE	PC-AE	PC-AE-classifier
Content space (Z_c)	0.953±.004	-	-
Style space (Z_s)	0.972±.006	-	-
Combined space (Z)	**0.979±.003**	0.935±.005	**0.977± .004**

In the Table 5.2, the 5D style space in Split-AE shows higher accuracy compared to the 5D content space. These results stress the effectiveness of our disentangling approach, as the style space holds higher class-specific information in comparison to the content space, even though the style space in Split-AE learns class labels based on shape similarity. Further, in the combined latent space of Split-AE, classification accuracy improved, and became similar to the PC-AE-classifier, signifying models trained with classification loss generated better disentangled latent representations based on style classes. In the next set of experiments, an intuitive interpretation of the features learned by the content and style variables of the Split-AE is provided for representing the underlying factors of variations in 3D shapes.

5.4.1 Disentangled shape features learned by Split-AE

Content and Style Variables: For qualitative analysis of the learned features and better understanding of the complex model architectures, the feature visualization approach [133] is utilized for projecting learned network features into human-comprehensible space, i.e., in the 3D input space. To verify the relation between latent variables and the point distributions of a 3D shape in the input space, the activation values of the last convolutional

layer of both encoders (E_c and E_s) are projected onto an input point cloud. Fig. 5.4, shows the visualization of features learned by content ($z_{c_l}, l = 1..., 5$) and style ($z_{s_l}, l = 1, .., 5$) variables of the trained Split-AE on an SUV car shape.

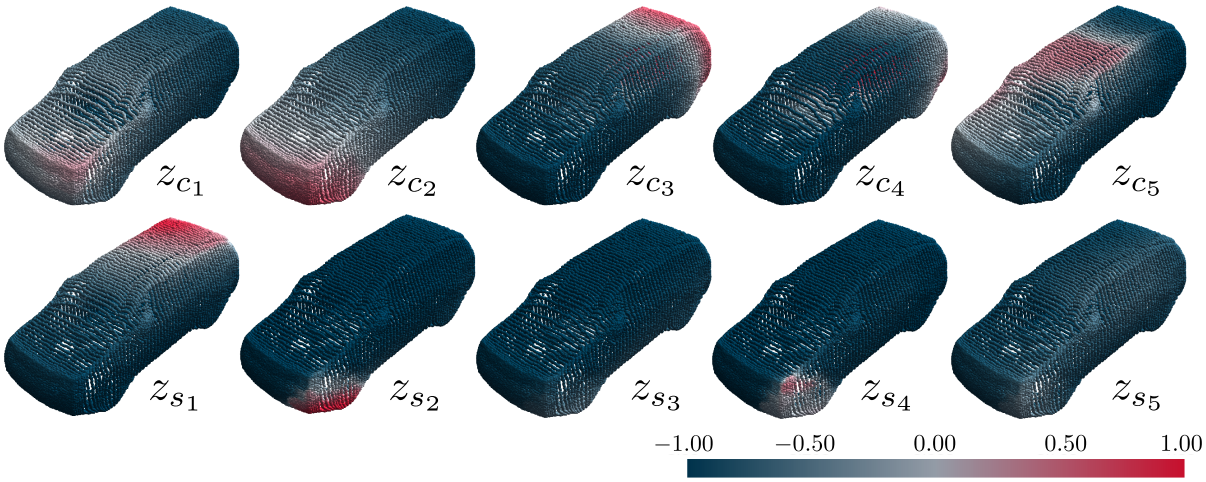


Fig. 5.4 Feature visualization method applied to a point cloud representation of an SUV car shape obtained using Split-AE with 5D content and 5D style spaces. The scale shows the activation values of the visualized features.

By analyzing the visualizations across multiple input shapes, the observation is that the highlighted color for each feature maps to a similar region in the input space rather than similar geometric characteristics of the input shapes [133]. Also, the regions of activation in the input space are different for style z_{s_i} and content variables z_{c_i}, where z_{c_i} mapped to wider regions in the input space compared to z_{s_i}, which focused on distinct smaller regions in the input space with higher activation values. Hence, the experiment shows that features learned by the style latent variables map to distinctive localized regions in the input space to ensure separability between classes.

Linear Combination of Features: As a second test to verify that content variables hinder learning distinct traits specific to the shapes of each car class, a mean representation x_μ is generated by estimating a mean content $\vec{z}_{c\mu}$ and style code $\vec{z}_{s\mu}$ of the car shapes from the training set, such as $x_\mu \sim \vec{z}_\mu = (\vec{z}_{c\mu}, \vec{z}_{s\mu})$. Next, three distinct shapes (x_1, x_2 and x_3) are considered from 3 car classes and combined the content codes of these shapes with the mean style code $\vec{z}_{s\mu}$ of the shape x_μ (Fig. 5.5). The assumption is that the 3 generated shapes would potentially represent shapes similar to the mean shape x_μ. Hence, these 3 generated shapes cannot be classified into their 3 respective car classes, since the distinct content codes learn an overall global representation of 3D shapes.

In Fig. 5.5, selected shapes from each car class are represented by content and style codes $\vec{z}_{cj}$ and $\vec{z}_{sj}$ ($x_j \sim \vec{z}_j = (\vec{z}_{cj}, \vec{z}_{sj}), j = 1, 2, 3$). The augmented shapes are generated ($x_{1\mu}, x_{2\mu}$ and $x_{3\mu}$) by combining distinct content codes ($\vec{z}_{c1}$, $\vec{z}_{c2}$ and $\vec{z}_{c3}$) with the mean

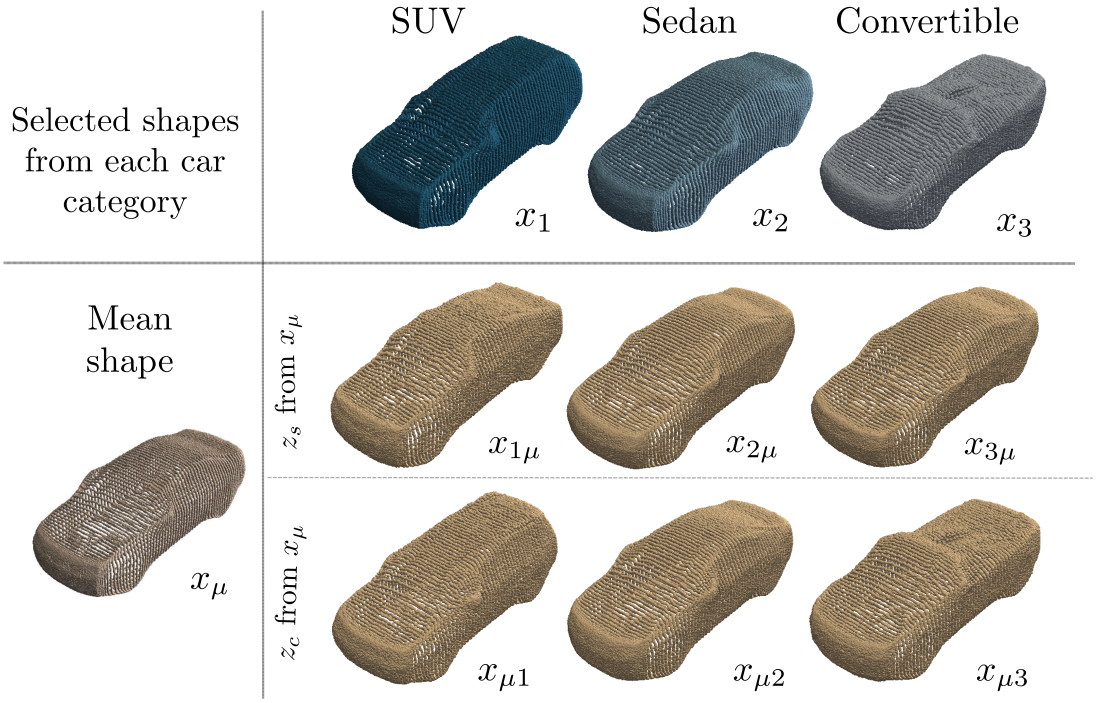


Fig. 5.5 First row: $x_{1\mu}, x_{2\mu}$ and $x_{3\mu}$ shapes generated by combining content code of x_1, x_2 and x_3 with the style code of x_μ, respectively. Second row: $x_{\mu 1}, x_{\mu 2}$ and $x_{\mu 3}$ shapes generated by combining the content code of x_μ with style code of x_1, x_2 and x_3, respectively.

style code $\vec{z}_{s_\mu}$ such as $x_{1\mu} \sim \vec{z}_{1\mu} = (\vec{z}_{c1}, \vec{z}_{s\mu})$. As observed (Fig. 5.5), the generated shapes $x_{1\mu}, x_{2\mu}$ and $x_{3\mu}$ are similar to each other without having distinctive shape parts (roof structures similar in all 3 shapes). Therefore, it signifies that content variables do not learn distinct features to differentiate shapes across shape categories.

Alternatively, combining the mean content code $\vec{z}_{c\mu}$ of the shape x_μ with distinct style codes of shapes from 3 car classes should generate shapes that can be classified into their respective car classes, since style learns localized traits distinct to each car classes. Each of the generated shapes $(x_{\mu j}, j = 1, 2, 3)$ represents a shape from a different car class, as observed in Fig. 5.5 by visual inspection. The generated augmented shape $x_{\mu 1}$ represents an SUV car design like x_1. Likewise, the generated shape $x_{\mu 2}$ resembles a sedan shape (x_2) and $x_{\mu 3}$ resembles a convertible shape (x_3).

Therefore, the style code learns localized distinctive shape parts that differentiate a shape from others with the same content and can describe shapes in more detail. While the content code learns a global representation of the 3D shape without specific details. In the next set of experiments, Split-AE is utilized as a shape generative model for generating new augmented shapes by style transfers.

5.5 Novel Shape Generations by Style Transfer

In this experiment, Split-AE is utilized to generate new augmented shapes by providing different 3D shapes for style reference (Section 5.2.4).

Augmented Shape Generation and Validation: 3D shape-to-shape style transfer is performed between shapes within a domain. The assumption was that the augmented shape should possess the distinct shape part of the target reference shape class from where it adapted the style code. Fig. 5.6 shows examples of augmented shapes generated by style transfer between source and target shapes in the car and airplane domains.

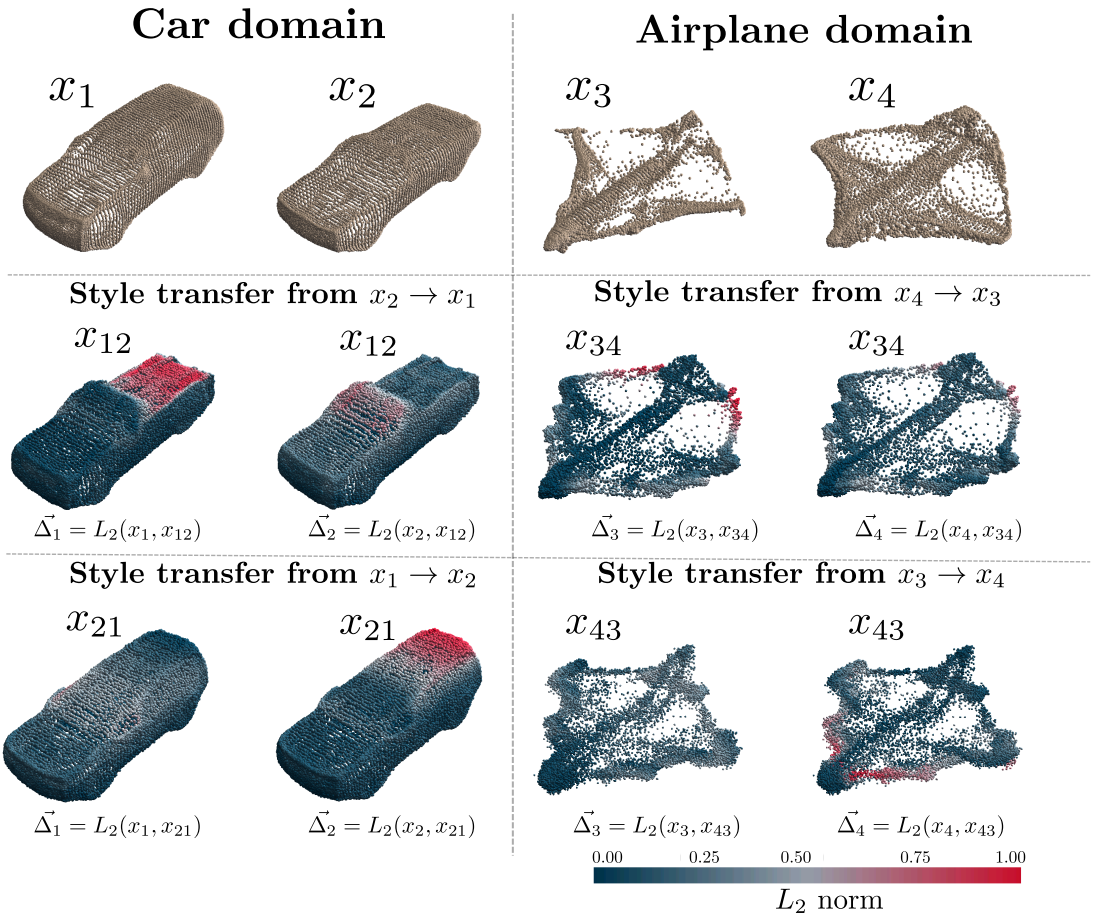


Fig. 5.6 In car and airplane domains, we illustrate style transfer results for pairs $x_1 \leftrightarrow x_2$ and $x_3 \leftrightarrow x_4$, respectively. Each point of the generated point cloud shapes (x_{12}, x_{21}, x_{34} and x_{43}) is color coded based on the Euclidean (L_2) distance vector ($\vec{\Delta}_i, i = 1, .., 4$) written below each generated shape.

In the car domain (Fig. 5.6), an SUV (x_1) and a convertible (x_2) are selected as a source-target pair. The augmented shapes x_{12} and x_{21} are generated by style transfer from $x_2 \rightarrow x_1$ and $x_1 \rightarrow x_2$, respectively. The roof structures of the two generated shapes (x_{12} and x_{21}) change according to the reference shape class from where it adapted the style codes, i.e., for x_{12} the roof structure is like x_2 and for x_{21} like x_1.

Further, for qualitative analysis of the structural similarity between the generated shape x_{12} to x_1 and x_2, two normalized Euclidean (L_2) distance vectors ($\vec{\Delta}_1$ and $\vec{\Delta}_2$) are calculated between each point in the generated point cloud shape x_{12} with x_1 and x_2. Fig. 5.6 shows two samples of the generated shapes x_{12} with projected $\vec{\Delta}_1$ and $\vec{\Delta}_2$ as color maps onto the two 3D point cloud representations of x_{12}, respectively. The generated shape x_{12} color coded with $\vec{\Delta}_1$ shows the largest differences (high normalized L_2 values) in the roof region with respect to x_1 and higher similarity (low normalized L_2 values) in the frontal and lower region with x_1 from where it adapted the content code. Analogously, for the generated shape x_{12} color-coded with $\vec{\Delta}_2$, the roof structure is similar (low normalized L_2 values) to the target shape x_2. Alternatively, the augmented shape x_{21} generated by style transfer from $x_1 \rightarrow x_2$ shows the roof structure like x_1 and the bottom region of the car shape similar to x_2. These results signify that style transfers between source-target pairs generate new shapes with modified shape parts.

Likewise, in the airplane domain (Fig. 5.6), style transfers between an airliner (x_3) and a propeller (x_4) shape pair generated shapes x_{34} and x_{43}. Different from the style transfer in the car domain, each of the generated augmented shape (x_{34} or x_{43}) in the airplane domain changes the wing structures according to the reference shape from where it adapted the style code, i.e., the wings of the generated shape x_{34} changes according to the propeller shape x_4. Thus, Split-AE correctly identifies distinctive styles in different domains.

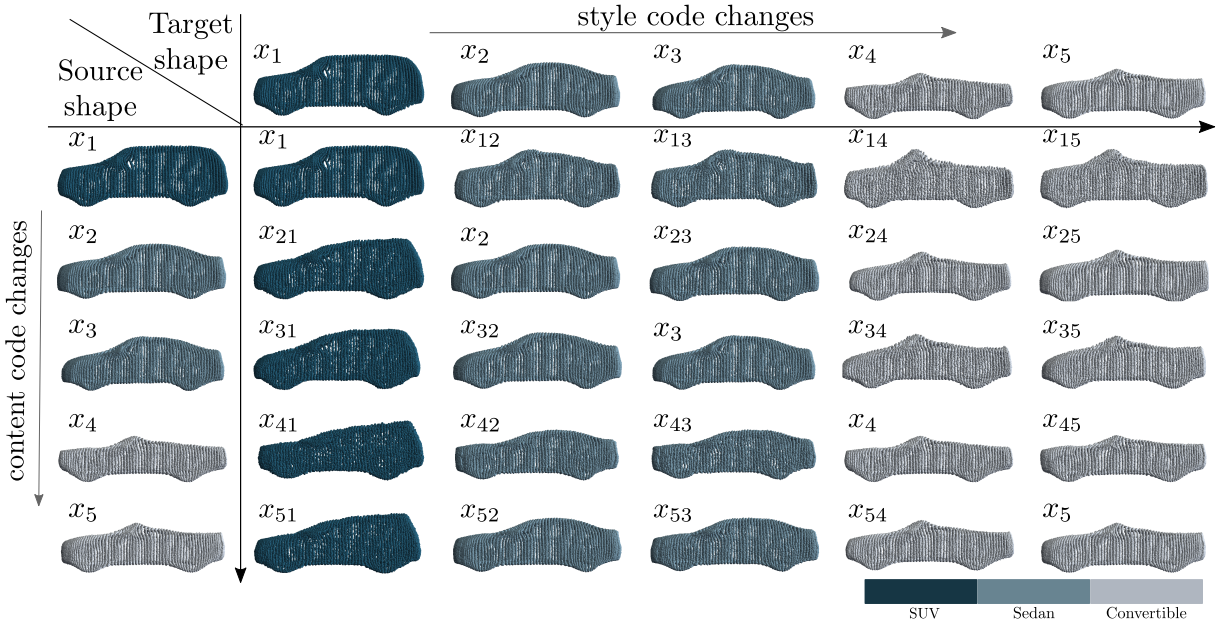


Fig. 5.7 Visualization of reconstructions of 25 augmented car shapes generated by style transfer from target to source shapes. The content code is fixed in each row while the style code varies. Similarly, the style code is fixed in each column while the content code varies. Each of the generated shapes is color-coded based on the predicted class label.

Additionally, to confirm our observations in Fig. 5.6 that each style transfer generates a novel augmented shape which has mixed shape parts from both source-target shapes, the performance of style transfers for additional augmented shapes from each domain is analyzed. 500 new augmented car shapes and 500 airplane shapes are generated by randomly selecting 500 source-target pairs from the car and airplane shapes in the data set, respectively. For qualitative visual inspection, 25 out of 500 augmented car shapes are shown in Fig. 5.7, where the target shape alters in each column and the source shape changes row-wise. Thus, the horizontal axis indicates traversing of style and the vertical axis shows the change of content; i.e., shapes in each row have fixed content with changing style codes. To verify the distinct shape parts similarity of the augmented shapes with the target shape class, the trained multi-class classifier is used to predict the class label of each augmented shape from its latent representation $\vec{z}$ and color-coded each shape based on the predicted class label (Section 5.2.3).

In the first row of Fig. 5.7, the augmented shapes $(x_{1j}, j = 1, .., 5)$ are generated by combining the content code of an SUV shape x_1 with style codes of shapes from different car classes. The roof structure of the augmented shapes in the first-row $(x_{1j}, j = 1, .., 5)$ are significantly different in Fig. 5.7. However, these shapes preserve similarity in the lower-body and frontal design with the source shape x_1. Likewise, the augmented shapes $(x_{j1}, j = 1, .., 5)$ in the first column hold distinct shape parts (roof structure) of the SUV class, but the frontal and lower-body design changes in each row according to each row-based source shape. Also, for each augmented shape in Fig. 5.6, the prediction of the class label using the trained multi-class classifier shows that the style of each target shape dominated the class allocation of each augmented shape. Hence, column-wise, each generated shape has a similar prediction label (same color) that matches the target shape label in that column.

In addition, the class labels of 500 shapes are predicted using the trained multi-class classifier and accuracy is measured based on the class label prediction of augmented shapes and their respective target shape class. Style transfer in both domains shows a high accuracy of 0.91 for 500 generated car shapes and 0.82 for 500 airplane shapes, indicating the success of our style transfer approach. Therefore, Split-AE provides the flexibility to combine features between two shapes with its disentangled latent spaces. Moreover, Split-AE excels in part-based modification by replacing distinct shape parts, while accurately maintaining the underlying structure for generating novel realistic 3D shapes. This helps to generate novel shapes which are modified versions of the existing designs.

Latent Space Exploration: As a second experiment to evaluate the goodness of learned disentangled variables, a latent space-based interpolation is performed both in the content and style space. Fig. 5.8 shows the uniform distance interpolations of the content and

style codes from a source shape (SUV) to a target shape (convertible). Since the Split-AE network was trained to generate an organized point cloud like that of the input shapes, the similarity is measured between each interpolated shape with the target shape to illustrate the change in each 3D shape by interpolation of the latent variables.

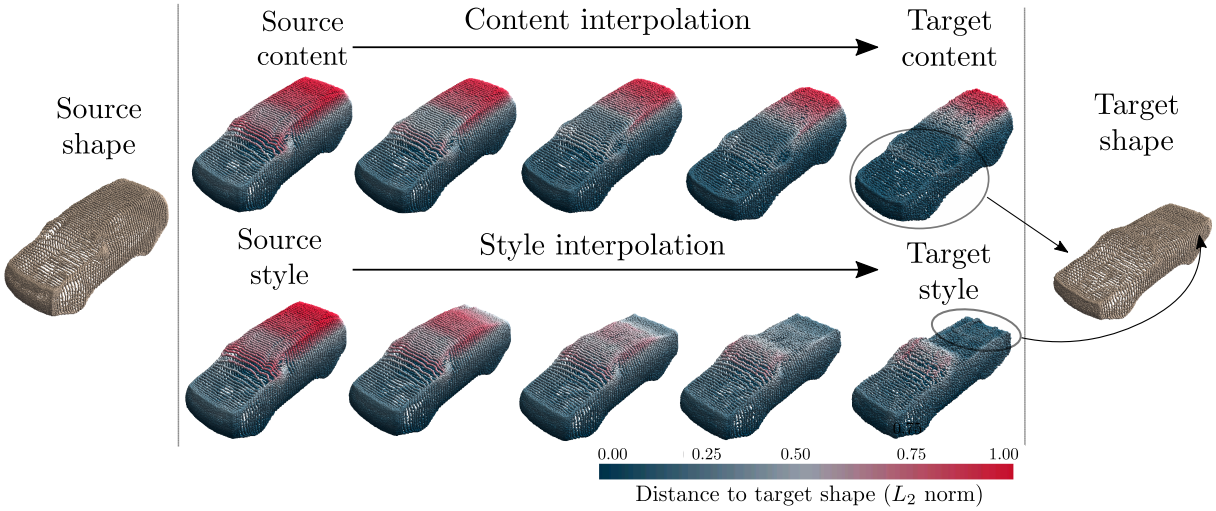


Fig. 5.8 Interpolation in the content and style spaces learned by Split-AE from a source to a target shape. In the first row, we generate shapes by uniformly interpolating between source content and target content, keeping the source style. In the second row, we generate shapes by uniformly interpolating between source style and target style, keeping the source content constant.

The shapes produced with the interpolated contents in the content space keeping the source-style code fixed (Fig. 5.8), show the frontal and lower region of the exterior design of the car shapes gradually becoming similar to the target shape (low normalized L_2 distances). Alternatively, shapes produced by interpolating style from the source to the target shape keeping the content of the source shape fixed, show a gradual change in the roof designs to make the roof design of the interpolated shapes similar to the convertible (low normalized L_2 distances). This shows the credibility of two separate content and style spaces for design exploration.

5.6 Conclusion

In this chapter, one of the challenges of controlling interpretable aspects of 3D shapes with the latent variables of autoencoders is addressed. As a solution, a novel deep-generative model is proposed, named as Split-AE. The proposed network helps to disentangle 3D representations into content and style features involving minimal supervision. Based on a set of experiments using quantitative and qualitative measures, Split-AE achieves an effective disentanglement of content and style space that represent various aspects of 3D shapes. The proposed architecture can perform 3D shape-to-shape feature transfer between shapes across multiple classes within a domain. Further, this architecture is

easily extendable to learn content and style codes for shapes across multiple classes and domains, which allows generating augmented shapes using style transfer in different domains. However, the main technical limitation of this work is that the style features considered here resemble localized regions of 3D shapes. A future analysis could consider more global style features, which will help to understand the style of cars from distinct brands.

The work in this chapter is based on a published work that was conducted by the author [135]. Also, based on the discussion in this chapter, the following answer is proposed for *RQ*5.

RQ 5. How can the design features learned by a deep generative model represent different interpretable aspects of 3D designs? How can we transfer the unique parts or design features of a 3D shape to another shape, such that the recombination of design features from two distinctive shapes generates a novel shape combining the essence of both 3D designs?

In the context of the experience-based design optimization framework (Fig. 1.4), transferring unique design features of one shape to another is a useful for generating novel design variations. Based on prior research in this thesis, the latent variables of the PC-(V)AE are favorable for low-dimensional representation of 3D designs. However, the relation of the latent variables with different interpretable features of 3D designs are less explored.

To address this challenge, Split-AE is proposed, which is built on the architecture of the PC-(V)AE. The proposed method learns the 3D shapes in an unsupervised fashion and generates two sets of latent variables named as content and style. Each of the content and style latent space is shared between shapes from multiple classes and across semantic categories. Through quantitative and qualitative evaluations, it is shown that the style variables map to distinctive localized regions in the input space to ensure separability between shapes from different classes. While the content variables learn an overall global representation of the 3D shapes, without learning distinctive traits specific to the shapes of each class.

Next, to generate augmented shapes that combine the essence of two different 3D shapes, the style latent variables of a target shape are transferred to a source shape. This recombination of the style code of the target shape and the content code of the source shape in the latent manifold, results in generating a novel realistic 3D shape. This newly generated shape is a combination of the essence of two existing designs. Therefore, the proposed method, involving minimum supervision to disentangle 3D representations, helps to generate novel shapes that are modified versions of the existing designs.

Based on the discussion presented in this chapter, the PC-VAE enables the generation of disentangled latent variables, that is efficient for feature transfer between 3D designs. Yet, one practical aspect that still hinders the use of PC-VAE in engineering design guidance applications, is that the latent manifold is still unknown to novice users. Therefore, in the following chapter, an interactive framework is proposed to solve an engineering design problem and provide target-oriented guidance to novice designers for latent space-based design explorations.

Chapter 6

An Interactive and Collaborative 3D Automotive Design Framework

To achieve the envisioned experience-based design optimization framework, the research in the previous chapters addresses the two main components of Fig. 1.4: The first step refers to developing a data-driven deep learning model that helps in generating low-dimensional latent manifold of 3D designs and evaluating the model's capability for generating novel realistic car designs. The second step focuses on analyzing the latent manifold for several tasks, such as generating 3D designs satisfying multi-criteria [125], building surrogate models for faster performance predictions of 3D designs [123] and unique feature transfer between 3D designs [135]. The research in this chapter, combines the findings from prior chapters to develop an interactive cooperative design system (CDS) framework. The aim is to demonstrate the usability of the framework in a real-world inspired optimization scenario, where the designer can interact with the proposed framework. The framework should provide guidance or suggest potential design alternatives, giving the designer the freedom to rethink and adapt promising design directions.

In the early concept ideation phase, this type of framework would be beneficial for designers or engineers, which they could use as an additional tool for 3D design explorations. Mostly, a designer starts either from scratch or an initial design or may choose some alternative methods to modify an initial design to match his/her targets. Such a target may be either given implicitly, e.g., based on aesthetic considerations, or explicitly, e.g., based on functional or structural requirements. A designer using a sketch-based approach to outline a target design, does not need any explicit parametrization. However, a designer using 3D tools utilizes a shape parametrization method that represents 3D geometries with a series of design variables and helps the designer in performing 3D design modifications. Often, the designer uses the shape parameterization parameters to modify 3D designs based on whatever is easiest, not based on what leads to a better decision. Hence, deciding the target-oriented modification steps is still challenging for a novice designer in automotive

optimization tasks. To support the designer with our proposed CDS, we imagine the CDS as a data-driven model that learns the notion of experience from past engineering optimization data and provides collaborative assistance during current automotive design optimization.

Engineering experience from prior optimizations can be represented using two-types of data: The first method represents the experience embedded in engineering data and the second method represents the human-experience, which resembles the engineer's approach of performing past design optimizations. Based on these two types of data, we envision two types of CDS: The first framework to model the experience that is embedded in the historical or benchmark 3D CAE data, which is majorly addressed in the research of this dissertation. The proposed PC-VAE [77] (Fig.3.1) trained on CAE models, generate latent representations of CAE models, which are used as a low-dimensional search space for 3D designs. An interactive framework is introduced using the latent variables of the PC-VAE to allow the designers/engineers to explore 3D designs with attributes according to their preference. The second framework focuses on the perception of human experience of performing past design optimization tasks, which is harder to capture. A preliminary study is done to capture the engineers' approach of 3D shape manipulation in a design optimization task, which results in a large data set of sequence data. Learning the existing 3D design manipulation sequences using deep learning models like recurrent neural network (RNN), long short-term memory (LSTM) and gated recurrent unit (GRU) facilitates the prediction of future design steps from the modification currently applied. Therefore, the objective of this research is to exploit these two categories of experience data using deep learning models to generate real-time feedback on the current design and collaboratively guide the designer through the design process.

The remainder of this chapter is outlined as follows: First, in Section 6.1, the existing design assistance frameworks are reviewed to decide the type of assistance that would be beneficial for an automotive designer. Section 6.2 demonstrates a real-world-inspired automotive design optimization task, which is used as a case study to demonstrate the application of the proposed frameworks. The next Section 6.3 introduces the 3D data-driven interactive framework, which is utilized to provide guidance to the designer while performing the design optimization task. Section 6.4, describes the second framework that focuses on learning the human-notion of experience, i.e., learning from prior shape manipulation data using an LSTM. The trained LSTM is then utilized for supporting the similar automotive design optimization task, where this framework predicts the future design steps from the modification currently applied. Finally, the chapter is concluded in Section 6.5 with a summary and outlook.

6.1 Prior 3D Design Assistance Systems

Recent advances in artificial intelligence (AI) aim to collect and use existing digital design and simulation data from past development cycles for the application of machine learning techniques and data analytics. In the context of 2D design, two systems have been proposed, namely SketchRNN [13] and ShadowDraw [12], which provide real-time guidance to human users while drawing online. Both models use a deep generative network and are trained on a database of drawing sequences by human users. During interaction for sketching, both networks propose potential design directions for the next drawing steps. However, the sketches generated by these models are far away from anything resembling an industrial design tool.

In the context of automotive digital development, designers rely on sketching for a first ideation and use existing tools such as, CAD/E, CATIA or Alias to accelerate various tasks, particularly through virtual prototyping. These tools enrich the design process by providing a means to alter shapes and create novel variations of the 3D designs. Designers and engineers use these tools to generate prototypes that match their requirements and estimate the technical performance of the 3D designs. Prior research on developing a design support system [136, 137] attempted to bridge the gap between the needs of an engineer and what is currently available on a CAD system. Nevertheless, there is still an existing gap which CAD systems don't address. For example, to generate novel prototypes or improve existing prototypes, it would be beneficial to provide guidance to the designer about structural or functional information of 3D designs while exploring the design space. Yet accurate performance simulation of 3D designs still requires excessive computational effort. This makes it a challenging task for the designer to consider multiple performance measures during the design ideation phase. Umetani [138] proposed a system using a deep learning model for faster modification of 3D digital vehicles. However, it lacks explicit assistance for user guidance. Therefore, to address the challenge of faster and informed 3D shape manipulation based on technical design quality indications and to support the designer in the creative thinking process, an interactive framework for automotive design exploration is required.

Ideally, 3D designs are parameterized with a series of design variables that help faster design modifications and generations. One popular traditional method of shape parameterization involves free-form deformation (FFD) [10] that helps to modify 3D designs with a higher degree of flexibility and a small number of parameters, named as control points. FFD decouples design complexity from parameterization. However, the number of control points scales with the details of design variations. Therefore, low dimensional detailed modifications of complex geometries either need experienced engineers or high dimensional parameterization. Either engineers or designers use FFD parameters

for shape modifications or perform design optimization with these parameters, which results in potentially large data sets of control point manipulations.

Alternatively, based on the reviewed methods in the earlier chapters, latent variables of the PC-(V)AEs are a suitable approach for shape parametrization that helps to modify 3D designs [77], combine geometric features between 3D shapes [135] and support performance prediction of 3D designs [125]. These PC-(V)AEs trained on design representations of CAE models can potentially learn enough information in the latent variables, so that these variables are applicable as control parameters for an interactive framework. With the increase in detail of the possible design modifications, the dimension of the latent variables needs to be increased. However, the advantage of this approach is the automatic learning of experience from engineering data and does not depend on the engineers' expertise for the scalability of this approach. Therefore, based on the objective of this chapter to model the data-driven and human-driven experiences, two assistance frameworks are set up. First, to build an interactive framework trained on 3D engineering data, the proposed PC-VAE is used to learn from 3D geometries data and produce a data-driven latent manifold for interactive search and guidance for 3D design generation. The second approach aims to learn from a designer's experience of performing past optimization tasks and provide real-time guidance to the designer. To capture the designer's experience of performing an engineering optimization task, we use the FFD and track the control point movements in a given shape manipulation task, which helps to generate a large dataset of sequential recording of control points. Next, we demonstrate the feasibility of recurrent neural networks for learning successful design sequences in order to predict promising next design steps in future design tasks, which is similar in line with the ideal engineering system suggested by [137].

6.2 Design Optimization Scenario

To demonstrate the application of a design assistance system, a real-world inspired design optimization task is set up. An optimal way of defining an optimization target is to select a design with minimal distance to the design that the designer wants to achieve. Thus, the optimization task for this set of experiments involves modifying an initial shape to match a target shape that represents the designer's target.

In this set of experiments, a shape representing an initial design (x_{in}) and two different shapes resembling two targets (x_{T1} and x_{T2}) are selected (Fig. 6.1). The initial design corresponds to the shape in the dataset that represents the mean 3D point cloud, which is obtained by averaging the point cloud coordinates of all samples in the data set.

For the given optimization task, we developed two assistance systems based on the two types of experience data (Fig. 6.2). The first framework follows the 3D data-driven

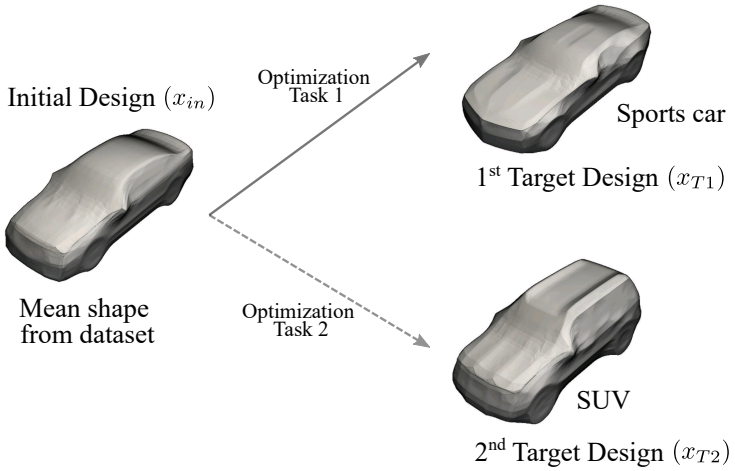


Fig. 6.1 Two optimization task to modify $x_{in} \rightarrow x_{T1}$ and $x_{in} \rightarrow x_{T2}$.

approach, which shows how to modify the initial design to the target design based on the latent space of the PC-VAE. The second method aims to capture the designers' approach of performing the similar past optimization task using FFD, which eventually results in a large dataset of sequences. An LSTM trained on the past design sequences is used to guide the designer in the current design task by predicting the next design step to achieve the target shape.

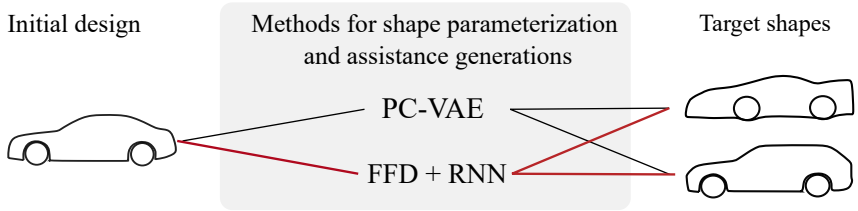


Fig. 6.2 Schematic overview of the two assistance approaches for the design optimization tasks.

6.3 3D Data-driven Design Assistance

Preliminary steps for building the first assistance framework using PC-VAE are described in this section. A trained PC-VAE is integrated into a graphical user interface (GUI) and the workflow of the interactive framework is introduced (Section 6.3.1). Next, a demonstration of the interactive framework is given for providing assistance in the considered optimization scenario (Section 6.3.2), along with real-time performance prediction of the current design (Section 6.3.3).

Experimental Setup

The data used in this set of experiments includes 3D point cloud samples from the car class of the ShapeNetCore repository [18]. The PC-VAE is trained with a latent dimension

of $L_z = 5$ and a mean-square distance as the reconstruction loss function (Section 4.2.2). To show the application of the proposed framework, the learned latent variables are used as interactive elements for modifying 3D designs. Additionally, for providing faster information about the performance measures of 3D designs, the surrogate model from prior research (Sec. 4.2.5) is integrated with this framework (Fig. 6.3).

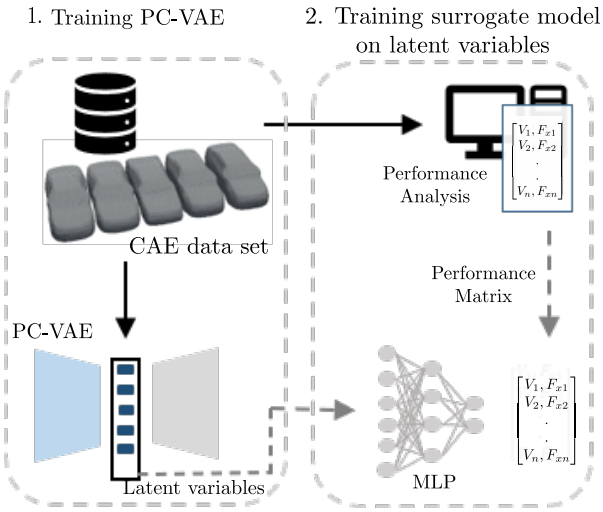


Fig. 6.3 Pre-processing steps and workflow for training the PC-VAE and MLP.

6.3.1 Interactive Framework

A graphical user interface (GUI) has been implemented using python and pyvista (Fig. 6.4), where each slider modifies one of the 5 latent variables. The GUI loads the initial design (x_{in}) in the form of a 3D point cloud representation, with the current position of the sliders representing the latent variables of x_{in}. Interacting with the sliders updates the new values of the latent variables, and the PC-VAE decoder automatically calculates and displays the updated 3D point cloud based on the modified 5 latent parameters.

To analyze the range of the distributions of the latent variables, a box plot of the latent variables ($z_i, i = 1, 2, ..5$) from the data set is shown (Fig. 6.5a). The lower ($z_{l_i}, i = 1, 2, ..5$) and upper limit ($z_{u_i}, i = 1, 2, ..5$) of each latent variable represents the lower and upper limits (min and max values) of the sliders in the GUI. The range of the distribution of the latent variables has values of different magnitudes, which is due to the latent values being generated by sampling from the latent distribution of the PC-VAE.

Further, to analyze the impact of each latent variable modification on the generated shape, a sensitivity analysis is performed to determine the regions of the shapes managed by each latent variable z_i. For calculating the sensitivity of each latent variable associated with different regions of the input shape, a total of 30-point clouds are sampled from the training dataset. The total sensitivity of the displacement of each point of 30-point clouds is estimated to determine the association with each latent variable z_i. For these

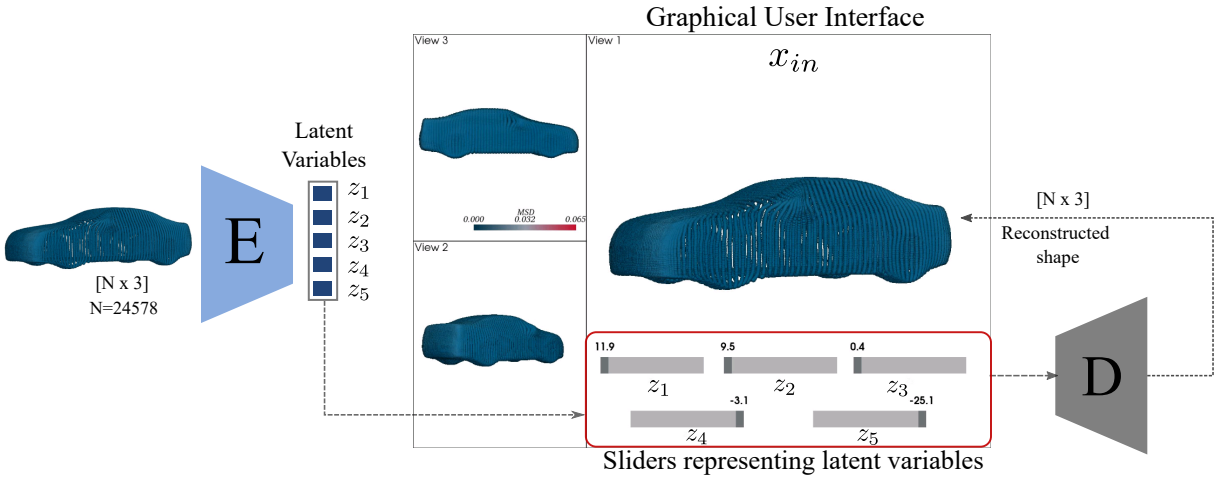


Fig. 6.4 Schematic overview of the interactive framework.

sensitivities calculations, the Sobol method with Satelli's sampling [139] is used, where samples are generated as $30(L_z + 2)$ variations in a range of $\pm 30\%$ of each latent variable, considering $L_z = 5$. The Sobol measures the sensitivity values of each point in the point cloud representation associated with each latent variable (z_i). The sensitivities obtained from the 5 latent variables are projected onto a 3D point cloud selected (Fig. 6.5b), which show that each of the latent variables z_i is associated with changes in larger or different regions in the output point cloud. However, the impact of change on the design topology of the output point cloud is different for each latent variable z_i, e.g., the sensitivity of displacement concerning z_5 is higher compared to the other latent variables. The sensitivity analysis is presented to the user when the user starts the GUI interface.

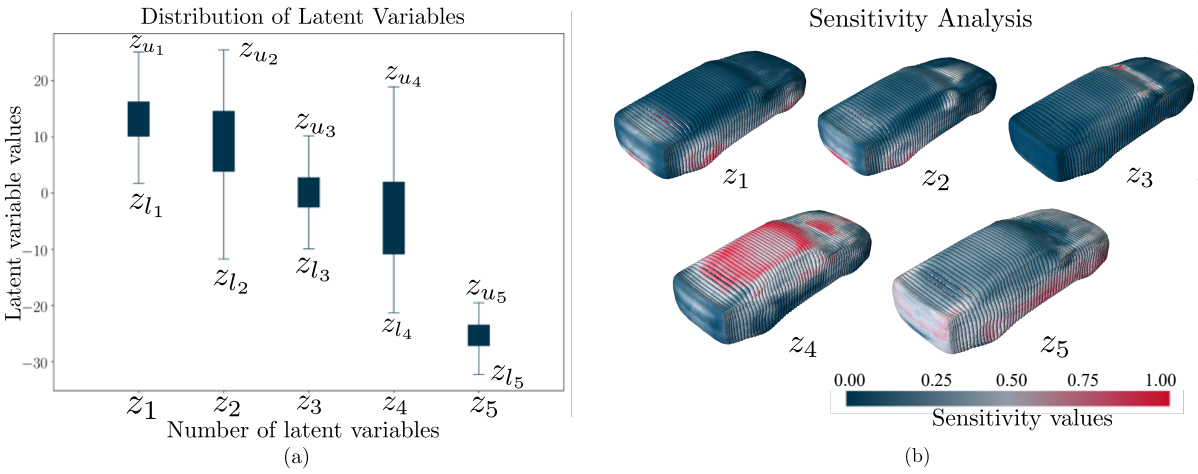


Fig. 6.5 a) Distribution of latent variables of the training set shapes. b) Visualization of the displacement sensitivity of points in 3D point cloud of the initial design (x_{in}) to the 5 latent variables. The brighter red color indicates higher sensitivity.

6.3.2 Framework for Target-oriented Design Exploration

This experiment demonstrates the applicability of the framework for a target oriented design generation task (Section 6.2), where depending on user preference, the framework provides personalized ranges of the latent variable sliders to guide the generation of a 3D design that matches the user target. Here, we considered that a design target is defined by the user as one of the car classes from the following car classes: "SUV", "Sedan", "Convertibles" and "Sports car" as provided with labels by the ShapeNetCore data set. For example, if the user's target is to explore sports car designs, the user chooses "Sports car" as the target car class from the above-mentioned car classes. Hence, the objective of this framework is the exploration of different Sports car or SUV designs. Based on the choice of a target class (Sports car), the region (cluster) in the latent manifold of the PC-VAE that represents the target sports car designs is estimated. The personalized range of the sliders is decided based on the location of the initial design and the selected target class region in the latent space. The user modifies the personalized latent sliders to change the initial design (x_{in}) to a 3D shape representing a sports car design.

However, many shapes from ShapeNetCore car classes are not labeled correctly. The shape-generative capability of the PC-VAE is used to generate a cluster of similar yet different sports car shapes from the 10 correctly labeled sports car templates selected from the ShapeNetCore car dataset. Ten distinct sports car templates are considered from the data set with the "Sports car" label after visual verification, since various categories of sports car exist in the dataset. From the 10 chosen sports car template shapes, the latent distributions of these 10 samples ($\vec{\mu}_i, \vec{\sigma}_i = 1, 2..., 10$) are estimated using the encoder of the PC-VAE, and 10 latent representations $\vec{z}$ are sampled from these 10 distributions ($\vec{\mu}_j, \vec{\sigma}_{j)}$ (Section 3.2.3). Thus, exploiting the generative capability of the PC-VAE assists in generating a cluster of 100 latent representations $\vec{z}$ for the sports car class. The latent variables of these 100 samples representing different types of sports car designs are analyzed to estimate the lower and upper limits (min and max values) of these 100 latent variables. Determining the limits of these latent variables helps in approximating the personalized ranges of the sliders in the interactive framework. This similar template-based sampling approach is used to generate a cluster of 100 latent variables for other target classes (SUV, sedan, and convertibles). Figure 6.6 shows the latent distributions of the 100 samples from the target sports car cluster and the initial design (x_{in}) to decide the lower ($z_{l_i}, i = 1, 2..., 5$) and upper value (z_{u_i}) of each of the 5 sliders.

To customize the slider ranges before starting the GUI interface, the latent variables of x_{in} are considered as one of the lower (z_{l_i}) or upper limit (z_{u_i}) of the slider and the other limit is determined from the latent variables of the target sports car class (similar to the range of z_2, z_3, z_4, and z_5 in Fig. 6.6b). If the latent variables of x_{in} are within the range of the latent variables of the target classes (similar to the range of z_1 in Fig. 6.6), both

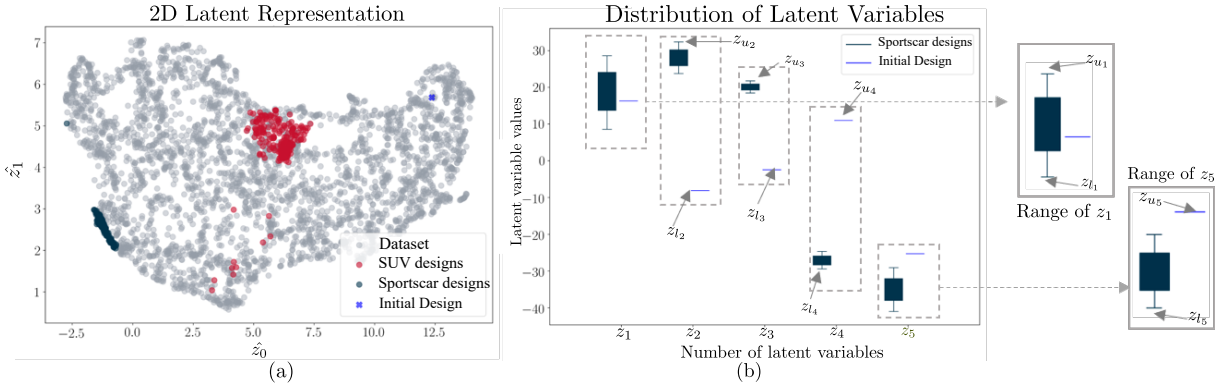


Fig. 6.6 a) 2D representation of the 5D latent variables of the training dataset and the target car class shapes. b) Distribution of the latent variables of the initial design (x_{in}) and the selected target class templates. The brighter red color indicates higher sensitivity.

the upper (z_{u_1}) and lower limits (z_{l_1}) of the target class are considered as the personalized range of the slider z_1. Hence, based on the selection of a target car class by the user, the templates of the selected target class are used to decide the personalized ranges of the latent variables (Fig. 6.7). In addition, to visualize the distinct locations of these target class designs in the latent manifold, the 5D latent representations of the training data set are embedded into a 2D representation using UMAP (Fig. 6.6a). Similarly, the 100 shapes from each of the two target car classes (SUV and sports car) are embedded in the 2D representation (Fig. 6.6a). The designs from the two target classes are in distinct regions of latent space, showing the reliability of the latent manifold since the sports car designs are very different from the SUV designs.

The framework loads the initial design (x_{in}) with personalized slider ranges provided at the bottom of the GUI (Fig. 6.7a), where the current position of each slider reflects the value of one of the latent variables of x_{in}. The user is able to move each slider to generate new designs, where each design is automatically reconstructed from the updated latent variables using the PC-VAE decoder. The first design (x_1) is generated by changing the latent variables z_1, z_2, and z_3 from the initial design x_{in} (Fig. 6.7b). Modifying each latent variable modifies a region of the point cloud, and the mean square distance (MSD) is measured between the current and the previous design. The MSD value between each point of the current shape x_1 and the previous design x_{in} is projected on each point onto the current 3D point cloud representation x_1 as color maps (Fig. 6.7b). This gives an indication to the user that with the current latent variable modifications, which regions of the generated point cloud are modified. The range of the color map is decided based on the min and max values of the difference between one of the target designs and the initial design (x_{in}).

Likewise, after n number of latent variable modifications, the final design of the sports car class is generated (Fig. 6.7c). The color map in the final design (x_n) shows the MSD

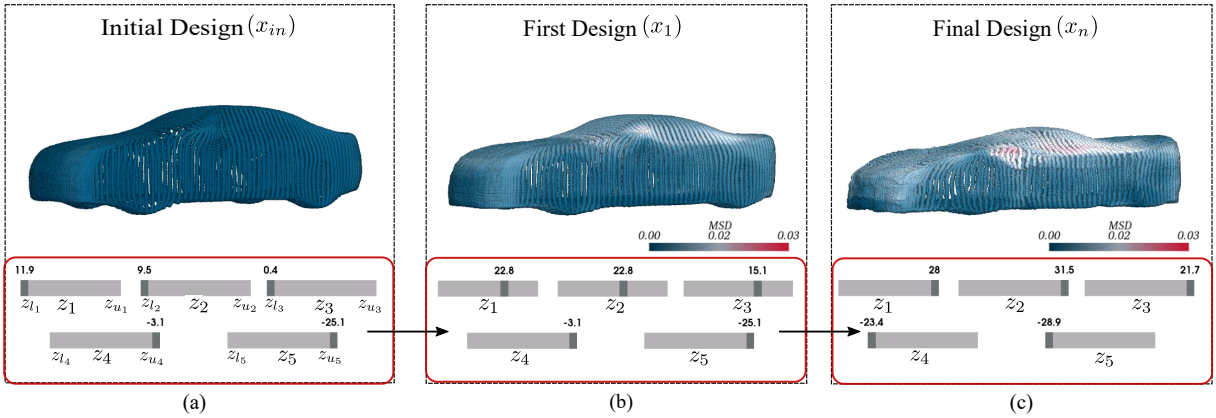


Fig. 6.7 Visualization of the interactive framework with personalized slider ranges to change the design topology from the initial design to a sportscar design. a) An initial design with customized slider ranges derived from (Fig. 6.6). b) The first point cloud design (x_1) generated with z_1, z_2, and z_3 latent variables modification. c) The final design (x_n) generated by moving the sliders. The color map indicates the mean square distance between two geometries, where ref color indicates higher distances.

difference between the points in the shape x_n and earlier shapes x_{n-1} (not shown in the Figure). Other than visually verifying that the final design x_n is representing a sports car, the similarity of the final design is measured with the sports car design (x_{T1}) (Fig. 6.1). This similarity is measured in the latent manifold using the Euclidean distance between the final design and the selected target shape from the sports car class. The observed Euclidean distances are always lower than the maximum pairwise similarity of the sports car templates, proving that the final generated shape (x_n) belongs to the sports car cluster. Hence, the explorative power of the interactive framework helps in guiding the modification of an initial design to a target sports car design x_{T1} that the designer wants to achieve in the design optimization task (Fig. 6.1). Also, while exploring the target-oriented designs using the CDS framework, it generates intermediate feasible designs from which the designers can get inspiration for new design ideas.

Similarly, in the second optimization task (Section 6.2), the use of the interactive framework is shown to modify the initial car design (x_{in}) to an SUV car design (x_n) (Fig. 6.8). Therefore, based on the choice of the target class - sports car (Fig. 6.7) and SUV (Fig. 6.8), the initial design x_{in} loads with different ranges of the sliders allowing the user to explore designs based on their targets. This example illustrates the usability of the proposed interactive framework for guiding design explorations by suggesting personalized latent variable slider ranges. The interactive GUI helps to navigate in the latent manifold for modifying the initial shape to a 3D shape representing the target class.

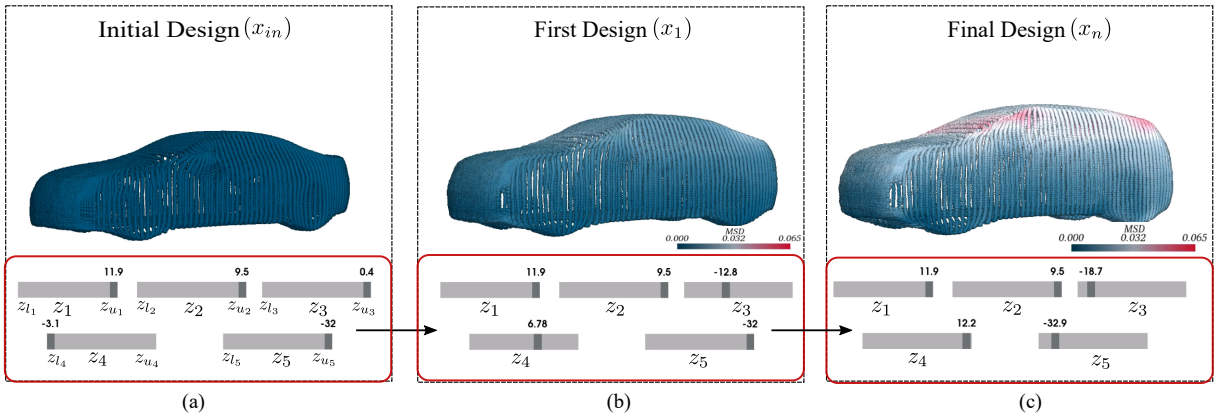


Fig. 6.8 Visualization of the interactive framework with personalized slider ranges to change the design topology from the initial design to a SUV design. a) An initial design with customized slider ranges derived from (Fig. 6.6). b) The first point cloud design (x_1) generated with z_3 and z_5 latent variables modification. c) The final design (x_n) generated by moving the sliders.

6.3.3 Framework for Performance Prediction of 3D Designs

A second set of experiments is performed to demonstrate the applicability of this interactive framework to provide real-time structural, aerodynamic or other functional performance estimation of the 3D designs by projecting the performance measure value onto each point of the 3D point cloud as color maps. The performance measures are predicted using trained surrogate models (Section 4.2.5).

For this task, the interactive GUI combines both the trained PC-VAE and the MLP as a surrogate model (Fig. 6.9). The two outputs of the MLP are normalized volume (V) measures and normalized drag coefficient (F_x) between 0 and 1. Based on the selection of performance of interest by the designer, the pre-trained model provides a performance measure (V or F_x) feedback to the designer through the user interface (Fig. 6.9).

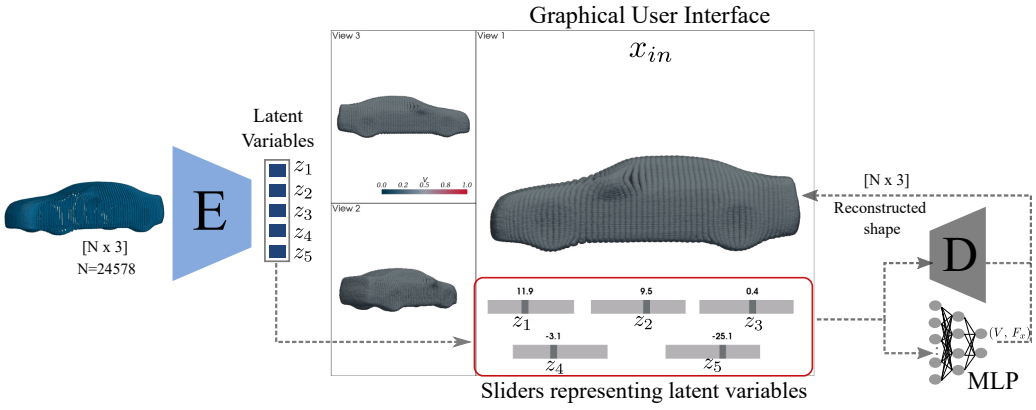


Fig. 6.9 Schematic overview of the interactive framework for 3D shape volume prediction.

To give an overview of the organization of the 3D designs on the latent manifold concerning performance measures, the 5D latent representations of the shapes in the dataset

are visualized in 2D representation using UMAP (Fig. 6.10). The 2D representation shows a gradual change in color, which signifies volume (V) in Fig. 6.10a and drag coefficient in Fig. 6.10b. This shows that exploring the distinct regions in the latent manifold and sampling latent representations from these regions will generate 3D designs with target performance measures.

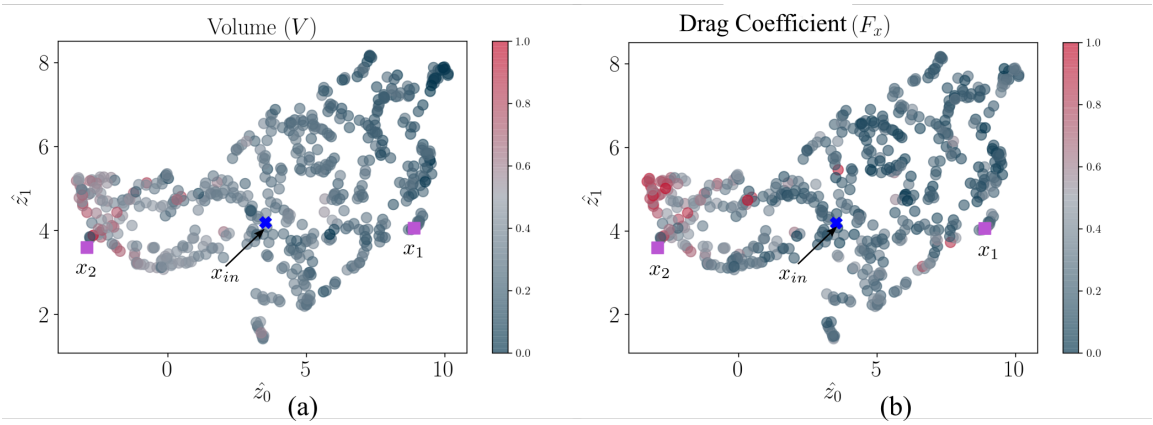


Fig. 6.10 2D latent representation of 3D car shapes organized with respect to two performance measures: a) Volume and b) Drag coefficient.

On launching a second GUI interface, the user sees the point cloud representation of the initial chosen design (x_{in}) with the normalized volume (V) projected as a color-map, and the sliders at the bottom of the GUI represent the latent variables of x_{in}. To illustrate the use of the framework for faster shape exploration along with real-time performance estimation of the 3D designs, we gradually changed the latent variables of the initial design x_{in} to generate the updated shape x_1 and x_2 (Fig. 6.11).

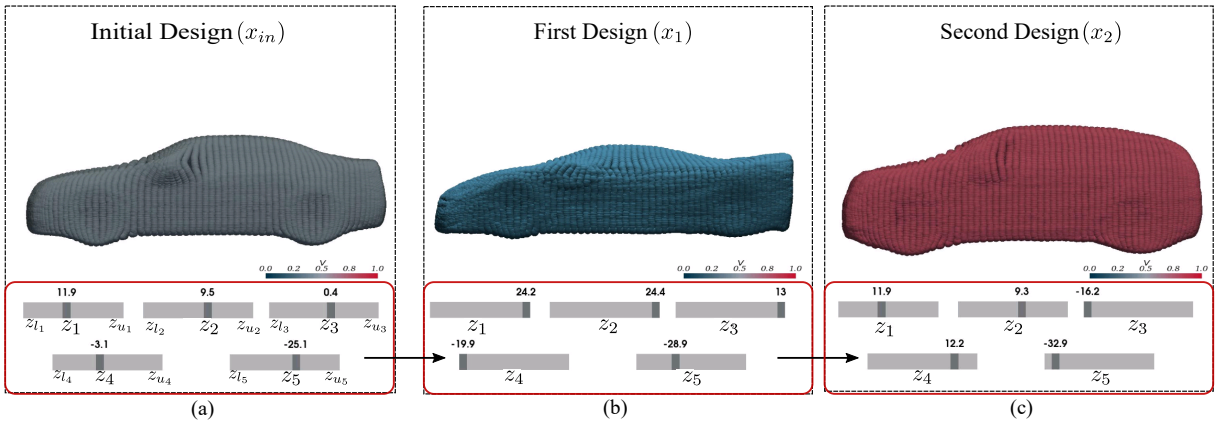


Fig. 6.11 Visualization of the interactive framework for predicting volume (V) value and its projecting onto the 3D point cloud representation as color-maps: a) Initial point cloud design x_{in} with $V = .41$. b) The first design (x_1) represents the final sportscar design from Fig. 6.7.c with $V = 0.22$. c)The second design (x_2) represents the final SUV design from Fig. 6.8.c with $V = 0.65$.

The latent variables of x_{in} are changed in such a way that the shape x_1 represents the similar sports car shape from Fig. 6.7c. The overall geometry of the car shape shrinks to

x_1, indicating the volume of x_1 reduced from x_{in}. Similarly, in the next step, the latent variables of x_1 are modified to generate the shape x_2, which represents the similar SUV design from Fig. 6.8c. In the SUV design (x_2) the structure of the car shape expands from the first shape x_1, thus the volume in x_2 is much higher compared to x_1 and x_{in}. The positions of these 3D shapes (x_{in}, x_1, and x_2) are visualized in the latent representation (Fig. 6.11), and all the 3 shapes are observed to be in distinct regions in the 2D latent manifold, verifying the difference in the volume color maps. Hence, based on the real-time performance feedback of 3D designs, the designer can explore any region of the latent manifold to achieve a targeted performance design.

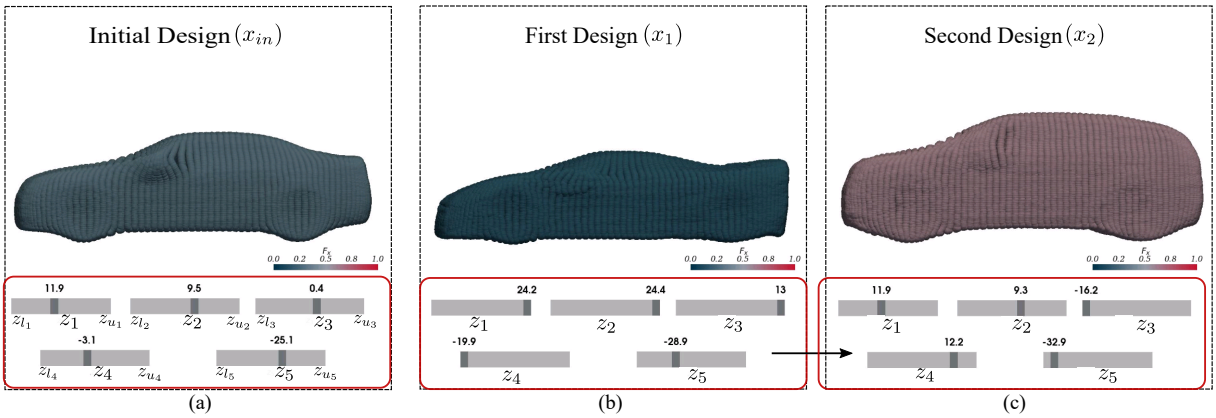


Fig. 6.12 Visualization of the interactive framework for predicting drag coefficient (F_x) value and its projecting onto the 3D point cloud representation as color-maps: a) Initial point cloud design x_{in} with $F_x = .25$. b) The first design (x_1) represents the final sportscar design from Fig. 6.7.c with $F_x = 0.18$. c) The second design (x_2) represents the final SUV design from Fig. 6.8.c with $F_x = 0.61$.

Similar to the volume performance prediction (Fig. 6.11), the usability of the framework for normalized drag coefficient estimation (F_x) is shown (Fig. 6.12), which has higher significance for critical decision-making in computational design studies. Figure 6.12 shows the three designs from Fig. 6.11, but color-coded with their respective normalized drag (F_x). Additionally, to verify the accuracy of the MLP, the R-square(R^2) scores are calculated for the true and predicted volume (V) and drag coefficient (F_x) values of the 3 shapes (Fig. 6.11 and 6.12). A higher R^2 score is observed for volume (0.84) compared to the drag coefficient (0.71), suggesting that volume estimation is an easier task compared to predicting non-linear drag coefficient, which is in line with the finding in [123]. Therefore, this experiment illustrates the usability of the proposed framework to generate shapes with target performance measures.

6.3.4 Summary

In this section, a novel interactive framework is presented, which is named cooperative design system (CDS). To build this framework, the notion of experience embedded in the

existing CAE models is learned using the PC-VAE to generate the latent manifold of the 3D designs. In the context of an automotive design optimization task, the CDS framework assists and guides the designer in exploring the latent manifold and helps in generating 3D designs based on their targets.

To demonstrate the guidance that the proposed framework can provide, a design task is formulated (Section 6.2), where the designer starts the task with an initial design (x_{in}) and gradually modifies it to reach a representative targets (x_{T1} or x_{T2}). The interactive framework provides guidance in two forms: First, based on the designer's target, it provides a personalized range of the latent variables, which guide the designer towards the target region in the latent manifold. Second, the framework predicts performance measures of the generated shapes based on the position in the latent manifold, which allows the designer to consider performance measures in different domains.

6.4 Assistance for Predicting Next Design Step Modification

This section focuses on introducing a second CDS framework that exploits the human experience, i.e., engineers' approach to performing design optimization tasks. For this experiment, a similar design optimization scenario is considered (Section 6.2). To gather human experience data, FFD is used for the shape manipulation task, which can modify the initial design to match the target design. Tracking the changes of the FFD parameters from the start till the 3D design is finalized, helps to generate sequences that signify the change of these parameters over each iteration. Hence, the aim of this set of experiments is to train a neural network-based model on existing successful shape manipulation sequences and use this trained model to predict potential future design steps from the modification currently applied.

The rest of the section is organized as follows: The experimental setup describes the method to gather the sequential data and the architecture and to train the LSTM model, which is used to learn the generated sequential data (Section 6.4.1). The trained LSTM model is used to predict future design steps (Section 6.4.2), based on the history of currently performed modifications for the given design task in Section 6.2.

6.4.1 Experimental Setup

To track the process of shape manipulation for the considered optimization scenario (Fig. 6.2), a human designer has to modify an initial shape such that it matches predefined target shapes. FFD parameterization is utilized to capture the designer's approach of performing the design modification from the initial shape to the target shape. In principle,

there are many engineers and designers in the automotive industry and they generate enough data to create a large dataset of shape manipulation sequences. However, currently this data is not available. Hence, for this thesis, we utilize an alternative approach of a target shape matching optimization (TSMO) method to generate these user approach sequences. Prior research on addressing the process of generating substantial amounts of human-like design sequences in a monotonic manner, shows that evolutionary algorithms can mimic human user behavior by adjusting the optimal hyper-parameters [140]. Hence, in this research, a target shape matching is used as a proxy for the design process, where a human user operates on the initial shape (x_{in}) and changes it with the intention to match the design target (x_{T1} and x_{T2}) (Fig. 6.1).

Shape Representation using FFD

Free-form deformation: The most popular deformation approach is standard free-form deformation (FFD) [10], which serves as a modeling tool which wraps the space surrounding an object and thereby transforms the object indirectly (Fig. 6.13). FFD embed the geometric model that has to be deformed into a parallelopipedical 3D lattice. In the first step, the representation of the 3D geometry is transformed from the Cartesian space $(x, y, z) \in \mathbb{R}^3$ to the lattice parametric space $(s, t, u) \in \mathbb{R}^3 | (s, t, u \in [0, 1])$ based on a reference point P_0 in the Cartesian space.

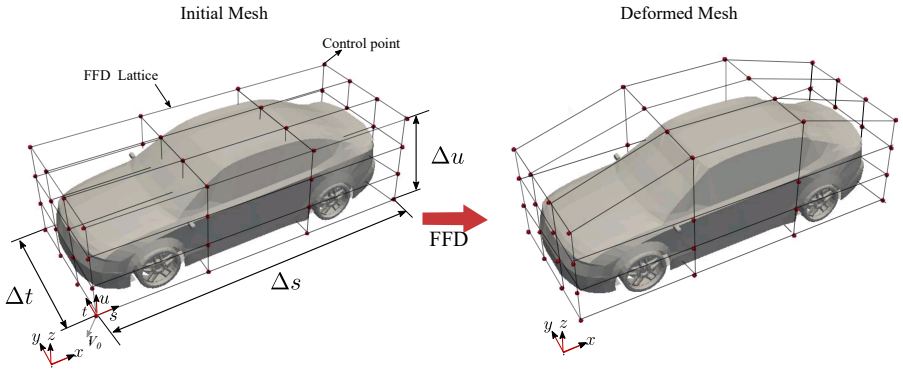


Fig. 6.13 Schematic representation of a car shape embedded in FFD lattice.

For example, a geometry is represented as a polygon surface mesh, $X = (G, P)$, where the mesh connectivity is described by a graph $G = (V, E)$ with mesh vertices V and $|V| = N$, edges $E \subseteq V \times V$, and 3D coordinates P. The geometry is embedded into the control lattice's local coordinate system,

$$\vec{v} = \vec{v}_0 + s\vec{s} + t\vec{t} + u\vec{u} \tag{6.1}$$

where $\vec{s}$, $\vec{t}$, and $\vec{u}$ are unitary vectors defining the coordinate system, $\vec{v}_0$ describes the system's origin, and s, t, and u, are the resulting local coordinates. Within the uniformly

spaced lattice, the control points, P_{ijk} are defined and displacing the control point deforms the geometry according to the Eq. 6.2.

$$\mathbf{v_{FFD}} = \sum_{i=0}^{l} \binom{l}{i} (1-s)^{l-i} s^i \left\{ \sum_{j=0}^{m} \binom{m}{j} (1-t)^{m-j} t^j \left[\sum_{k=0}^{n} \binom{n}{k} (1-u)^{n-k} u^k P_{ijk} \right] \right\} \tag{6.2}$$

where $\mathbf{v_{FFD}}$ is the deformed surface mesh vertex, and l, m, n are the number of control planes in $\vec{s}$-, $\vec{t}$-, $\vec{u}$-direction, respectively. Each plane defines a set of control points that are moved simultaneously.

Shape-parameterization using FFD: To gather a large quantity of sequential data that represents human experience of design modifications, we utilize the FFD for performing the design task modification from Section 6.2. The initial shape (x_{in}) is parameterized using FFD, which is then modified to match the target shape (x_{T1}), representing a sports car. In this experiment, the initial design is parameterized using an FFD lattice comprising of $64(4, 4, 4)$ control planes i.e., 192 control points ($cp_i, i = 1, 2, .., 192$) uniformly distributed along the x, y, and z axis (Fig. 6.14).

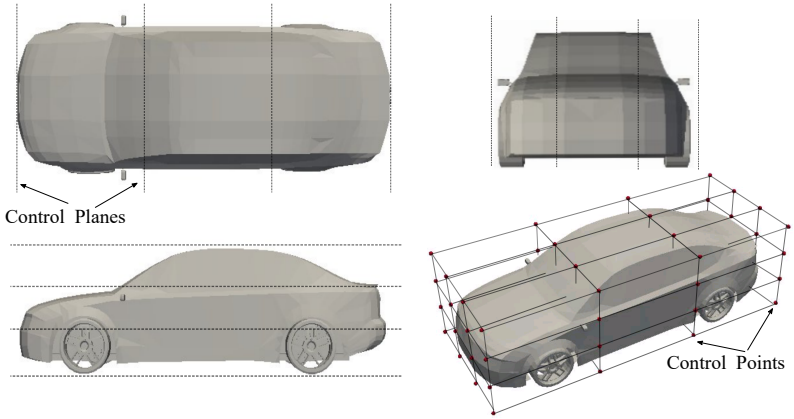


Fig. 6.14 Control points used for free form deformation of the car shape.

Optimization Algorithm Used for Data Generation

To avoid the manual generation of a large data set of shape deformation sequences, a synthetic data set is generated by solving the target-shape matching task using a computational optimization approach. In such an optimization task, the objective is to modify an initial shape (x_{in}) represented by a set of FFD control points, cp_i ($i \in 1, .., 192$), to match the target shape x_T by minimizing a distance metric (x_j, x_T), which measures the difference between the shapes. The metric (x_i, x_T) is a function based on point-wise distance, e.g., Chamfer Distance (CD).

$$\begin{aligned} &\min \quad CD(x_i(cp_i), x_T(cp_i)) \\ CD(x_j, x_T) &= \sum_{p_i \in x_j} \min_{p_o \in x_T} \|p_i - p_o\|_2^2 + \sum_{p_o \in X_T} \min_{p_i \in x_j} \|p_i - p_o\|_2^2 \end{aligned} \tag{6.3}$$

The objective function (Eq. 6.3) is minimized using the covariance matrix adaptation evolutionary strategy (CMA-ES), which is suitable for small population sizes and has a low number of parameters [106]. The CMA-ES utilizes a (μ, λ) strategy, where the chosen population size λ is set to 10, and the number of parents μ is set to 3. The optimizations are performed over 100 generations for 150 different target shapes. The target shapes (x_T) are selected from car classes: "SUV" and "sports car". 75 shapes from each of these classes are used as target shapes for this optimization task, where each shape is sampled with 24578 points using the shrink-wrapping (SWM) technique. In each optimization run, a 192-D design sequence is generated that represents individual control point movements, cp_i $(i = 1, 2, .., 192)$. The sequences denote the location of the control points in each generation of the optimization runs.

Recurrent Neural Network Model

Architecture of LSTM: For learning sequences from the design modifications, a LSTM is trained on the 192D sequences describing control point positions. The neural network has 192 inputs, a hidden layer with 256 LSTM blocks or neurons, followed by a fully connected output layer having 192 cells, corresponding to the number of desired outputs. The network was trained using the Adam optimizer [76] and with a learning rate of 0.01. Other parameters for training the network, such as batch size and number of epochs, were set to 32 and 128, respectively. The LSTM network was implemented using the Keras Framework [141] with a TensorFlow back-end [142] and mean square distance is utilized as a loss function for training.

Data set pre-processing steps: To train the LSTM network for prediction, the recorded design sequences are converted into individual *samples* using a sliding window of length w and step-size 1, leading to a three-dimensional tensor with dimensions *samples*, *time steps* and *features*. Here, each sample corresponds to one window of w time steps and features denote the positions of the 192 control points, cp_i. By associating each window with its next actual time step, a supervised learning task is formulated to predict the next time step from the window of the immediate w past time steps. Based on the w steps input data, the LSTM output layer predicts the 192D vector, which represents the next positions of the control points.

The window length w was determined by choosing the maximum lag, d, at which the partial auto-correlation function (PACF) of each design sequence feature decayed below the threshold $1.96/\sqrt{T-d}$. Here T is the length of the time series and 1.96 defines the critical alpha level at 5 under the null hypothesis that the PACF is zero (assuming the PACF asymptotically follows a standard normal distribution). The window length of w is determined using the PACF of the sequence. Through cross-validation, the w size is set to 50, for which the prediction accuracy was highest in the training data.

Training of LSTM: The training of the LSTM network is similar to a regression problem, where the input sequence with window size w is mapped to the values at $w+1$. The LSTM is trained by minimizing the mean square distance loss function. Given an input sequence of width w to the trained model, it predicts control point positions at $w+1$, which is a 192D vector.

Out of 150 design sequences that are gathered from the TSMO task, the sequence data is divided into two different splits: First, a 90% training and 10% test set, i.e., 15 sequences out of 150 sequences are kept as held out sequences for testing. Second, the data set is split into a 80% training and 20% test set, where 30 sequences are held aside for testing purposes. For evaluation, the prediction errors are calculated on the training set and the test set. As a baseline (BL), a moving average baseline model is considered that performs single-step prediction as the average over the past w input samples.

6.4.2 Single-Step Prediction

The LSTM model trained to perform single-step prediction outperforms the baseline model for all 192D features, showing that the LSTM network successfully learned the multivariate design sequences from the data. To estimate the accuracy of the LSTM network's prediction, a mean absolute error and mean square error are compared between true values and model-prediction (Table 6.1). The LSTM network was able to predict the multivariate output, i.e., the joint movement of all 192D control points.

Table 6.1 Mean absolute error (MAE) and mean squared error (MSE) plus standard deviations for baseline model (BL) and single step LSTM for 15 and 30 test set sequences

		BL		LSTM	
		Training set	**Test set**	**Training set**	**Test set**
15/135 **split**	**MAE**	0.182 ± 0.004	0.191 ± 0.002	0.147 ± 0.003	0.119 ± 0.004
	MSE	0.058 ± 0.004	0.040 ± 0.001	0.030 ± 0.005	0.024 ± 0.007
30/120 **split**	**MAE**	0.162 ± 0.003	0.173 ± 0.002	0.123 ± 0.003	0.132 ± 0.001
	MSE	0.057 ± 0.005	0.050 ± 0.002	0.026 ± 0.004	0.040 ± 0.003

Figure 6.15 shows two examples of the predictions result by the LSTM i.e., given the design sequence of modification from step 1 to step 50 as input to the LSTM, the model

predicts the location of the control points at step 51. Based on the LSTM's prediction, the geometry is generated by reverse engineering. The predicted shape at step 51 is overlapped with the shape at step 50, to demonstrate the region of modification on the current step.

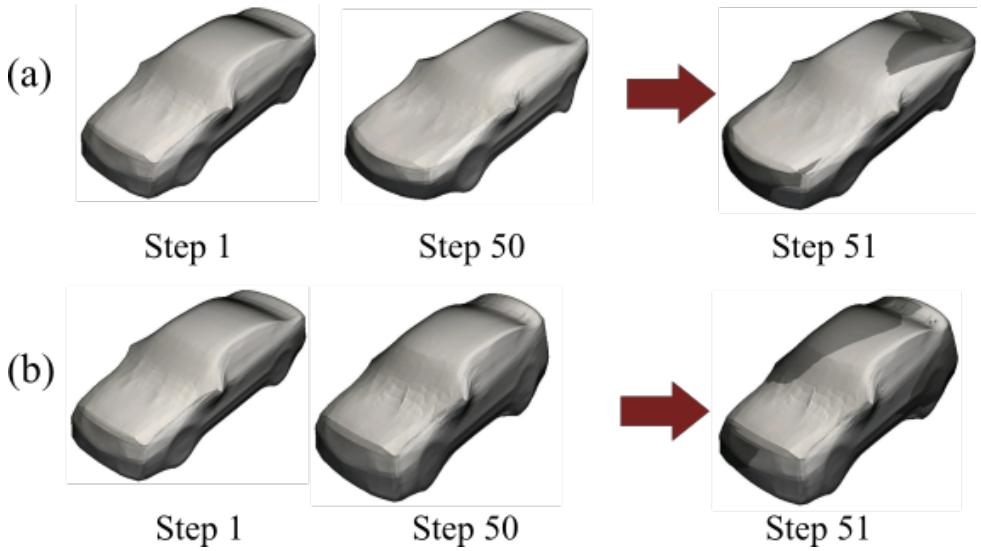


Fig. 6.15 Design changes predicted by the LSTM model following two different target designs (a) Sports car, (b) SUV. For each of the design processes, deformed car shapes from time-step 1 (left) to time-step 50 (middle) form a representative window. The LSTM model prediction for time-step 51 is superimposed on the car shape from time-step 50 (right). For (a) the LSTM model proposes to move the right part outward; and for (b) the LSTM model proposes to move the left side of the roof structure upward.

6.4.3 Summary

The motivation of this research was to learn the notion of designers' experience from past design optimization tasks and develop an interactive framework, which will guide (novice) designers by predicting the next potential design modification direction. To build this framework, first, a large dataset of sequences is gathered, which represents past design modifications in an optimization task. To track design modification in an optimization, 3D designs are parameterized with FFD and the parameters of FFD are tracked from the beginning to the end of the optimization process. The optimization parameters are tuned such that the optimizer produced design sequences that mimicked the design process as carried out by an experienced human user.

Secondly, a state-of-the-art LSTM model is used for learning 3D design sequences to predict future design steps. This is a promising step for modeling design experience using a human-machine system that can cooperatively support an engineer in a design task. The LSTM-based prediction task is performed to explore how the trained model may be used to guide a (novice) designer in future design processes by proposing potential next design steps based on the history of currently performed design modifications.

6.5 Conclusion

The research in this chapter is mainly based on the following published research papers [143, 140, 144] and has been extended based on more recent findings. This chapter shows an important step in analyzing the capability of the neural network models for modeling prior design experiences in order to provide guidance to the engineers and designers during the design process.

The analyses in this chapter address the research question $RQ6$, which is answered in the following paragraph.

RQ 6. How to exploit and learn engineering experience from past design optimization tasks using DL models? And how to utilize the trained DL model for formulating different assistance to collaboratively help designers in a present design ideation task?

The perception of engineering experience is exploited in two different methods in this research: First, learning the experience embedded in the engineering data and second, learning the engineer's or designer's experience. We introduce two frameworks to model these two distinct types of data and provide useful guidance to the designer in a digital design development process. For demonstration of these frameworks, a realistic design optimization task is considered, and the two frameworks are used to generate different forms of assistance to guide the user.

The first framework is designed to model the experience embedded in the 3D data, where the generative PC-VAE is used to create a low-dimensional data-manifold (latent manifold) for easy exploration of 3D designs. An interactive graphical user interface (GUI) is implemented, where sliders are associated to the latent variables of the PC-VAE. This complete framework is named a cooperative design system (CDS). The CDS helps the designer in changing an initial design to a design that matches his/her target, along with estimated performances of different domains. However, there are open challenges that need to be addressed. From a representation point of view, designers may need to recreate CAD/E models from the point cloud representations, which further needs reverse engineering either manually or using shrink-wrapping methods.

The second framework focuses on modeling human-experience data and guides the designer in real-time by predicting the next step for design modification based on the currently applied modifications. Here, we considered human-experience data as sequential data that represents engineers' approach to performing shape modifications for a particular design optimization task. As an initial approach to mimic human experience data, a computational optimization approach is used to solve the similar design optimization task, and the design modifications over the

optimization runs are recorded. Next, LSTMs are trained on the multi-variate design modification sequences and the trained model is used to predict the next step of design modification based on the current applied changes. Hence, guiding the designer in modifying the initial design to achieve the design that matches his/her target. The LSTMs' ability to generate novel design steps and to provide next-step predictions, make them promising candidates for learning design-modification data. However, one limitation of this approach is the lack of gathered data that represents a design method for more complex design modification tasks. Therefore, future research focusing on modeling human-experience, should consider at least partial data collection from engineers and designers.

Nevertheless, the results in this chapter show a promising step towards using geometric deep learning-based systems as an additional tool in the automotive design exploration phase, which provides the benefit of combining knowledge from data-driven AI models.

Chapter 7

Discussion and Outlook

This chapter provides the conclusions and outlook drawn from the research findings that led to this thesis.

7.1 Summary and Discussion

Experts in any domain intuitively exploit experience from previously solved different optimization problems to address new and more challenging tasks. Exploiting past experience allows us to propose better initial solutions in an optimization setting rather than starting without any prior information about the target problem. Hence, to transfer the notion of experience between different automotive design tasks, we envision a cooperative design system (CDS) that learns the notion of experience from past design optimization tasks and aims to assist designers with potential alternatives. This supports as an assistance tool for informed decision making by the designers. The notion of experience is defined using two methods: First, the engineering experience embedded in the 3D data is gathered from previous optimization tasks; and the second method refers to capturing the human experience of performing design optimization tasks.

The availability of a large number of digital data generated during the past design optimizations makes it feasible to learn the embedded experience in the 3D data using deep learning networks. Yet, the high-dimensionality and unstructured nature of the data poses challenges for state-of-the-art machine learning algorithms. Recent advancements in geometric deep learning (GDL) methods extend the application of ML algorithms to both structured and unstructured 3D data. In GDL, deep generative models are used for learning low-dimensional representations of computer-aided engineering (CAE) models. Hence, applying the GDL technique to exploit the notion of experience is a promising approach for an experience-based engineering optimization framework, which is the area of focus of this thesis.

As the representation of CAE data is not canonical, a first step for learning design features from CAE data is to define the most suitable representation of 3D designs for training a deep generative model. Hence, in Chapter 2, a review of the types of 3D representation is presented, which answers *RQ*1: *"Which 3D representation is suitable for efficient training of deep learning models?"*. In the context of selecting a suitable 3D representation that is easily scalable and captures details of 3D shapes, the point cloud representation of 3D geometries provides more flexibility and requires less pre-processing effort compared to polygonal meshes. Additionally, this chapter provides a review of the popular deep generative models, out of which the variational autoencoder (VAE) proves to be the most suitable generative architecture for learning 3D data. This model provides the flexibility to generate new designs, which is a key element for creative design exploration tasks. Therefore, from the reviewed representations, 3D point cloud representations of CAE models are most suitable for learning using a VAE.

In Chapter 3, an architecture of a 3D point cloud variational autoencoder (PC-VAE) is proposed for learning design point cloud representations of CAE models, which answers the research questions (*RQ*2): *How well can a generative model learn a compact representation of 3D designs for engineering optimization? And is this model suitable for generating diverse unseen designs that are useful for engineering applications?*. The shape-generative performance of PC-VAE is comparable to the state-of-the-art point-based generative networks [60, 70]. To use the PC-VAE for engineering optimization tasks, the *realism* and *diversity* of the shapes generated by this model are evaluated using pre-defined quantitative metrics. These experiments show that the PC-VAE generates a suitable low-dimensional latent space for 3D designs, which allows sampling of novel designs that the model has not seen before. Additionally, it is shown that performing gradient-based optimization on the latent space of the PC-VAE helps in generating 3D designs from unexplored regions in the latent search space.

However, the generation of car designs ideally involves both aesthetic design targets and engineering criteria, which leads to the question (*RQ*3): "*How can the data-driven latent space be utilized to generate design solutions that satisfy multiple criteria?*". To answer this question, a real-world-inspired, multi-objective optimization task is set up. Two scenarios are considered where the designer can incorporate preference prior to the start of the optimization task, or posterior selection of preferred solutions generated from the optimization task. The optimization task is performed in the low-dimensional search space of the PC-VAE and the latent space knowledge is exploited to improve the convergence of optimization algorithms in both optimization scenarios. In the first scenario, the designer assumes to have no prior preference information and aims to utilize the optimization task to generate a diverse range of design solutions to select from it later. The latent space knowledge is exploited to initialize the optimization algorithm (*Lerp-seed*), such that the optimization converges faster and generates more diverse solutions. These generated

diverse solutions are returned as suggestions to the designer to select the preferred one. The second scenario mainly resembles experienced designers, who can specify the desired preference information prior to the start of the optimization task. The weighted-sum method is used to solve the optimization process, where an adaptive strategy is proposed to determine the weights such that the final generated solution of the optimization matches the designer's preference. Hence, exploiting the latent space knowledge improves the convergence time of the optimization task and generates a target solution faster.

Often, the evaluation of the objective function in a design optimization requires time-consuming simulation runs. To replace costly objective function calls with surrogate models, the characteristics of the latent variables are evaluated from a more practical perspective to answer the following questions ($RQ4$): "*Does the latent space of the deep generative models hold relevant information about the structural or functional properties of 3D designs? Can a surrogate model be trained on the latent manifold to replace expensive performance simulation or function evaluations in a design optimization problem?*". To answer $RQ4$, the latent space of the PC-VAE is analyzed from an information encoding perspective. Since the PC-VAE is built upon the architecture of the autoencoder, the latent space of both models is compared to analyze the effect of additional regularization of the PC-VAE. Information theoretic measures are utilized to analyze the information content in the latent variables to represent geometric and functional performances of 3D designs. The main outcome of the experiments is that training the PC-VAE with the KL-divergence loss function helps to generate a probabilistic continuous latent space. But this additional regularization loss function reduces the performance information content in the latent variables of the PC-VAE. It is also shown that learning on organized point clouds using the mean-square distance (MSD) between corresponding points as loss functions improves the quality of the shapes generated by the PC-VAE. Learning on organized point clouds requires lower computational effort, and the models are able to generate shapes that maintain the organization of the point clouds. This leads to the generation of simplified meshes on the output point clouds, reducing the post-processing time. Surrogate models trained on the latent space of PC-AE and PC-VAE, are able to predict the volume and aerodynamic drag coefficient of 3D shapes. Hence, these surrogate models can be used to replace costly evaluation of objective functions. Furthermore, the advantage of the PC-VAE is demonstrated in addressing the data-limitation by sampling realistic designs from sparsely sampled regions of the data set.

Previous experiments showed that the latent variables were suitable for modifying 3D designs. Yet the interpretation of a particular latent variable mapping to a distinct region of the 3D designs was lacking, which leads to the question ($RQ5$): "*How can the design features learned by a deep generative model represent different interpretable aspects of 3D designs? How can we transfer the unique parts or design features of a 3D shape to another shape, such that the recombination of design features from two distinctive shapes*

generates a novel shape combining the essence of both 3D designs?". To address this challenge, a novel deep-generative model that builds upon the architecture of the PC-VAE is proposed: Split-AE. The novel architecture combines two classifiers to learn two sets of latent variables, where the first set helps to understand the whole underlying structure of the 3D shape to discriminate across other semantic shape categories (content). The second set of variables represents distinctive parts of the 3D designs that allow grouping of shapes into shape classes (style). The proposed architecture is further used to answer the question "*How can we transfer the unique shape parts of a 3D design to another design, such that the recombination of design features from two distinctive designs generates a novel shape that combines the essence of both 3D designs?*". A series of experiments were performed to demonstrate the network's ability to generate augmented shapes that combine the essence of two different shapes; i.e., combining the style latent variables of a target shape with the content latent variables of a source shape. This approach of disentangling feature transfer between 3D designs with minimum supervision, helps in generating novel shapes that are modified versions of the existing designs.

Finally, once the architecture and capabilities of the PC-VAE are better understood, the following question targets the initial vision of a cooperative design assistance system (*RQ*6): "*How to exploit and learn engineering experience from past design optimization tasks using DL models? And how to utilize the trained DL model for formulating different assistance to collaboratively help designers in a present design ideation task?*". To answer these questions, we exploited the perception of experience in two different methods. The first approach aims at learning the experience embedded in the engineering data, and the second, referring to learning from engineering experience. Two frameworks are introduced to model these two distinct approaches. Both provide useful guidance in a design process. The first framework is a data-driven cooperative design system (CDS) that serves as an assistance system for designers to explore design ideas. The data-driven PC-VAE is utilized to create a low-dimensional data-manifold (latent manifold) for easy exploration of 3D designs through our proposed framework. For a real-world-inspired design task, the CDS provides assistance for exploring the latent manifold to generate target-oriented designs, along with real-time performance predictions of the generated shape. Providing this assistance enables the designers to explore designs taking into account the data-driven knowledge and estimated performances of different domains. The second framework aims to model human experience and guides the designers in real-time by predicting the next step for design modification based on the currently applied modifications. To develop this framework, sequential data is collected that represents an engineer's approach to shaping modifications for a particular design task. A computational optimization method is proposed to mimic human experience data, such that enough data could be generated to train deep learning models. Next, a long short-term memory (LSTM) network is trained on

the sequential data, such that the model can provide guidance to the designer in modifying a given 3D design to match his/her target.

7.2 Outlook and Future Work

The contribution of the work in this thesis covers a limited scope of the research on experience-based representation learning. Hence, several challenges remain unsolved and there is scope for improvement of the present proposed methods.

One of the limitations of using GDL methods for engineering applications is the lack of engineering-tailored data sets. ShapeNetCore [18] provides a large collection of 3D models that allowed us to advance the research in GDL, but there is a limitation on the information of the model properties and class labels.

Another limitation in line with training the variational autoencoder with shapes from one domain needs several trial-and-error runs for balancing the reconstruction and KL-divergence loss functions. Hence, without expertise variational autoencoder needs excessive re-training to adapt it to shapes from other domains. One possible improvement to the training of variational autoencoder could be using transfer learning to learn the latent representations from one data domain and using it to the re-parametrization of the latent variable in another domains [145].

Further, the Split-AE proposed in this research is regarded as a first step towards a generic model for feature disentanglement of shapes across domains. Also, the car and airplane shapes were utilized in this research for evaluation of the proposed model and therefore did not use the model for style transfer across domains. Therefore, in future work, the aim is to identify style as a global feature common to car shapes, e.g., from a brand and investigate 3D style transfer between cross-domain shapes. Additionally, in the line of style transfer between two 3D designs, Split-AE always needs a source-target 3D design pair. Following the idea of optimization in multiple latent spaces and transfer of knowledge from one latent space to another, an alternative way of generating a 3D design with specific content and style would be to perform optimization in the content and style space [146].

Lastly, the contributions of the research in this thesis cover the first prototypical demonstration of an experience-based cooperative design assistance system. However, there are several challenges that remain unsolved, and the methods proposed in the present research have to be improved to adapt to a wider range of applications. Future work should also focus on integrating the CDS framework into the real-world automotive design process and gathering feedback from designers. Additional steps should include re-training and updating the DL models with 3D geometries tailored for particular automotive design exploration. Finally, the biggest challenge of developing a CDS framework that guides

the designer and the engineers in the cooperative design process is the lack of data that represents the human approach of complex design modifications.

List of figures

List of tables

References

[1] S. J. Louis and J. McDonnell, "Learning with case-injected genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 4, pp. 316–328, 2004.

[2] P. Cunningham and B. Smyth, "Case-based reasoning in scheduling: Reusing solution components," *International Journal of Production Research*, vol. 35, no. 11, pp. 2947–2962, 1997.

[3] L. Feng, Y. S. Ong, S. Jiang, and A. Gupta, "Autoencoding evolutionary search with learning across heterogeneous problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 760–772, 2017.

[4] M. Pelikan and M. W. Hauschild, "Learn from the Past: Improving Model-Directed Optimization by Transfer Learning Based on Distance-Based Bias," Missouri Estimation of Distribution Algorithms Laboratory, University of Missouri, Tech. Rep. 2012007, 2012.

[5] J. Cagan, M. I. Campbell, S. Finger, and T. Tomiyama, "A Framework for Computational Design Synthesis: Model and Applications," *Journal of Computing and Information Science in Engineering*, vol. 5, no. 3, pp. 171–181, 2005.

[6] M. I. Campbell, "A Generic Scheme for Graph Topology Optimization: Recent Results," *NSF Engineering Research and Innovation Conference*, no. June, pp. 1–10, 2009.

[7] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," *The Fifth International Conference in Genetic Algorithms*, vol. 93, pp. 416–425, 1993.

[8] G. Renner and A. Ekárt, "Genetic Algorithms in Computer-Aided Design," *CAD Computer Aided Design*, vol. 35, no. 8, pp. 709–726, 2003.

[9] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[10] T. W. Sederberg and S. R. Parry, "Free-form Deformation of Solid Geometric Models," *ACM Special Interest Group on Computer Graphics and Interactive Techniques*, vol. 20, no. 4, pp. 151–160, 1986.

[11] S. Menzel and B. Sendhoff, "Representing the change - Free form deformation for evolutionary design optimization," *Studies in Computational Intelligence*, vol. 86, pp. 63–86, 2008.

[12] Y. J. Lee, C. L. Zitnick, and M. F. Cohen, "ShadowDraw: Real-time user guidance for freehand drawing," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 1–10, 2011.

[13] D. Ha and D. Eck, "A Neural Representation of Sketch Drawings," in *6th International Conference on Learning Representations, ICLR*, 2018.

[14] Autodesk, "Maya," 2019. [Online]. Available: https:/autodesk.com/maya

[15] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[16] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning Mesh-Based Simulation with Graph Networks," in *International Conference on Learning Representations, ICLR*, 2020, pp. 1–18.

[17] C. C. Wang, "Realizing CAD/CAM by Polygonal Meshes," *Computer-Aided Design*, vol. 43, no. 4, p. 457, 2011.

[18] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University - Princeton University - Toyota Technological Institute at Chicago, Tech. Rep., 2015.

[19] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018, pp. 4490–4499.

[20] T. Friedrich, N. Aulig, and S. Menzel, "On the Potential and Challenges of Neural Style Transfer for Three-Dimensional Shape Data," in *International Conference on Engineering Optimization, EngOpt*, 2019, pp. 581–592.

[21] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017, pp. 6620–6629.

[22] P. S. Wang, Y. Liu, Y. X. Guo, C. Y. Sun, and X. Tong, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[23] I. Carlbom, I. Chakravarty, and D. Vanderschel, "A Hierarchical Data Structure for Representing the Spatial Decomposition of 3-D Objects," *IEEE Computer Graphics and Applications*, vol. 5, no. 4, pp. 24–31, 1985.

[24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017, pp. 652–660.

[25] C. R. Qi, O. Litany, K. He, and L. Guibas, "Deep Hough Voting for 3D Object Detection in Point Clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9276–9285.

[26] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu, "PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation," *arXiv*, 2018.

[27] H. Y. Chiang, Y. L. Lin, Y. C. Liu, and W. H. Hsu, "A Unified Point-Based Framework for 3D Segmentation," *Proceedings - 2019 International Conference on 3D Vision, 3DV 2019*, no. Table 2, pp. 155–163, 2019.

[28] G. Yang, X. Huang, Z. Hao, M.-Y. Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3D point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE International Conference on Computer Vision*, jun 2019. [Online]. Available: http://arxiv.org/abs/1906.12320

[29] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, "Deep Learning on Point Clouds and its Application: A Survey," *Sensors (Switzerland)*, vol. 19, no. 19, p. 4188, 2019.

[30] C. Lv, W. Lin, and B. Zhao, "Voxel Structure-Based Mesh Reconstruction from a 3D Point Cloud," *IEEE Transactions on Multimedia*, vol. 24, pp. 1815–1829, 2022.

[31] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A Deep Representation for Volumetric Shapes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[32] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veli, "Geometric Deep Learning," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2021.

[33] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, 1985.

[34] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. July, pp. 504–507, 2006.

[35] N. Zeng, Z. Wang, H. Zhang, W. Liu, and F. E. Alsaadi, "Deep Belief Networks for Quantitative Analysis of a Gold Immunochromatographic Strip," *Cognitive Computation*, vol. 8, no. 4, pp. 684–692, 2016.

[36] M. Tschannen, O. Bachem, and M. Lucic, "Recent Advances in Autoencoder-Based Representation Learning," in *Thirty-second Conference on Neural Information Processing Systems, NeurIPS*, 2018, pp. 1–25.

[37] W. Gao and C. Su, "Analysis on block chain financial transaction under artificial neural network of deep learning," *Journal of Computational and Applied Mathematics*, vol. 380, p. 112991, 2020. [Online]. Available: https://doi.org/10.1016/j.cam.2020.112991

[38] I. J. Goodfellow, J. Pouget-abadie, M. Mirza, B. Xu, D. Warde-farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[39] K. Lata, M. Dave, and N. K.N., "Data Augmentation Using Generative Adversarial Network," *SSRN Electronic Journal*, pp. 1–14, 2019.

[40] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4217–4228, 2021.

[41] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning*, vol. 12, no. 4, 2019.

[42] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," in *International conference on machine learning, PMLR*, 2017, pp. 214–223.

[43] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," *7th International Conference on Learning Representations, ICLR*, pp. 1–35, 2019.

[44] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

[45] J. Ngiam, Z. Chen, and A. Y. Ng, "Learning Deep Energy Models," in *Proceedings of the 28th International Conference on Machine Learning , ICML*, 2011, pp. 1–8.

[46] J. Xie, Y. Lu, S. C. Zhu, and Y. N. Wu, "A Theory of Generative ConvNet," in *33rd International Conference on Machine Learning, ICML*, 2016, pp. 3895–3904.

[47] S. Fine, Y. Singer, and N. Tishby, "The Hierarchical Hidden Markov Model: Analysis and Applications," *Machine Learning*, vol. 32, no. 1, pp. 41–62, 1998.

[48] D. P. Kingma and M. Welling, "Auto-encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR*, 2014, pp. 1–14.

[49] M. Ding, "The road from MLE to EM to VAE: A brief tutorial," *AI Open*, vol. 3, no. July 2021, pp. 29–34, 2022. [Online]. Available: https://doi.org/10.1016/j.aiopen.2021.10.001

[50] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference," in *31st International Conference on International Conference on Machine Learning*, 2014, pp. 1278 – 1286.

[51] M. Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2017.

[52] A. Radford, "Improving GANS using Optimal Transport," in *6th International Conference on Learning Representations, ICLR*, 2018.

[53] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled Generative Adversarial Networks," in *5th International Conference on Learning Representations, ICLR*, 2017, pp. 1–25.

[54] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks," in *5th International Conference on Learning Representations, ICLR*, 2017.

[55] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, "Stabilizing Training of Generative Adversarial Networks through Regularization," in *Advances in Neural Information Processing Systems*, 2017, p. 30.

[56] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.

[57] L. Landrieu and M. Simonovsky, "Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs," in *IEEE conference on computer vision and pattern recognition*, 2018, pp. 4558–4567.

[58] X. Wang, J. He, and L. Ma, "Exploiting Local and Global Structure for Point Cloud Semantic Segmentation with Contextual Point Representations," in *Advances in Neural Information Processing Systems*, 2019, p. 32.

[59] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data Charles," in *IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927.

[60] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning Representations and Generative Models for 3D Point Clouds," in *35th International Conference on Machine Learning (ICML)*, vol. 80, 2018, pp. 40–49.

[61] T. Rios, B. Sendhoff, S. Menzel, T. Bäck, and B. Van Stein, "On the Efficiency of a Point Cloud Autoencoder as a Geometric Representation for Shape Optimization," in *IEEE Symposium Series on Computational Intelligence, SSCI*, 2019, pp. 791–798.

[62] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "Beta VAE," *International Conference on Learning Representations*, 2017.

[63] H. Fan, H. Su, and L. Guibas, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 2463–2471, 2017.

[64] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3D Shape Segmentation with Projective Convolutional Networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017, pp. 3779–3788.

[65] C. Nash and C. K. Williams, "The Shape Variational Autoencoder: A Deep Generative Model of Part-Segmented 3D Objects," *Computer Graphics Forum*, vol. 36, no. 5, pp. 1–12, 2017.

[66] Z. Wan, Y. Zhang, and H. He, "Variational Autoencoder based Synthetic Data Generation for Imbalanced Learning," in *IEEE Symposium Series on Computational Intelligence, SSCI*, 2018, pp. 1–7.

[67] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3D Faces Using Convolutional Mesh Autoencoders," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11207 LNCS, pp. 725–741, 2018.

[68] Z. Islam, M. Abdel-Aty, Q. Cai, and J. Yuan, "Crash Data Augmentation using Variational Autoencoder," *Accident Analysis and Prevention*, vol. 151, no. July, 2021.

[69] N. Schor, O. Katzir, H. Zhang, and D. Cohen-Or, "CompoNet: Learning to Generate the Unseen by Part Synthesis and Composition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8758–8767.

[70] M. Zamorski, M. Zięba, P. Klukowski, R. Nowak, K. Kurach, W. Stokowiec, and T. Trzciński, "Adversarial Autoencoders for Compact Representations of 3D Point Clouds," *Computer Vision and Image Understanding*, vol. 193, 2020.

[71] D. Berthelot, I. Goodfellow, C. Raffel, and A. Roy, "Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer," *7th International Conference on Learning Representations, ICLR*, pp. 1–20, 2019.

[72] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *4th International Conference on Learning Representations, ICLR*, pp. 1–16, 2016.

[73] Z. Guo, H. Liu, Y. S. Ong, X. Qu, Y. Zhang, and J. Zheng, "Generative Multiform Bayesian Optimization," *IEEE Transactions on Cybernetics*, pp. 1–14, 2022.

[74] A. Tripp, E. Daxberger, and J. M. Hernández-Lobato, "Sample-Efficient Optimization in the Latent Space of Deep Generative Models via Weighted Retraining," in *Advances in Neural Information Processing Systems*, no. 33, 2020, pp. 11 259–11 272. [Online]. Available: http://arxiv.org/abs/2006.09191

[75] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[76] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015, pp. 1–15.

[77] S. Saha, S. Menzel, L. L. Minku, X. Yao, B. Sendhoff, and P. Wollstadt, "Quantifying The Generative Capabilities Of Variational Autoencoders For 3D Car Point Clouds," in *IEEE Symposium Series on Computational Intelligence, SSCI*, 2020, pp. 1469–1477.

[78] K. Li, R. Chen, G. Min, and X. Yao, "Integration of preferences in decomposition multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 48, no. 12, pp. 3359–3370, 2018.

[79] L. Li, I. Yevseyeva, V. Basto-Fernandes, H. Trautmann, N. Jing, and M. Emmerich, "An Ontology of Preference-Based Multiobjective Metaheuristics," *Neural and Evolutionary Computing*, no. March, 2016.

[80] K. Li, M. Liao, K. Deb, G. Min, and X. Yao, "Does preference always help? A holistic study on preference-based evolutionary multi-objective optimisation using reference points," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 6, pp. 1078–1096, 2020.

[81] R. Winter, F. Montanari, A. Steffen, H. Briem, F. Noé, and D. A. Clevert, "Efficient Multi-objective Molecular Optimization in a Continuous Latent Space," *Chemical Science*, vol. 10, no. 34, pp. 8016–8024, 2019.

[82] R. R. Griffiths and J. M. Hernández-Lobato, "Constrained Bayesian Optimization for Automatic Chemical Design using Variational Autoencoders," *Chemical Science*, vol. 11, no. 2, pp. 577–586, 2020.

[83] J. D. Cunningham, D. Shu, T. W. Simpson, and C. S. Tucker, "A Sparsity Preserving Genetic Algorithm for Extracting Diverse Functional 3D Designs from Deep Generative Neural Networks," *Design Science*, vol. 6, pp. 1–33, 2020.

[84] N. Srinivas and K. Deb, "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[85] J. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," *The 1st international Conference on Genetic Algorithms*, 1985.

[86] E. Zitzler and L. Thiele, "Multi-objective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[87] J. Knowles and D. Corne, "The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC*, 1999, pp. 98–105.

[88] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, p. 103, 2001.

[89] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II," in *International conference on parallel problem solving from nature*, 2000, pp. 849–858.

[90] C. García-Martínez, O. Cordón, and F. Herrera, "An Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-criteria TSP," in *International Workshop on Ant Colony Optimization and Swarm Intelligence*, 2004, pp. 61–72.

[91] J. J. Durillo, J. García-Nieto, A. J. Nebro, C. A. Coello Coello, F. Luna, and E. Alba, "Multi-objective Particle Swarm Optimizers: An Experimental Comparison," in *International conference on evolutionary multi-criterion optimization.*, 2010, pp. 495–509.

[92] C. Haubelt, J. Gamenik, and J. Teich, "Initial Population Construction for Convergence Improvement of MOEAs," in *International Conference on Evolutionary Multi-Criterion Optimization*, 2005, pp. 191–205.

[93] S. Poles, Y. Fu, and E. Rigoni, "The Effect of Initial Population Sampling on the Convergence of Multi-objective Genetic Algorithms," *In Multiobjective programming and goal programming*, pp. 123–133, 2009.

[94] E. Khaji and A. S. Mohammadi, "A Heuristic Method to Generate Better Initial Population for Evolutionary Methods," *Neural and Evolutionary Computing*, pp. 1–8, 2014.

[95] T. Chen, M. Li, and X. Yao, "On the Effects of Seeding Strategies: A Case for Search-based Multi-objective Service Composition," in *GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference*, 2018, pp. 1419–1426.

[96] G. Fraser and A. Arcuri, "The Seed is Strong: Seeding Strategies in Search-based Software Testing," in *Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST*, 2012, pp. 121–130.

[97] Y. Jin, "Effectiveness of Weighted Aggregation of Objectives for Evolutionary Multiobjective Optimization : Methods , Analysis and Applications," *Analysis and Applications*, no. x, pp. 1–32.

[98] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, and T. Zhang, "Localized Weighted Sum Method for Many-Objective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 3–18, 2016.

[99] S. Mardle and K. M. Miettinen, "Nonlinear Multiobjective Optimization," *The Journal of the Operational Research Society*, vol. 51, no. 2, p. 246, 2000.

[100] R. T. Marler and J. S. Arora, "The Weighted Sum Method for Multi-objective Optimization : New Insights," *Structural and Multidisciplinary Optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[101] S. Phelps and M. Köksalan, "An Interactive Evolutionary Metaheuristic for Multiobjective Combinatorial Optimization," *Management Science*, vol. 49, no. 12, pp. 1726–1738, 2003.

[102] M. Koksalan and I. Karahan, "An Interactive Territory Defining Evolutionary Algorithm: ITDEA," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 702–722, 2010.

[103] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired Co-evolutionary Algorithms using Weight Vectors," *European Journal of Operational Research*, vol. 243, no. 2, pp. 423–441, 2015.

[104] W. W. Chang, C. J. Chung, and B. Sendhoff, "Target Shape Design Optimization with Evolutionary Computation," *2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings*, vol. 3, no. January, pp. 1864–1870, 2003.

[105] Z. Yang, B. Sendhoff, K. Tang, and X. Yao, "Target Shape Design Optimization by Evolving B-Splines with Cooperative Coevolution," *Applied Soft Computing Journal*, vol. 48, pp. 672–682, 2016.

[106] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies." *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[107] Y. Jin, M. Olhofer, and B. Sendhoff, "Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: {W}hy Does It Work and How?" *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pp. 1042–1049, 2001.

[108] Y. Jin, T. Okabe, and B. Sendhoff, "Adapting weighted aggregation for multiobjective evolution strategies," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1993, no. D, pp. 96–110, 2001.

[109] S. C. Endres, C. Sandrock, and W. W. Focke, "A Simplicial Homology Algorithm for Lipschitz Optimisation," *Journal of Global Optimization*, vol. 72, no. 2, pp. 181–217, 2018.

[110] N. Hansen, "The CMA Evolution Strategy: A Comparing Review," in *Towards a new evolutionary computation*, 2006, pp. 75–102.

[111] D. A. Van Veldhuizen and G. B. Lamont, "On Measuring Multiobjective Evolutionary Algorithm Performance," *Proceedings of the 2000 Congress on Evolutionary Computation, CEC 2000*, vol. 1, pp. 204–211, 2000.

[112] Jason R. Schott, "Fault Tolerant Design using Single and Multicriteria Genetic Algorithm Optimization," Massachusetts Institute of Technology, Tech. Rep., 1995.

[113] L. Gräning, Y. Jin, and B. Sendhoff, "Individual-based management of meta-models for evolutionary optimization with application to three-dimensional blade optimization," *Studies in Computational Intelligence*, vol. 51, pp. 225–250, 2007.

[114] M. N. Le, Y. S. Ong, S. Menzel, Y. Jin, and B. Sendhoff, "Evolution by adapting surrogates," *Evolutionary Computation*, vol. 21, no. 2, pp. 313–340, 2013.

[115] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel, "Variational Lossy Autoencoder," in *5th International Conference on Learning Representations (ICLR)*, 2017, pp. 1–17.

[116] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules," *ACS Central Science*, vol. 4, no. 2, pp. 268–276, 2018.

[117] S. Eismann, S. Bartzsch, and S. Ermon, "Shape Optimization in Laminar Flow with a Label-guided Variational Autoencoder," *ArXiv*, 2017.

[118] N. Umetani and B. Bickel, "Learning Three-Dimensional Flow for Interactive Aerodynamic Design regression prediction for new shape," *ACM Transactions on Graphics*, vol. 37, no. 4, 2018.

[119] F. Murtagh, "Multilayer Perceptrons for Classification and Regression," *Neurocomputing*, vol. 2, no. 5, pp. 183–197, 1991.

[120] Y. Hirose, K. Yamashita, and S. Hijiya, "Back-propagation Algorithm which Varies the Number of Hidden Units," *Neural Networks*, vol. 4, no. 1, pp. 61–66, 1991.

[121] T. Rios, B. van Stein, T. Bäck, B. Sendhoff, and S. Menzel, "Multi-Task Shape Optimization Using a 3D Point Cloud Autoencoder as Unified Representation," *IEEE Transactions on Evolutionary Computation*, pp. 1–12, 2021.

[122] C. Smith, J. Doherty, and Y. Jin, "Convergence Based Prediction Surrogates for High-lift CFD Optimization Christopher," in *Royal Aeronautical Society – Applied Aerodynamics Conference 2014*, 2014, pp. 1–13.

[123] S. Saha, T. Rios, L. L. Minku, B. Stein, P. Wollstadt, X. Yao, T. Bäck, B. Sendhoff, and S. Menzel, "Exploiting Generative Models for Performance Predictions of 3D Car Designs," in *IEEE Symposium Series on Computational Intelligence, SSCI*, 2021, pp. 1–9.

[124] A. Kraskov, H. St, and P. Grassberger, "Estimating Mutual Information," *Physical Review*, vol. 69, no. 6, 2004.

[125] S. Saha, L. L. Minku, X. Yao, B. Sendhoff, and S. Menzel, "Exploiting Linear Interpolation of Variational Autoencoders for Satisfying Preferences in Evolutionary Design Optimization," in *IEEE Congress on Evolutionary Computation, CEC*, 2021.

[126] M. Segu, M. Grinvald, R. Siegwart, and F. Tombari, "3DSNet: Unsupervised Shape-to-Shape 3D Style Transfer," *arXiv preprint arXiv:2011.13388*, 2020. [Online]. Available: http://arxiv.org/abs/2011.13388

[127] M. Y. Liu, T. Breuel, and J. Kautz, "Unsupervised Image-to-image Translation Networks," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 701–709, 2017.

[128] R. Kuga, A. Kanezaki, M. Samejima, Y. Sugano, and Y. Matsushita, "Multi-task Learning Using Multi-modal Encoder-Decoder Networks with Shared Skip Connections," in *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW*, 2017, pp. 403–411.

[129] F. Locatello, S. Bauer, M. Lucie, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations," *36th International Conference on Machine Learning, ICML 2019*, vol. June, pp. 7247–7283, 2019.

[130] Q. Zhu and R. Zhang, "A Classification Supervised Auto-Encoder Based on Predefined Evenly-Distributed Class Centroids," *arXiv preprint arXiv:1902.00220*, 2019. [Online]. Available: http://arxiv.org/abs/1902.00220

[131] R. Das and S. Chaudhuri, "On the Separability of Classes with the Cross-Entropy Loss Function," *arXiv preprint arXiv:1909.06930*, 2019. [Online]. Available: http://arxiv.org/abs/1909.06930

[132] K. Yin, Z. Chen, H. Huang, D. Cohen-Or, and H. Zhang, "LOGAN: Unpaired Shape Transform in Latent Overcomplete Space," *ACM Transactions on Graphics*, vol. 38, no. 6, 2019.

[133] T. Rios, B. van Stein, S. Menzel, T. Bäck, B. Sendhoff, and P. Wollstadt, "Feature Visualization for 3D Point Cloud Autoencoders," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN*, 2020, pp. 1–9.

[134] T. Rios, P. Wollstadt, B. V. Stein, T. Bäck, Z. Xu, B. Sendhoff, and S. Menzel, "Scalability of Learning Tasks on 3D CAE Models Using Point Cloud Autoencoders," in *IEEE Symposium Series on Computational Intelligence, SSCI*, 2019, pp. 1367–1374.

[135] S. Saha, L. L. Minku, X. Yao, B. Sendhoff, and S. Menzel, "Split-AE : An Autoencoder-based Disentanglement Framework for 3D Shape-to-shape Feature Transfer," in *Internation Joint Conference on Neural Networks, IJCNN*, 2022.

[136] L. A. Stauffer and D. G. Ullman, "Fundamental Processes of Mechanical Designers Based on Empirical Data," *Journal of Engineering Design*, vol. 2, pp. 113–125, 1991. [Online]. Available: https://www.researchgate.net/publication/233065903

[137] D. G. Ullman, "Toward the ideal mechanical engineering design support system," *Research in Engineering Design - Theory, Applications, and Concurrent Engineering*, vol. 13, no. 2, pp. 55–64, 2002.

[138] N. Umetani, "Exploring generative 3D shapes using autoencoder networks," *SIGGRAPH Asia 2017 Technical Briefs*, vol. 4, pp. 1–4, 2017.

[139] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola, "Variance-based Sensitivity Analysis of Model Output. Design and Estimator for the Total Sensitivity Index," *Computer Physics Communications*, vol. 181, no. 2, pp. 259–270, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cpc.2009.09.018

[140] S. Saha, T. Rios, L. L. Minku, X. Yao, Z. Xu, B. Sendhoff, and S. Menzel, "Optimal Evolutionary Optimization Hyper-parameters to Mimic Human User Behavior," in *IEEE Symposium Series on Computational Intelligence, SSCI*, 2019, pp. 858–866.

[141] F. Chollet, "Keras," 2015. [Online]. Available: https://github.com/fchollet/keras

[142] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, D. C. May, and G. Brain, "TensorFlow : A System for Large-Scale Machine Learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[143] S. Saha, L. L. Minku, X. Yao, B. Sendhoff, and S. Menzel, "Exploiting 3D Variational Autoencoders for Interactive Vehicle Design," in *International Design Conference*, 2022, pp. 1747–1756.

[144] S. Saha, T. Rios, S. Menzel, B. Sendhoff, T. Bäck, X. Yao, Z. Xu, and P. Wollstadt, "Learning Time-series Data of Industrial Design Optimization using Recurrent Neural Networks," in *IEEE International Conference on Data Mining Workshops, ICDMW*, 2019, pp. 785–792.

[145] J. Hou, J. D. Deng, S. Cranefield, and X. Ding, "Cross-domain latent modulation for variational transfer learning," *Proceedings - 2021 IEEE Winter Conference on Applications of Computer Vision, WACV 2021*, pp. 3148–3157, 2021.

[146] J. Pan, T. Cui, T. Duy Le, X. Li, and J. Zhang, "Multi-Group Transfer Learning on Multiple Latent Spaces for Text Classification," *IEEE Access*, vol. 8, pp. 64 120–64 130, 2020.