

Universiteit
Leiden
The Netherlands

Deep generative models for engineering design

Fan, J.

Citation
Fan, J. (2026, March 24). *Deep generative models for engineering design*. Retrieved from https://hdl.handle.net/1887/4298630

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/4298630

Note: To cite this publication please use the final published version (if applicable).

Deep Generative Models for Engineering Design

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr. S. de Rijcke,
volgens besluit van het college voor promoties
te verdedigen op dinsdag 24 maart 2026
klokke 16:00 uur

door

Jiajie Fan

geboren te Wenzhou, China
in 1996

Promotor:

Prof.dr. T.H.W. Bäck

Co-promotor:

Dr. H. Wang

Promotiecommissie:

Prof.dr. M.M. Bonsangue

Prof.dr. V. Dunjko

Prof.dr. A. Plaat

Dr. E. Raponi

Dr. K. Li (University of Exeter, United Kingdom)

Prof.dr. B. Sendhoff (Technical University Darmstadt, Germany)

Copyright © 2026 Jiajie Fan All rights reserved.

This dissertation was made possible through the support of BMW AG which provided me with a full-time PhD position, thereby facilitating my research and studies as a guest PhD candidate at the Leiden Institute Of Advanced Computer Science, Leiden University.

"Nothing whets the intelligence more than a passionate suspicion, nothing develops all the faculties of an immature mind more than a trail running away into the dark."

Stefan Zweig

Contents

1 Introduction **1**

1.1 Industrial Design Process 5

1.2 Generative Engineering Design 7

1.3 Deep Generative Modeling 8

1.4 Deep Generative Models for Engineering 10

1.5 Publications 13

1.5.1 Other Work 14

2 Generation of Plausible Designs **15**

2.1 Introduction 16

2.2 Plausibility-oriented Diffusion Modeling 17

2.2.1 Background 17

2.2.2 Noise Range Relevant to Plausibility 20

2.2.3 Training and Generation Procedures 25

2.3 Evaluation and Results 26

2.3.1 Training Configurations 27

2.3.2 Results 27

2.3.3 Alignment Test 31

2.4 Controllable Generation and Design Editing 31

2.5 Conclusion 33

3 Evaluating the Plausibility with Deep Learning **39**

3.1 Introduction 40

3.2 Related Work on DGM Evaluation 42

3.3 Preliminaries 43

3.4 Method 44

3.5 Experiments . . . 45
3.5.1 Datasets . . . 45
3.5.2 Experimental Settings . . . 45
3.5.3 Sensitivity Test . . . 48
3.5.4 Consistency with Increasing Disturbances . . . 51
3.5.5 Model Ranking . . . 52
3.5.6 Grad-CAM Visualization . . . 54
3.5.7 Reconstruction with Denoising Autoencoder . . . 55
3.6 Conclusion . . . 56

4 Shape Generation with Learning-free Decomposition 59
4.1 Introduction . . . 60
4.2 Method . . . 61
4.2.1 Spectral Representation of Mesh . . . 61
4.2.2 Spectral Domain Diffusion . . . 66
4.2.3 Mesh Generation . . . 67
4.3 Experiments . . . 67
4.3.1 Experimental Dataset and Setup . . . 67
4.3.2 Evaluation Metrics . . . 67
4.3.3 Ablation Study . . . 69
4.3.4 Results . . . 73
4.4 Conclusion . . . 75

5 Learning Efficient Representations for 3D-Surfaces 77
5.1 Introduction . . . 77
5.2 Preliminaries . . . 80
5.3 Method . . . 81
5.3.1 Preprocess NURBS Parameters . . . 82
5.3.2 Learning NURBS Features with Autoencoder . . . 83
5.4 Experiments . . . 84
5.4.1 Datasets . . . 84
5.4.2 Surface reconstruction . . . 85
5.4.3 CAD (B-Rep) Generation . . . 87
5.4.4 Segmentation . . . 89
5.5 Implementation Details . . . 91
5.6 Conclusion . . . 92

6 Application in Industrial Development Processes 95
6.1 DGM-based Generative Engineering Design 95
6.2 Automotive A-pillar Design 97
6.2.1 Background and Motivation 97
6.2.2 Cross-Section Blueprints 99
6.2.3 Generation of Cross-Section Designs with GANs 100
6.2.4 A-pillar Blueprint Autocompletion 103
6.3 Vehicle Rim Design 104
6.3.1 Rim Design Data 105
6.3.2 Design Inspiration 105
6.4 Conclusion 107

7 Conclusion and Outlook 109
7.1 Research Question Revisited 109
7.2 Limitations and Future Work 111
7.3 The Future of DGMs in Generative Engineering Design. 113

Bibliography i

List of Abbreviations xxi

Summary xxiii

Samenvatting xxv

Acknowledgment xxvii

Curriculum Vitae xxix

Chapter 1

Introduction

Human design is a ubiquitous element of the modern world and plays a crucial role in the technologies involved in creating the products we use. However, product design processes can be extremely time-consuming due to the increasing complexity of the products, e.g., the development process for one car product takes five years. Time invested in a product may thus become critical in light of rapidly evolving market needs and preferences for more demanding and customized products. Automating and accelerating the design process would yield significant cost reductions and increase productivity within industries, an immeasurable gain for global productivity and prosperity. In Section 1.1, we introduce a traditional industrial design process in detail, as well as the challenges the industry faces therein.

Generative engineering design (GED) [3, 34, 115, 138] is proposed to overcome this issue. GED represents a new trend toward generative design originating from a more function-oriented design exploration, utilizing existing designs and algorithms for assisting engineers in developing complex structures. Most of the early-stage GED methods involve the manual definition of design rules based on human experience, i.e., explicit programming of design constraints and objectives. Developing such rule-based algorithms is very time-consuming and exceedingly costly. However, comparatively, the implicit learning of existing designs holds a lot of promise in generating powerful design exploration algorithms at a plurality of times lower costs than possible. Moreover, manually-programmed GED algorithms are essentially always project-dependent and involve a significant number of predefined parameters. However, strong algorithms intended for modern industries should be highly flexible in various design requirements and scenarios. Some more insight into the impact GED will have on modern industries

is provided in Section 1.2.

In recent years, machine learning has exhibited tremendous potential in surpassing the constraints of conventional algorithms across various real-world applications, including text understanding [174], image classification [81], object detection [136] and game playing [107]. Noteworthy achievements are evident in the field of deep generative models (DGMs, known as GenAI), like large language models [40, 20, 1], image synthesis [49, 72] and generation of other data formats [117, 108, 169, 130]. Since the emergence of ChatGPT [118], there has been a keen interest in deep generative models (and even large generative models) and their applications in industry. Further insights into DGMs are discussed in Section 1.3.

With the rapid development of DGM, utilizing it for generative engineering design becomes an intuitive decision due to its impressive ability to create new data. Regenwetter et al. have conducted a comprehensive survey [138], delving into this area and revealing the potential of DGMs in facilitating industrial design processes. In Section 1.4, we also demonstrate the most recent applications of DGMs for engineering designs. Despite the impressive results achieved in modern DGM researches, there is still a significant gap between current DGM technology and industrial applications. Below, we list the primary challenges faced by generative engineering using DGMs:

1. Poor plausibility of the designs generated. For the purpose of assisting the industrial design process, the generated designs must observe certain semantic constraints (referred to as plausibility, e.g., no floating material or missing parts). While the samples generated by current DGMs show outstanding visual quality (referred to as fidelity, e.g., no background noise and clear details), they often fail in generating plausible results. This directly results in these DGM-driven solutions being inefficient and unreliable.

2. Lack of metrics for measuring plausibility. Model evaluation is essential in the development of DGMs, as it allows for the ranking of models during the development phase and assesses the efficiency and reliability of the final model. When using DGM to enhance engineering design processes, evaluating the model's performance in terms of plausibility becomes critical. However, unlike metrics for fidelity or diversity, plausibility is an inherently subjective criterion, making it challenging to measure automatically. Current evaluation metrics often neglect the model's performance regarding plausibility. Relying solely on these existing metrics can lead to misguided decisions, such as selecting a DGM that ultimately fails to generate plausible designs.

3. Computation inefficiency for 3D objects. Advances in DGMs have made them proficient in generating regular data formats, such as images. However, 3D shapes (such as meshes and B-Reps) are not so straightforward for recent DGMs due to various reasons. These processes are sometimes energy and computation intensive, translating to prohibitive costs. Again, 3D object generation often relies on large-scale datasets; hence, performance obtained from modestly sized datasets hinders practical applications.

4. Inability to directly generate CAD-native representations of geometric objects. In general, DGMs used for generating engineering designs typically rely on cross-sections or meshes to represent source designs. Although widely used, such representations do not correspond to the most natural form of CAD data, which is primarily based on B-Rep solids described using parametric geometries. This discrepancy presents a significant limitation, as the conversion from images or meshes to B-Reps remains challenging at present. Learning DGMs directly from B-Reps is therefore a more logical and effective approach, as it allows for the generation of designs that are inherently compatible with CAD systems. By focusing on B-Reps representations, DGMs can produce more accurate and usable geometric models, facilitating smoother integration into engineering workflows.

5. Lack of historical successful application in industrial cases. The successes of DGMs are often showcased in controlled environments where the data is well-curated, characterized by a rich data pool and high quality. However, industrial applications frequently encounter a range of challenges that hinder the effective deployment of DGMs. These challenges include limited sample sizes, a significant amount of invalid or noisy data, and data formats that are not directly learnable by current models. As a result, the performance of state-of-the-art DGMs in these industrial contexts may not match their effectiveness in more general tasks. This gap highlights the need for further research and development to adapt DGMs for real-world industrial applications, ensuring they can handle the complexity and variability of engineering data. Addressing these issues is crucial for unlocking the full potential of DGMs in driving innovation and efficiency in industrial design processes.

While the above-mentioned limitations prevent engineering processes from taking advantages of the benefits of DGM, they indicate challenges with great industrial interest and research value. This thesis summarizes these challenges into corresponding research topics and solves them one by one by providing general solutions, which will

not only benefit engineering design processes, but also contribute to the DGM research community. The research questions are as follows:

RQ1: How to prioritize plausibility in generation with DGMs? Prioritizing plausibility ensures that the designs generated by DGMs are practical and can be realistically implemented in real-world scenarios. This is crucial because generating implausible designs can lead to wasted resources and inefficiencies. By focusing on plausibility, engineers and designers can rely on DGMs to produce viable and innovative solutions, accelerating the design process and reducing trial-and-error phases. Therefore, we improve the noise scheduling of the stat-of-the-art diffusion model (EDM [71]) and propose plausibility-oriented diffusion model (PoDM) [167], details can be fond in Chapter 2.

RQ2: How to automatically evaluate the plausibility of designs generated by DGMs? Metrics are an important assistant in evaluating and ranking models during the development of DGM, but they also reveal an unsolved challenge: due to the lack of ground truth in generation, it is difficult to automatically evaluate the synthetic samples while aligning with human judgments. After reviewing existing research on developing DGMs, we note that plausibility is overlooked not only in DGM-driven generation (as questioned in **RQ1**) but also in model evaluation. To address this, we propose a novel metric Fréchet Denoised Distance (FDD) [164], detailed in Chapter 3, which complements other metrics and aligns with human designers in evaluating result plausibility.

RQ3: How to use DGMs to generate high-dimension designs more efficiently? High-dimension designs often involve complex structures and multiple variables, so training DGMs is time-consuming and challenging. DGM developers tend to generate high-dimensional data via training an autoencoder, while the DGM is trained in the learned latent space. This can be computational costly and challenging in engineering process, where sample number is limited and structure of each sample is complicated and in high dimension. In Chapter 4, we propose a decomposition-based generation framework, SpoDify [168], for high-dimensional data, where our method enables a learning-free encoding and the decomposition outcomes can be used for DGM training.

RQ4: How to enable DGMs to directly synthesize CAD native representation? Directly synthesizing CAD native representations allows for seamless integration of generated designs into the existing workflow and tools used by professionals.

This capability will reduce the need for manual conversion and adjustments, thereby enhancing productivity and ensuring that the generated designs can be immediately utilized in practical applications. We delve into this in Chapter 5 and propose NeuroNURBS [166] that is able to learn and generate directly B-Rep (the CAD native representations) and NURBS (the native representation of parametric geometries in B-Reps).

RQ5: How can DGMs be beneficial for real-world industrial design applications? Understanding the practical benefits of DGMs in real-world applications is essential for their adoption and integration into industry practices. This question addresses the tangible impact of DGMs on the field of industrial design. Demonstrating the benefits of DGMs—such as improved design efficiency, cost reduction, and enhanced innovation—will encourage industry adoption, leading to widespread improvements in design practices and outcomes. In Chapter 6, we take car rims and A-pillars as design targets and demonstrate the efficiency of DGMs in industrial cases.

Together all the above research questions are aiming to solve one primary research topic, which is:

How to enable DGMs to synthesize engineering designs?

1.1 Industrial Design Process

In modern industrial design processes, a design consists of a digital or virtual 3D object that defines the product of interest. For instance, a car or a subset of a car (such as a breaking system) is assembled from many connected components. The field of computer-aided design (CAD) encompasses all topics related to such virtual models. By nature, this virtual model or "CAD model" contains geometrical and topological information about the object. To some extent, material information can also be defined in the CAD model.

In industry, engineers conduct design, simulation, and manufacturing using CAD software such as `CATIA` of `Dassault Systems`, where the standard is to represent a solid model with boundary representation (B-Rep). In a B-Rep, the solid boundaries are defined using a set of surfaces [177, 39], which are, by default, parameterized by non-uniform rational B-splines (NURBS) [128]. A NURBS surface leverages weights assigned to the control points and non-uniform knot vectors to represent more complex shapes compared to Bézier or B-splines. It is worth noting that NURBS is often used as an intermediate data format when exchanging data between CAD software. NURBS

allows to define shapes with perfect precision due to their continuity.

Often, CAD software allows the generation of alternative representations of the CAD model, such as images (cross-section or snapshot) or meshes, which can then be easily used outside of the CAD framework. However, converting these design representations back into CAD models can be challenging. In sum, industrial CAD models use relatively complex data representations with the following characteristics:

- A mixture of parametric data (material properties, geometric parameters such as thickness or radius, 3D geometry described using NURBS) and non-parametric data (3D geometry described using a point cloud or image).
- High-dimensionality. The virtual car model is full-scale and consists of numerous (up to hundreds) connected parts.
- Due to the lack of fixed design conventions (different sizes, various parameters and constituent objects), there is a lack of unique definitions when considering variants of the same design.

Within the framework set by predefined requirements, the designer can modify the geometry by adjusting geometric parameters or reshaping certain components. The designer may also modify the topology of the model by adding holes to the design. These design modifications are triggered by the designer's creativity and feedback from the engineers, who collaborate closely during the design process.

The product design process [60] begins with creating a preliminary CAD model, which evolves as designers incorporate creative ideas and perform functional evaluations. As design progresses, details are added and changes are made based on feedback from engineers to meet predefined requirements. These requirements, which range from quality and safety to manufacturability, are essential to ensuring compliance with legal and industry standards [51]. Traditionally, functional performance was gauged by experts and sometimes involved physical experiments. However, digital methods like numerical simulations have become more prevalent, offering cost-effective and faster evaluations in a partially automated way. This is the essence of computer-aided engineering (CAE), which employs various numerical techniques based on the physical aspect being assessed. Finite-element methods (FEM) are used for structural problems, while finite-volume simulations deal with fluid mechanics. For simulations, CAD models must be transformed into CAE-compatible models, involving meshing (discretizing the continuous geometry) to handle complex geometries—a task requiring expertise. Although once set up, a simulation can run autonomously, substantial

expert input is necessary for initial data preparation. Additionally, simulations demand expensive software licenses, considerable computing power, and time, sometimes taking minutes to days depending on model complexity. As such, simulation-based design evaluations remain resource-intensive. In early design stages, experts often rely on their judgment for functional assessments to conserve time and resources, introducing certain bias and communication challenges. To mitigate costs and inefficiencies, early identification of underperforming design variants is key, so that numerical simulations are reserved for promising designs only. This would enhance objectivity in functional assessments and reduce the experts' reliance on them when simulations are overly costly.

1.2 Generative Engineering Design

In the traditional engineering design process, designs need to be modified repeatedly to meet predefined requirements. In this design cycle, engineers are tasked with manually adjusting the design based on their expertise due to the lack of intelligent tools, making it a trial-error process. This involves constructing and simulating designs until achieving an optimal outcome, often without knowing how to improve the design exactly to reach the desired performance without adversely affecting other aspects. As a result, the engineering design process tends to be very time-consuming and expensive, indirectly contributing to the slow development of modern products and the high cost of new designs. Consequently, this approach is often very time-consuming and costly, which contributes to the slow pace of modern product development and the high expenses associated with new designs.

Recently, advanced algorithms either employ explicit programming or implicit learning to explore design possibilities that meet engineer predefined requirements [138]. This concept has been introduced as the term "generative design" [77] and yielded a tons of form-finding tools, which shows a great potential in complementing the previous "form-making" engineering process [153]. Generative design is proposed to support designers/engineers in developing complex structures, in which algorithms explore possible design alternatives based on input constrains. In practice, designers or engineers pre-determine several technical constraints and then develop algorithms to generate or optimize the design to meet these requirements. Generative design here is further referred as generative engineering design (GED) and is a subsequent step to parametric modeling in product development [153]. More specifically, GED helps develop designs that better or best meet the predefined requirements, whereas the current product

development relies heavily on the construct-evaluate cycle of human designers [34].

Early computational tools that enable GED tend to be manually-defined pipeline. For instance, parametric modeling of a structural component allows designers to focus on shape optimization with limited parameters. These tools help with developing novel solutions that bypass fixation of human designers, and automate some routine design processes to improve efficiency [25]. However, these rule-based frameworks rely heavily on human design expertise and require significant cost in terms of financial investment and time in the development process. Later, with the rapid development of artificial intelligence (AI), the advantages of implicit leaning on information and knowledge encoded in the vast expanse of existing designs has been seen. This part has been introduced in detail in Section 1.4.

1.3 Deep Generative Modeling

Machine learning is a field of artificial intelligence that enables computers to learn and improve from experience. It involves the development of algorithms and statistical models that allow systems to perform specific tasks effectively without using rule-based programming. Prior to the rise of deep learning, a variety of applications have introduced machine learning techniques, including linear regression [54], decision trees [132], support vector machines (SVMs) [37], etc. These traditional machine learning techniques were widely used before the advent of deep learning, which has since become the dominant approach in many areas of artificial intelligence and data analysis.

Deep learning is capable of learning meaningful distributions from extensive quantities of data through a deep neural network with millions of trainable parameters. Starting with the compelling results achieved using autoencoder [59] to reduce the dimensionality of large-scale data with neural networks, the era of deep learning begins. With the introduction of convolutional layers, deep convolutional neural networks (DCNNs) can better capture patterns from the locality of pixel dependencies in images compared to fully-connected neural networks and hereby excel in the vision understanding tasks, e.g., the ImageNet classification task [38, 55, 81]. Despite the powerful learning capacity of deep neural networks, it is not trivial to train such deep architectures with a surge of parameters [47]. More precisely, training deep neural network suffers from long training time, early saturation, overfitting. To improve the training stability, a set of approaches are proposed, e.g., rectified linear units (ReLU) [81], dropout [160], residual block [55], batch normalization [65], etc.

Meanwhile, a new subfield that is making astonishing breakthroughs is the deep generative models (DGMs) — deep neural networks (DNNs) that can learn the complex probability distribution of large datasets and generate new samples. Unlike previous generative models, deep Boltzmann machines [142], generative stochastic networks [12] and variational autoencoders (VAE) [75], generative adversarial networks (GANs) [49, 133] have first yielded convincing results in image generation. Within a short period of time after that, a series of derivatives have been proposed to further improve the performance of the vanilla GAN, such as Wasserstein GAN with gradient penalty (WGAN-GP) [6] to stabilize the training, progressively growing GAN (ProGAN) [70] for producing images of large resolutions, conditional GAN (CGAN) [104] to enable conditional generation and StyleGAN [72] that allows for intuitive, scale-specific control of the synthesis. Note that SyleGAN is the first work observes that the level of the affected feature depends on the scale of the noise input. For example, tiny noises can edit the curves of the hair, while coarse noises can change gender. Our work in Chapter 2 is greatly inspired by their observation.

Diffusion models [155] present a novel idea for capturing the data distribution and generation. Diffusion models did not attract much attention until the convincing implementation of the denoising diffusion probabilistic model (DDPM) [61], which leverages tremendous sampling time to generate images with quality comparable to GANs. A further developed diffusion model, denoising diffusion implicit model (DDIM) [156], improves generation speed by trading off image quality. Meanwhile, in the domain of score-based generation, Song et al. [159] propose to use a stochastic differential equation (SDE) for the forward process and a corresponding reverse-time SDE for sampling, which allows continuous diffusion processes. SDE derives a deterministic sampling process based on a corresponding ordinary differential equation (ODE), that enables identifiable encoding-decoding and more importantly flexible data manipulation via latent space. Then, Karras et al. [71] clean the design space of diffusion-based generative models and propose a novel framework, denoted as EDM. EDM achieves a new state-of-the-art performance on the generation of CIFAR-10 [80] and ImageNet-64 [38]. Diffusion-based generative models have been introduced in controlling generation and data manipulation, e.g., interpolation via latent space [61, 159, 156, 41], free-form inpainting [159, 97] and point-based dragging [150]. These image editing methods tend to be applied on natural images, e.g., CelebA [94], LSUN bedroom images [192] and ImageNet [38].

A novel backbone, Transformer (consists solely of attention layers), has achieved compelling results in text translation task [174], which changes the dominant situation

of convolutional and recurrent neural networks. Inspired by this, researchers start introducing attention layer in DGMs for computer vision tasks, e.g., self-attention GAN [186], which demonstrates superior performance than its contrastive version (i.e., GANs without attention layer). As an explanation to this, self-attention mechanism is able to capture long-range dependencies across image regions, whereas convolutional layers are restricted due to their limited receptive fields. Inspired by this, we use Self-Attention Adversarial Latent Autoencoder (SA-ALAE) [165] developed by ourself to generate cross-section images of car A-pillars, more details can be found in Chapter 6. Soon after this, introducing attention layers into the DGMs becomes a new standard. Most recently, Vision Transformer (ViT) [78] that applies a standard Transformer model directly to images, outperforms CNNs in image recognition tasks and exceeds the state of the art on many image classification datasets. This breaks the dominance of convolutional architectures and also the boundaries between the field of neuro-linguistic processing (NLP) and computer vision. While standard diffusion models utilize a U-Net architecture [61] to perform the denoising for each step, right after ViT showing its excellent performance in computer vision, transformer-based diffusion models (DiTs) [125] are invented.

While diffusion models have efficiently addressed the unstable training issues associated with GANs, they face a significant drawback: slow sampling speed. This limitation becomes particularly pronounced when working with high-resolution images, as both the model size and evaluation times increase exponentially. The latent diffusion model (LDM) introduced by Rombach et al. [139] mitigates this challenge by conducting diffusion and denoising processes in a lower-dimensional latent space, which is encoded from the source data using a pretrained autoencoder. Additionally, LDMs leverage cross-attention layers, making them versatile generators capable of handling various conditioning inputs, including text, semantic maps (masks for inpainting), and images. Another approach to improve the sampling speed of diffusion models involves distilling a pretrained diffusion model into a new model that requires significantly fewer steps for sampling, all while maintaining sample quality. Recent advancements in this area include progressive distillation [144], consistency models [157], adversarial diffusion distillation [145], distribution matching distillation [191].

1.4 Deep Generative Models for Engineering

Deep generative models (DGMs) have been successfully employed to synthesize general images, e.g., animals, human faces, and landscapes. This promising advancement leads

to the idea of utilizing DGMs to generate novel structural designs, thereby facilitating industrial engineering processes. However, industrial design data, e.g., blueprints or engineering drawings, is fundamentally different from the images of natural scenes. They contain rich structural patterns and long-range dependencies, which are challenging for convolution-based DGMs to generate. Recently, some models (PaDGAN [27] and BézierGAN [28]) are able to generate UIUC Airfoil shapes [170], however, when it comes to more complicated design like BIKED bicycle designs [137], DGMs fail to synthesize feasible results [137]. In order to tackle this issue, Fan et al. [165] propose a novel model self-attention adversarial latent autoencoder (SA-ALAE), which allows generating feasible design images of complex engineering parts, but still suffers from unstable training due to the implementation of adversarial training.

Moreover, learning from 2D geometric designs is not sufficient for the rapidly evolving industry. Huge progress has been made in using modern neural networks to tackle 3D learning tasks. It starts with representing 3D solids in voxels for its compatibility with convolutional mechanism, hereby yielding a set of early 3D generation models, e.g., 3D-GAN [181] and 3D U-Net [33]. To avoid voxel's unbearable memory and computation consumption, researchers have turned their attention to point clouds [129]: the generation of point clouds can be done by combining an point-cloud autoencoder and a GAN that is trained in the latent space of the autoencoder [2]; Luo et al. [98] enable the generation of high-quality 3D point clouds using diffusion models. In fact, point-based DGMs are extremely inefficient at modeling sparse data, and the lack of neighborhood information leads to poor model performance. Point-voxel CNN (PVCNN) [93] breaks the isolation wall between point cloud and voxel, the proposal of point-voxel diffusion (PVD) can significantly improve the fidelity of generated shapes. Most recently, LION [194] combines various advancements, such as PVCNN and latent diffusion model (LDM) [139], and achieves the novel state-of-the-art quality of generation results.

However, the usability of point clouds in industrial applications is limited. More precisely, point clouds often fail to accurately represent the complex shapes and geometries encountered in industrial settings. The rendered meshes derived from these point clouds can not be used for simulations, limiting their practical utility. In contrast, meshes [96, 154] are much more commonly used form in industry, as the native representation of many CAE software and Finite Element tools. Nevertheless, due to the non-uniform representation of meshes (which consist of nodes and edges), it's non-trivial to leverage CNNs. To tackle this, MeshCNN [53] has been designed to enable the use of CNN on irregular mesh data forms, and has achieved initial success in direct

mesh learning. Afterwards, substantial progress has been made in directly learning on meshes, e.g., convolutional mesh autoencoder (CoMA) [135], MeshGAN [30] and MeshingNet [199]. Most recently, researchers [32, 151, 64] in this field have greatly facilitated the solid generation task with the implicit representation signed distance field (SDF) [162].

Despite the considerable progress made in learning from various solid representations, it is more natural and advantageous to learn directly from Boundary Representation (B-Rep) [177]: B-Rep is more precise, easier to manipulate, and takes up less memory [56]. However, directly learning B-Rep entities remains challenging because B-Rep consists of parametric surfaces, parametric curves and vertices. These entities need to be trimmed and sowed to become solid models. Also, B-Reps store adjacency information for neighbouring edges and vertices, allowing the structure to provide a complete description of the final entity shape [187]. DeepCAD [182] is able to generate CAD source data, i.e., STEP files, by learning each CAD model as a sequence of modeling operations such as sketching and extruding. Following this idea, a set of works have been done, e.g., Datasets (DeepCAD [182] and Fusion 360 gallery [180]), CAD generative models [85, 188] and STEP-based learning models [102]. However, these works mainly focus on CAD models, the modeling operations of which are limited to sketching and stretching, and they are difficult to scale [66].

To directly consume the geometric and topological information from B-Rep data, recent attempts convert the B-Rep into a graph, then pass through a graph neural network (GNN), hereby performing learning tasks such as face segmentation or solid classification [5, 67, 36]. Prediction accuracy has been significantly improved by increasing the information added to the graph, i.e., from using face normal and distance to origin as node feature in CADNet [36] to additionally introducing UV-grids, trimming mask and curve geometries into the graph in UV-Net [67]. While converting B-Rep to graphs performs well in shape classification and segmentation tasks, in the field of solid generation, B-Rep is often represented as a predefined hierarchical structure and generated by autoregressive prediction of B-Rep entities. This approach has given rise to some convincing methods, such as SolidGen [66] and BrepGen [187].

It is worth noting that current solid generation research is avoiding the generation of parametric surfaces with their natural representations — non-uniform rational B-splines (NURBS) [128]. However, as a structured form of a point cloud, UV-grids suffer from inefficient memory consumption, inaccurate surface representation, and cubic growth of computing costs with the complexity of the surface.

1.5 Publications

Most of the work in this dissertation has been previously published. The content of this dissertation includes the following papers:

[167] **Jiajie Fan**, Laure Vuaille, Thomas Bäck, and Hao Wang. On the noise scheduling for generating plausible designs with diffusion models. *CoRR*, abs/2411.10848, 2023.

This work addresses the issue of poor plausibility of results generated by diffusion models by proposing a novel noise scheduling method, which is demonstrated in Chapter 2. Upon completion of this dissertation, this paper has been submitted to the *Computer Graphics Forum* and is currently under review.

[164] **Jiajie Fan**, Amal Trigui, Thomas Bäck, and Hao Wang. Enhancing plausibility evaluation for generated designs with denoising autoencoder. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 88–105, Cham, 2025. Springer Nature Switzerland.

To address the challenge of automatically evaluating result plausibility for assessing DGMs, we designed a new metric FDD in this work. See detailed description in Chapter 3.

[168] **Jiajie Fan**, Amal Trigui, Andrea Bonfanti, Felix Dietrich, Thomas Bäck, and Hao Wang. A mesh is worth 512 numbers: Spectral-domain diffusion modeling for high-dimension shape generation. *arXiv preprint arXiv/2503.06485*, 2025.

This work enables a training-free encoding method for high-dimensional shape data, where the encoded features are proven to be learnable with DGMs. We introduce this work in Chapter 4. Upon completion of this dissertation, this paper has been accepted for *2026 IEEE Conference on Artificial Intelligence (CAI)*. Following our presentation of the paper at the conference, it will be included in the IEEE Xplore.

[166] **Jiajie Fan**, Babak Gholami, Thomas Bäck, and Hao Wang. NeuroNURBS: Learning efficient surface representations for 3D solids. *CoRR*, abs/2411.10848, 2024.

In this work, we addressed the challenge of learning directly on parametric geometries (i.e., NURBS surfaces) and successfully applied it to various downstream tasks, such as solid generation and solid segmentation. We demonstrate

this work in Chapter 5. This work spawned numerous subsequent peer-reviewed papers, making significant contributions to the field of AI-assisted computer-aided design, such as: from NURBS to neural network [147], TPDLF [189], GEOM-GNN [17], NURBGen [172] and BrepARG [89]. Upon completion of this dissertation, this paper has been submitted to the *ACM Transactions on Graphics* and is currently under review.

[165] **Jiajie Fan**, Laure Vuaille, Thomas Bäck, and Hao Wang. Adversarial latent autoencoder with self-attention for structural image synthesis. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 119–124, 2024.

As an early work in this thesis, we proposed SA-ALAE in this paper and evaluated it on real-world engineering designs — 2D blueprints of car A-pillar design. More details can be found in Section 6.2.3.

1.5.1 Other Work

Besides the papers that form the content of this thesis, I have also been involved in the conceptualization, writing, and review process of the following paper:

[110] Phillip Mueller, Talip Uenlue, Sebastian Schmidt, Marcel Kollovieh, **Jiajie Fan**, Stephan Günnemann, and Lars Mikelsons. Geodiffusion: A training-free framework for accurate 3d geometric conditioning in image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6374–6384, October 2025. In this work, we addressed the challenge of conditioning generative models on 3D parametric geometries (via our "training-free" framework) and demonstrated its efficacy across accurate geometry alignment and downstream editing tasks.

Chapter 2

Generation of Plausible Designs

In generative engineering design, the task of DGM is to generate structural meaningful designs, which refers to the plausibility criterion in the field of DGM. This chapter studies the diffusion model (the state-of-the-art DGM backbone) for structural designs and proposes a solution that prioritizes the plausibility when using DGMs in generating designs, aiming to addressing the research question 1: *How to prioritize plausibility in generation with DGMs?* The content of this chapter has been published in the paper [167].



(a) Implausible bicycles generated by EDM [71]



(b) Bicycles generated by our PoDM [167]

Figure 2.1: Generated bicycle designs. Our work aims to minimize the proportion of implausible designs generated by focusing on a certain range of noise levels.

2.1 Introduction

Diffusion-based generative models have been reported to surpass GANs [49, 133, 72, 165] in various image synthesis tasks [61, 41, 71]. Despite the success of diffusion-based models, several issues exist to address when generating design structures. For instance, denoising diffusion probabilistic models (DDPM) [61] can generate structure images with high visual quality; however, it suffers from a slow generation speed due to the usage of an excessive number of denoising steps [79, 156, 71]. As a remedy, the denoising diffusion implicit model (DDIM) [156] greatly reduces the denoising steps, which, however, compromises the visual quality. Recently, the influential work "Elucidating the Design Space of Diffusion-Based Generative Models" (which introduced the method termed EDM) [71] incorporated a probability distribution to sample noise levels during the denoising process. This approach enhances generation speed while maintaining satisfactory visual quality, particularly excelling in rendering image details such as human hair curls and skin pores. However, when assessing the plausibility of the generated design images, we observe that the EDM often produces structurally implausible images, see Figure 2.1a.

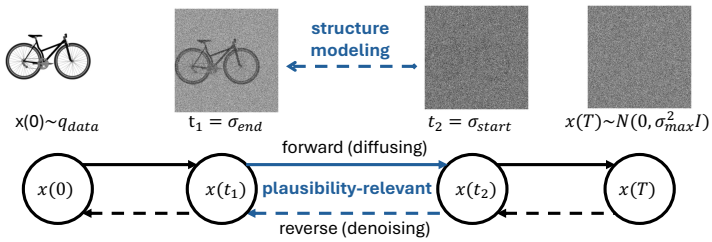


Figure 2.2: Forward and reverse processes of diffusion models in directed graphs, with a highlight on the plausibility-relevant range of noise levels, i.e., $[\sigma_{\text{end}}, \sigma_{\text{start}}]$.

For the denoising process, recent research has explored the impact of the noise schedule on the properties of generated images [114, 158, 159, 71]. For example, while lower noise levels can enhance the quality of image details [71], higher noise levels can affect the diversity of generated results [158]. In this chapter, we discover that in using diffusion models, there exists a plausibility-relevant range of noise levels that predominantly affect the plausibility of the images (see Figure 2.4). More importantly, this range can be determined by the evolution of pixel-value distributions in the forward diffusion process (see Section 2.2.2). We visualize the plausibility-relevant range and its relevance to structural generation in Figure 2.2. To determine the plausibility-

relevant range, we simulate the forward diffusion process on real structural designs and trace the distribution of pixel values as the noise level increases. We observe that the disappearance of the structural signal has a clear corresponding phase in the development of pixel-value distributions.

Taking this observation, we modify the training and generation procedures of EDM to prioritize sampling the noise levels in the plausibility-relevant range (see Figure 2.3a for an illustration), resulting in a new method, plausibility-oriented diffusion model (PoDM). We experimentally test PoDM on three datasets, the BIKED dataset [137] (see some generated examples in Figure 2.1b), Seeing3DChairs [7] and Shoes [193], in terms of the following metrics: (1) Fréchet inception distance (FID) [58] for visual quality. FID measures how close the generated data distribution is to the real data distribution by comparing their feature representations extracted from a pretrained network: lower FID scores indicate more realistic and higher-quality generations. FID has served as the golden standard metric in generative modeling since it exists. We defer the detailed introduction of FID in Chapter 3; (2) plausible design rate (PDR), the proportion of plausible designs for 1 000 evaluated images (see Section 2.3.2 for details); and (3) frames per second (FPS) for generation speed. On the BIKED data, our PoDM outperforms EDM on PDR: 93.5% (PoDM) vs. 83.4% (EDM) and on FID: 4.87 (PoDM) vs. 7.84 (EDM), while achieves comparable FPS with EDM. Compared to DDPM, PoDM has a comparable PDR value (PDR: 94%, FID: 11.77) but is ca. 15× faster in terms of FPS.

Lastly, we further test the performance of PoDM in incorporating modern image-editing methods, e.g., inpainting, interpolation via latent space and point-based dragging, and hereby manipulating structure.

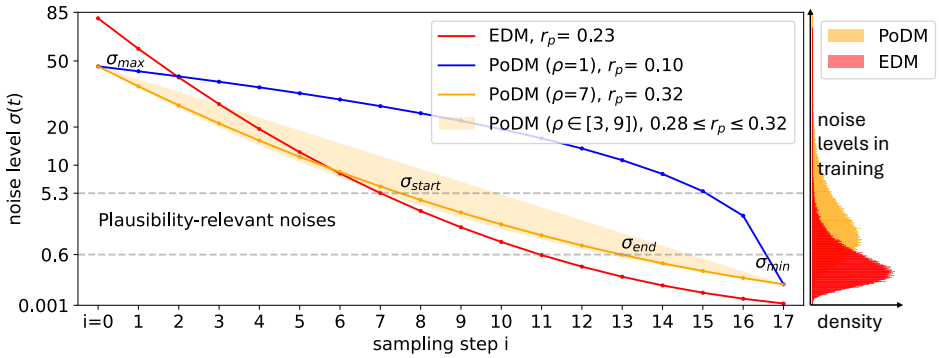
2.2 Plausibility-oriented Diffusion Modeling

2.2.1 Background

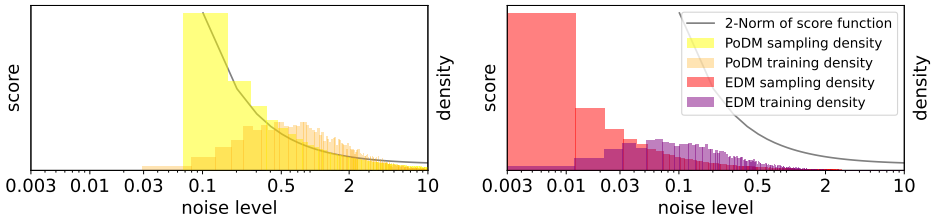
We built up our contribution based on the stochastic differential equation (SDE) model of the diffusion process [159, 71]. Given a data point $\boldsymbol{x} \in \mathbb{R}^d$, we corrupt it with the following forward Itô SDE [116]:

$$d\boldsymbol{x} = f(\boldsymbol{x}, t)dt + g(t)dB_t, \tag{2.1}$$

where $f\colon \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ is the drift vector, $g\colon [0, T] \to \mathbb{R}$ is the dispersion coefficient, and $B_t \in \mathbb{R}^d$ is the standard Brownian motion. Notably, f and g, are



(a) Plausibility-relevant noise range and noise schedules



(b) Magnitude of the score function and density of noise schedules

Figure 2.3: (a) We showcase the plausibility-relevant noise range $[\sigma_{\text{end}}, \sigma_{\text{start}}]$ (*dashed interval*) computed on the BIKED dataset with the techniques proposed in Section 2.2.2. For the denoising process, our PoDM method takes an exponentially decaying schedule (*blue and orange curves*) with a parameter ρ controlling the decay rate. PoDM method determines the minimal/maximal noise levels of the schedule (σ_{min} and σ_{max}, respectively) based on $[\sigma_{\text{end}}, \sigma_{\text{start}}]$ (Equation (2.8)). As for the noise levels sampled in the training process, our PoDM method uses a log-normal density concentrating on $[\sigma_{\text{end}}, \sigma_{\text{start}}]$ (*the orange histogram shown vertically*), in contrast to the fixed distribution used in EDM (*the red histogram*). (b) For PoDM and EDM, we depict the magnitude of the score function (Equation (2.2)) at different noise levels, which is compared to the density of the noise levels in the training and denoising steps. The noise density in the denoising process of PoDM matches closely with the score function.

pre-determined by the user and have no trainable parameters. The corresponding reverse-time/backward SDE is [4]:

$$d\boldsymbol{x} = [f(\boldsymbol{x}, \tau) - g(\tau)^2 \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, \tau)]d\tau + g(\tau)d\mathbf{B}_\tau, \tag{2.2}$$

where τ goes from T to 0, $p(\boldsymbol{x}, \tau)$ is the probability density of $\boldsymbol{x}$ at τ in the forward process, and $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, \tau)$ is known as the score function. The flow of probability mass in Equation (2.2) can be equivalently described by an ordinary differential equation (ODE) [100, 159, 71]:

$$\frac{d\boldsymbol{x}}{d\tau} = f(\boldsymbol{x}, \tau) - g(\tau)^2 \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, \tau). \tag{2.3}$$

We follow the choice of the drift and dispersion terms in [71] (a.k.a. EDM):

$$f(\boldsymbol{x}, \tau) = 0, \quad g(\tau) = \sqrt{2\sigma(\tau)}, \quad \sigma(\tau) = \tau,$$

and the score function is approximated by $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, \tau) = (D_\theta(\boldsymbol{x}; \sigma(\tau)) - \boldsymbol{x})/\sigma(\tau)^2$, where D_θ is a neural network trained on samples drawn from the forward SDE (see [71] for details on the loss function). Due to the above choice, σ and τ are interchangeable henceforth. To solve/sample from Equation (2.3), an N time-step discretization is used with the following noise schedule: $\sigma_N = 0, \forall i \in [0..N-1]$:

$$\sigma_i = \left(\sigma_{\max}^{\frac{1}{\rho}} + \frac{i}{N-1}(\sigma_{\min}^{\frac{1}{\rho}} - \sigma_{\max}^{\frac{1}{\rho}})\right)^\rho, \tag{2.4}$$

where $\sigma_0 = \sigma_{\max}$ and $\sigma_{N-1} = \sigma_{\min}$. EMD recommends the setting: $\sigma_{\min} = 0.002, \sigma_{\max} = 80, \rho = 7$. The exponent ρ affects how much the steps near $\sigma_{\min}$ are shortened at the cost of longer step length near $\sigma_{\max}$. In the stochastic sampling procedure, we denote by $\boldsymbol{x}_i$ the data point obtained at σ_i. We first increase the noise level slightly and perturb $\boldsymbol{x}_i$:

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + \sqrt{\hat{\sigma}_i^2 - \sigma_i^2}\mathcal{N}(\mathbf{0}, S_{\text{noise}}^2\mathbf{I}), \tag{2.5}$$

$$\hat{\sigma}_i = \sigma_i \left(1 + \mathbb{1}_{[S_{\min},\, S_{\max}]}(\sigma_i) \min\left(S_{\text{churn}}/N, \sqrt{2} - 1\right)\right), \tag{2.6}$$

where S_{churn} controls the degree of randomness in sampling: $S_{\text{churn}} = 0$ realizes deterministic generation. Afterwards, we apply the reverse-time ODE (Equation (2.3)) with $\boldsymbol{x}_i'$ from $\hat{\sigma}_i$ to σ_{i+1}. The default settings of stochastic sampling are: $S_{\text{churn}} = 40$, $S_{\min} = 0.05$, $S_{\min} = 50$, $S_{\text{noise}} = 1.003$. The training data of $D_\theta(\boldsymbol{x}; \sigma(t))$ are

Table 2.1: Grid search results of the parameter ρ of the noise schedule on the BIKED data w.r.t. three performance metrics. The *column* r_p measures the proportion of noise levels falling into the plausibility-relevant range, which is strongly correlated with the performance metrics.

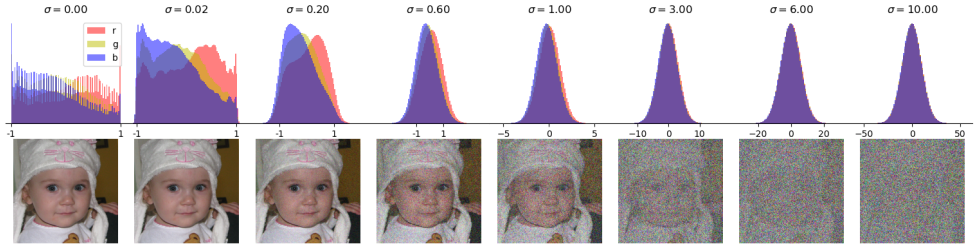
ρ	r_p	FID↓	DPS↑	PDR↑
1	0.10	12.67	4.70	82.8%
3	0.28	5.64	4.81	88.6%
5	0.31	5.25	4.88	89.6%
7	0.32	**4.87**	**4.90**	**93.5%**
9	0.32	5.18	4.87	91.5%

sampled from Equation (2.1) with a log-normal distribution: $\ln(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$. In [71], the following empirical setting is suggested: $P_{\text{mean}} = -1.2, P_{\text{std}} = 1.2$.

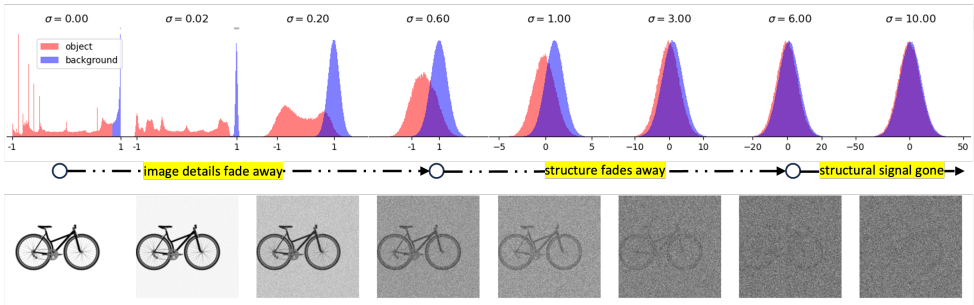
2.2.2 Noise Range Relevant to Plausibility

We conjecture that in the forward/backward diffusion process, there exist a range of noises that plays a crucial role in object construction. We validate it by running the forward process with fine-grained noise levels: starting with a source image $\boldsymbol{x}(0)$ (the pixel values are standardized to $[-1, 1]$ before adding the Gaussian noise), we keep adding small Gaussian noises thereto until its pixel-value distribution becomes indistinguishable from a Gaussian: $\boldsymbol{x}(t) = \boldsymbol{x}(0) + 0.1t \times \mathcal{N}(\mathbf{0}, \mathbf{I})$. In Figure 2.4b, we show the pixel-value distribution at intermediate time steps computed from 100 images sampled from BIKED [137]. The BIKED images consist of a single bicycle object and a monotone background. We depict the pixel-value distribution separately for the object and the background (the red and blue histograms, respectively). We observe three phases: (1) from the beginning to the first time when the pixel-value distribution of the bicycle converges to a Gaussian, i.e., $\sigma \in [0, 0.6]$. The pixel-value distribution of the bicycle is substantially different from that of the background in this phase; (2) the bicycle structure starts to fade away while the pixel-value distribution thereof overlaps more with the background, i.e., for $\sigma \in [0.6, 6.0]$; (3) the bicycle structure almost disappears for $\sigma > 6$. Empirically, we assume that the noise range ($\sigma \in [0.6, 6]$ in Figure 2.4) in which the bicycle structure fades away determines the plausibility of the generated structural design. We denote this noise range as $[\sigma_{\text{end}}, \sigma_{\text{start}}]$ and propose two techniques to determine this interval.

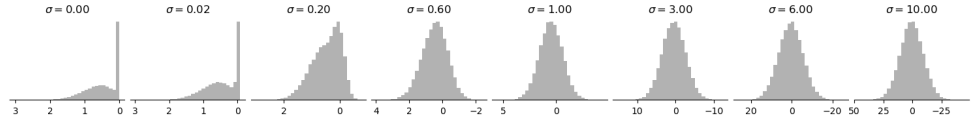
Technique 1 *Choose σ_{end} to be the largest noise level at which the object pixel values are not normally distributed according to the Shapiro-Wilk test with a significance level*



(a) FFHQ [72] (resolution: 64×64)

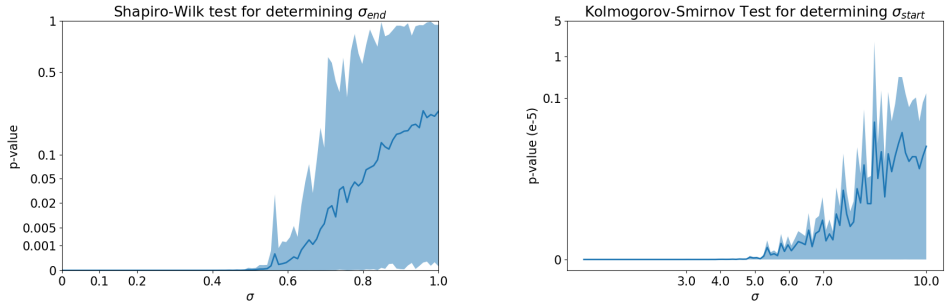


(b) BIKED [137] (resolution: 256×256)



(c) Distribution of latent features extracted from the BIKED data [137] with a convolutional autoencoder.

Figure 2.4: Evolution of the pixel-value distribution in perturbation experiments for (a) FFHQ, a real-world data set, and (b) the BIKED data, a set of design structures. In (c), we show the evolution of the latent-value distribution after encoding the BIKED data into a 64-dimensional space with a convolutional autoencoder.



(a) Shapiro-Wilk test to determine σ_{end} **(b)** KS test to determine σ_{start}

Figure 2.5: On the BIKED data, (a) we illustrate **Technique 1** by plotting the p-value of the Shapiro-Wilk test computed from 100 randomly picked images, which are perturbed by different noise levels; (b) for **Technique 2**, we show the Kolmogorov-Smirnov (KS) test between the bicycle's and background's pixel-value distribution as a function of the noise levels. In both plots, the variability is depicted as the *shaded region*.

of 0.01.

In a sampling process, σ_{end} is the noise level at which the generation is structurally finalized and object pixel values remain normally distributed. Denoising with noise levels of $\sigma \leqslant \sigma_{\text{end}}$ performs a refinement, during which the backward process approximates the object pixel values from a normal distribution to a local data distribution. To estimate σ_{end}, we propose to use the Shapiro-Wilk test [149] to track the distribution of the object's pixel values during the perturbation test. As displayed in Figure 2.5a, the measured p-value increases with the noise level. In practice, specific dataset might exist, where the object pixel-value distribution is already Gaussian distributed, here we set a minimum limitation of σ_{end} to be 0.08 (converting the default parameter values of EDM to our parameter system, we obtain the value $\sigma_{\text{end}} = 0.08$).

Technique 2: *Choose σ_{start} to be the noise level at which pixel-value distributions of object and background are sufficiently close, with the p-value of a Kolmogorov-Smirnov test begins to diverge.*

In the synthesis of structural design images, σ_{start} is the noise level at which the structural formation begins, i.e., pixels of objects begin to distinguish themselves from pixels of the background. To measure such a difference, we first approximate the pixel-value distributions of the object and background with Gaussians, respectively, and then conduct the Kolmogorov-Smirnov test between the two Gaussian approximations. As shown in Figure 2.5b, σ_{start} is taken when the curve of p-value measured in the

Kolmogorov-Smirnov test begins to diverge.

Our work gives an insight into defining the plausibility-relevant range of noise levels so that the training and sampling effort can prioritize this range. By implementing our two techniques, for BIKED images, we determine σ_{end} to be 0.6 and σ_{start} to be 5.3. This observation can also be seen in real-world images, such as those from FFHQ [72]. Since it is difficult to automatically separate faces from backgrounds, we first track the distribution of pixel values in each RGB channel without distinguish pixels of faces and of backgrounds: as seen in Figure 2.4a, the distribution of pixel values in each color channel starts from an irregular distribution, gradually converges to a Gaussian distribution (at about $\sigma = 0.60$), and then overlaps each other at a certain noise scale (at about $\sigma = 0.60$). To showcase that the above observation can also be seen in the latent diffusion models [139], we first encode 100 BIKED images with a convolutional autoencoder trained on BIKED into a 64-dimensional latent space and then show the evolution of the latent-value distribution (see Figure 2.4). We observe a pretty similar trend in the latent-value distribution as with the pixel-value distribution. Note that the plausibility of design images, i.e., BIKED [137] and Seeing3DChairs [7], is easier to assess, the background can be automatically separated from the main object, and therefore the plausibility-relevant noise range can be more accurately estimated. Thus, this work will focus on structural designs to demonstrate the efficiency of our proposal.

Analysis on more datasets In Figure 2.6 and Figure 2.7, we additionally plot the pixel-value distribution during the perturbation experiment on datasets, e.g., the Seeing3DChairs [7] and Shoes [193].

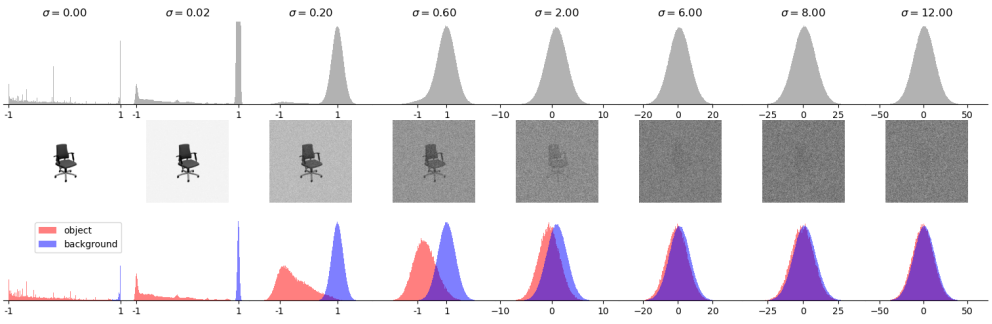


Figure 2.6: Evolution of the pixel-value distribution in perturbation experiments for Chair designs. In the top row, all pixel values are plotted as histograms in gray, while in the next row, the pixels are divided into background pixels and target pixels and then plotted as histograms.

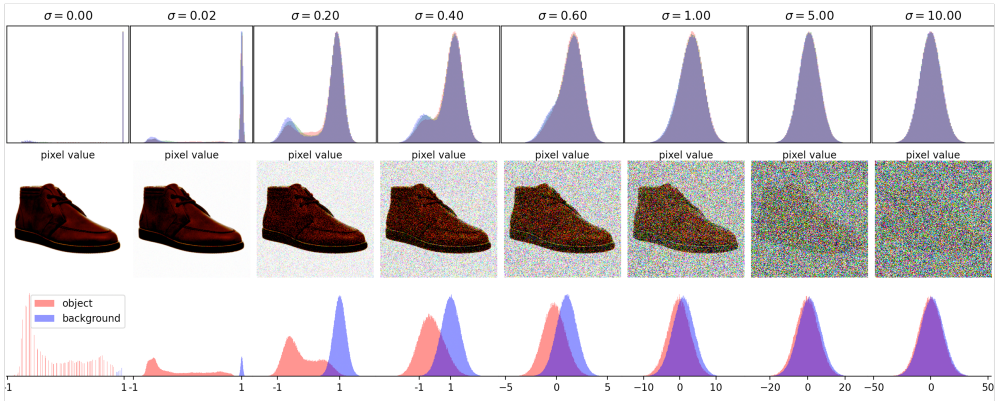


Figure 2.7: Evolution of the pixel-value distribution in perturbation experiments for Shoes designs. In the top row, we plot the pixel values of the three R-G-B channels as the corresponding colors; while in the bottom row, we disregard the color channels and divide them into pixel values (background) and pixel values (object). This plot shows that design images with color channels still follow our observation.

With color images in the FFHQ [72] data set, we track the pixel-value distribution for each color channel with object (human face) and the background separated in two distributions. In Figure 2.8, we plot the evolution of the pixel-value distribution in perturbation experiments. It can be clearly seen that the evolutionary process of FFHQ images is the same as the evolutionary process of the BIKED images, where the three phases can be observed.

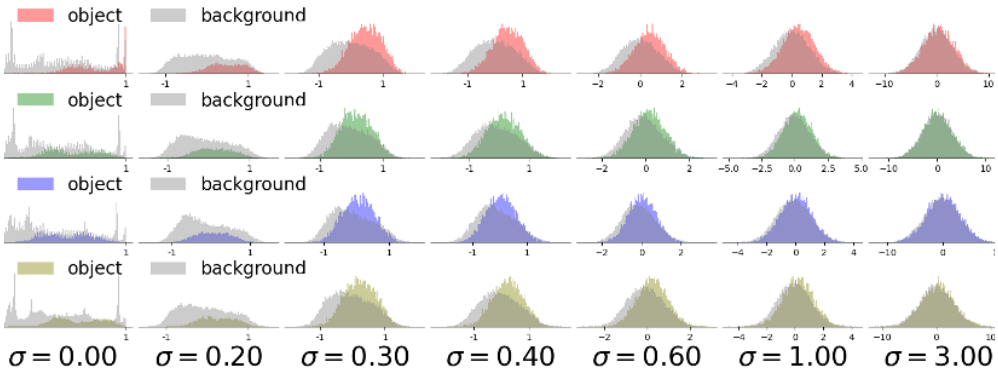


Figure 2.8: Evolution of the pixel-value distribution in perturbation experiments for FFHQ [72] (resolution: 64×64). Here, we manually separate the face from the background pixels. Rows from up to down: the R, G, B color channels and the grayscale.

2.2.3 Training and Generation Procedures

We modify the training and generation procedures so that our diffusion model can concentrate on the plausibility-relevant range of noise levels.

Noise density in training For the structure images, the generation of the structure takes place mostly in the noise range $[\sigma_{\text{end}}, \sigma_{\text{start}}]$ while the noise levels that are too small or large have marginal effects on the plausibility of the final outcome. Hence, it is sensible to sample more noise levels in this interval from the forward SDE. We propose the following log-normal distribution to sample noise levels:

$$\ln(\sigma) \sim \mathcal{N}(\mu,\, \zeta^2),\ \mu = \frac{\ln(\sigma_{\text{start}}) + \ln(\sigma_{\text{end}})}{2},\ \zeta = \frac{\ln(\sigma_{\text{start}}) - \ln(\sigma_{\text{end}})}{2}, \tag{2.7}$$

which implies $\Pr(\sigma \in [\sigma_{\text{end}}, \sigma_{\text{start}}]) \approx 68\%$. In this method, the majority of the noise levels are drawn in $[\sigma_{\text{end}}, \sigma_{\text{start}}]$ while we have ca. 32% probability to sample noise levels at the beginning and the end of the forward process.

Noise schedule for image generation In the backward diffusion process (Equation (2.3)), there are two important factors w.r.t. the noise levels: (1) the noise range $[\sigma_{\min}, \sigma_{\max}]$ in which we apply the ODE (Equation (2.3)) and (2) the decaying noise schedule. For the former, we determine the range based on the training noise density as follows: Equation (2.7) implies that the score function $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, \sigma)$ is trained on noise levels drawn almost in $[\mu-3\zeta, \mu+3\zeta]$, i.e., $\Pr(\log(\sigma) \in [\mu-3\zeta, \mu+3\zeta]) \approx 99.7\%$. Therefore, when sampling new images, applying the reverse-time ODE out of $[\mu-3\zeta, \mu+3\zeta]$ requires the score function to extrapolate, which we have no guarantee about its accuracy. Hence, we set

$$\log \sigma_{\min} = \mu - 3\zeta = 2\log \sigma_{\text{end}} - \log \sigma_{\text{start}}\ , \tag{2.8}$$

$$\log \sigma_{\max} = \mu + 3\zeta = 2\log \sigma_{\text{start}} - \log \sigma_{\text{end}}\ . \tag{2.9}$$

For the latter, we follow the exponential decay in Equation (2.4), where, in addition, we tune the hyperparameter ρ for the BIKED dataset. In Table 2.1, we summarize the tuning results from a simple grid search, where $\rho = 7$ is the best setting. Also, we observe that the performance metrics (e.g., FID, DPS, and PDR) are quite sensitive to ρ, suggesting that tuning this parameter is necessary across different structural image data. Moreover, we calculate the proportion of the noise levels $\{\sigma_{N-1}, \ldots, \sigma_0\}$ (determined by Equation (2.4)) falling into the plausibility-relevant range $[\sigma_{\text{end}}, \sigma_{\text{start}}]$, which we call the prioritization density r_p. It measures how much training effort is

targeted at the structure modeling. In Figure 2.3 and Table 2.1, we show r_p with varying hyperparameter ρ, and we observe that the performance metrics are positively related to it.

We demonstrate a theoretical insight into the noise schedule in Figure 2.3b. On the BIKED data, we depict the norm of score function $\nabla_{\vec{x}} \log p(\vec{x}, \sigma(t))$ over noise levels. Comparing it with the histograms of the noise schedule, we see that PoDM's noise scheduling in sampling/denoising is concordant with the score function, meaning that finer steps of simulating the backward ODE/SDE (see Equation (2.3)) are taken where the norm of the score function is large. We argue that it is sensible to do so since the score function is the major drift term of the backward ODE/SDE.

2.3 Evaluation and Results

In this section, we compare PoDM with several cutting-edge models: the foundational Denoising Diffusion Probabilistic Models (DDPM) by Ho et al. [61], the faster-sampling variant Denoising Diffusion Implicit Models (DDIM) by Song et al. [156], the highly-tuned design-space model Elucidating the Design Space of Diffusion-Based Generative Models (EDM) by Karras et al. [71], and also the non-diffusion generative model Adversarial Latent Autoencoder with Self-Attention for Structural Image Synthesis (SA-ALAE) [165]. We selected these models because they together provide a broad spectrum of generative modeling approaches, sampling schemes, and architectural innovations: DDPM serves as the basic diffusion baseline; DDIM introduces a practical improvement in sampling efficiency; EDM represents the state of the art in diffusion-model design choices (including noise scheduling, preconditioning and sampling strategies); and SA-ALAE offers a useful structural-design-specific comparison beyond diffusion models, targeting complex engineering images.

By benchmarking PoDM against these four models, we are able to situate our contributions with respect to both the canonical diffusion approach (DDPM) and more advanced variants (DDIM, EDM) that already address sampling speed, noise schedule or other design choices, as well as against a structurally-focused adversarial model (SA-ALAE) from the engineering-design domain. Such a comparison allows us to evaluate how much PoDM's novel noise-scheduling strategy adds (i) over the vanilla diffusion chain, (ii) relative to diffusion models that already optimize schedule or sampling protocol, and (iii) in the specific context of plausible structural image synthesis typified by engineering blueprints rather than natural images.

2.3.1 Training Configurations

Our work utilizes the model architecture from the DDIM [156] repository for all diffusion-based models, which follows the U-Net proposed by Ho et al. [61]. More precisely, the implemented model has six feature map resolutions from 256×256 to 4×4, one residual block for each upsampling/downsampling, and an attention layer at the feature map resolution of 16×16. For sampling with DDPM and DDIM, we use the same trained model with default training settings, i.e., timesteps of 1 000 and linear schedule of β with $\beta_0 = 10^{-4}, \beta_T = 0.02$. For EDM, we remove EDM's preconditions, since they did not bring much enhancement to the results according to their experiments, and implement their noise schedules for both training and sampling with default parameters, i.e., $\sigma_{\text{min}} = 2 \times 10^{-3}, \sigma_{\text{max}} = 80, \rho = 7, P_{\text{mean}} = -1.2, P_{\text{std}} = 1.2$. For our PoDM, we determine $\sigma_{\text{start}} = 5.3$ and $\sigma_{\text{end}} = 0.6$ by analyzing the BIKED dataset and inheriting the loss function from EDM. For stochastic sampling in both EDM and our PoDM, we allow the "churn" modification (Equation (2.5)) for all sampling steps, i.e., $S_{\text{min}} = 0$, $S_{\text{max}} = \infty$, and set the S_{churn} to 5. For DDIM, we use 50 as the number of sampling steps, whereas for both EDM and our PoDM, the number of sampling steps is set to 18. The set "Standardized Images" from BIKED Dataset [137] contains 4 512 grayscale pixel-based images with original shape of $1\,536 \times 710$. We pad them with background pixels to a square form with the shape of $1\,536 \times 1\,536$. Then, we reshape these images into a resolution of 256×256 with the scale of $[-1, 1]$ in order to maintain the height-width ratio and ease the complexity in generation. From the whole dataset, we randomly select 100 images for validation, 1 000 images for testing, and the rest of the images for training. We run training on four NVIDIA DGX-2's Tesla V100 GPUs with a batch size of 32 and a learning rate of 5×10^{-5}. Model parameters are saved every 1 000 steps. If the loss converges, we keep training until 100 000 steps and then stop it when the denoising loss does not decrease for 20 epochs. For each model, we select the best-performing model within the saved checkpoints in the last 20 000 steps.

2.3.2 Results

In our work, we evaluate the generative models regarding sampling speed, visual quality, and design plausibility. For the sampling speed, we simply record the sampling time for generating 5 000 images and calculate the sampling speed in FPS (frames per second) for each model. For visual quality, we further use the 5 000 images generated and calculate the FID [58] between the test images and the generated images. The

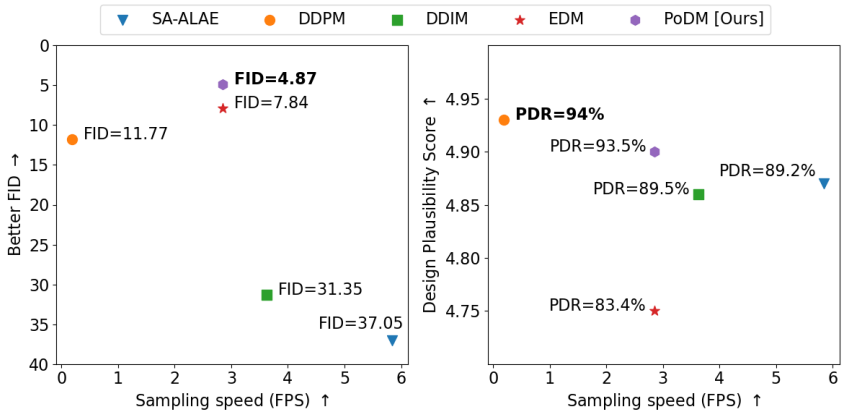


Figure 2.9: On the BIKED data, we show two performance views of the five considered generative models: FID vs. FPS and design plausibility score (DPS) vs. FPS.

measured FIDs are displayed in Figure 2.9.

To quantitatively evaluate the plausibility of generated designs, we implement a human evaluation method, in which the human evaluator bypasses visual qualities (e.g., blurriness and background noise) and scores the represented design in terms of plausibility. We refer to the evaluation score as the design plausibility score (DPS). In this work, the generated bicycle designs are evaluated using a five-point scoring system based on the following criteria:

- No missing fundamental part;
- No floating material or extra part;
- Every part is complete;
- Parts are connected;
- Rational positioning.

For generative model considered, we randomly select 1 000 samples from the 5 000 generated bicycle images. We shuffle all selected images and keep tracking their DPS in a manner that associates each image's score with its corresponding model. This experiment aims to prevent potential biases in the evaluation of the generated images by individual target models and to sustain a uniform evaluation standard across all images. We record the measured DPSs in Figure 2.10 and an average DPS for each model in Figure 2.9. Besides, we calculate the (PDR), which is the proportion of plausible designs, i.e., designs with DPS of 5, in 1 000 generated images.

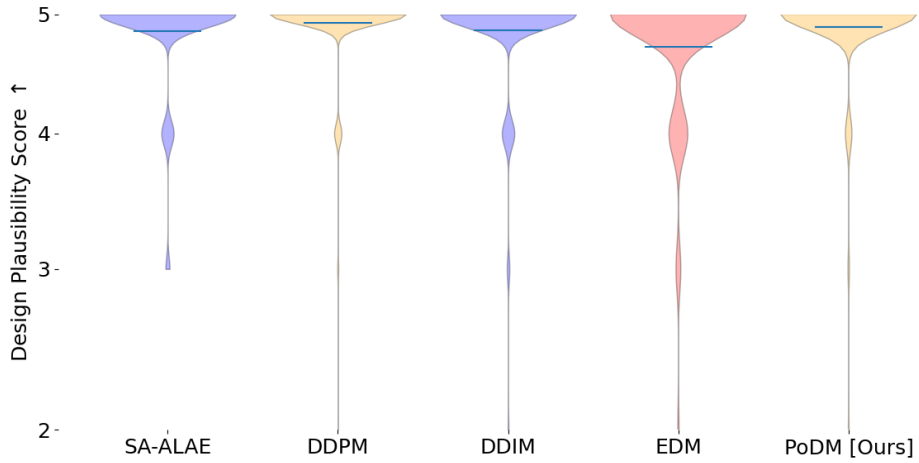


Figure 2.10: On the BIKED data, we show the detailed empirical distribution of the design plausibility scores measured for each model.

On the BIKED data, we first show the overall performance values in Figure 2.9: our PoDM achieves a compelling FID of **4.87**, a satisfactory DPS of **4.90**, and a high plausible design rate of **93.5%**. DDPM [61] requires the longest sampling time of 5.26 seconds for each image but performs decently well in terms of image quality, i.e., FID of 11.77, and design plausibility, i.e., DPS of 4.93 with only 6.0% implausible outcomes. As shown in Figure 2.9, EDM [71] can significantly improve the sampling speed to 2.85 FPS and even enhance the visual quality to a FID of 7.84. However, EDM performs poorly in design plausibility compared to PoDM, i.e., DPS of 4.75 and a plausible design rate of 83.4%. As seen, DDIM and EDM demonstrate a trade-off between visual quality and plausibility of generated images, whereas the DDPM leverages extremely slow sampling speed to perform decently in both aspects. We need to address that although it seems that DDPM's DPS value (= 4.93) is slightly higher than PoDM's (= 4.90), there is actually no statistical difference between them (based on a Mann–Whitney U test). Hence, we state that our PoDM method can achieve the same design plausibility, a better FID, and a much faster generation/sampling speed than DDPM.

In Figure 2.16, Figure 2.17, Figure 2.18, Figure 2.19, Figure 2.20, we plot a certain number of randomly generated bicycle designs for each model trained on the BIKED dataset, respectively. Here, we provide the results for the qualitative evaluation.

We additionally train PoDM and EDM on Seeing3Dchair [7] images of resolution



(a) Chairs generated by EDM with scores: DPS-4.13, DPR-51.5%, FID-33.48



(b) Chairs generated by our **PoDM** with scores: DPS-**4.65**, DPR-**74.5%**, FID-32.15

Figure 2.11: Generated chair designs with limited training epochs

(128×128) with only 100 epochs. After training, we showcase the generated designs for a visual comparison in Figure 2.11. Visually, PoDM generates much more plausible chairs than EDM. Measured on 1k generated images, the mean design plausibility score (DPS) and design plausibility rate (DPR: DPS=5) are: PoDM (DPS 4.65, DPR 74.5%, FID 32.15); EDM (DPS 4.13, DPR 51.5%, FID 33.48). Overall, we state that our method can *significantly improve the speed of generating structural designs while maintaining their plausibility.*

2.3.3 Alignment Test

It is not surprising that the most-used automatic metrics in the generative modeling community do not align well with human judgments and fail to capture the plausibility of generated designs. To demonstrate this, we visualize the correlation between human evaluation results and metrics results in Figure 2.12.

In our procedure we collect human plausibility ratings for generated designs and compare them with automatic metric scores: Fréchet Inception Distance (FID), which measures the distance between the distribution of generated images and the distribution of reference real images based on deep feature statistics; the Structural Similarity Index (SSIM), which assesses luminance, contrast and structural similarity between two images; and the Learned Perceptual Image Patch Similarity (LPIPS), which computes deep-feature distances learned to match human perceptual judgments. What we observe is only a weak or moderate alignment: designs judged more plausible by humans do not consistently receive better metric scores. This highlights a risk of relying solely on these metrics when structural plausibility is the focus of evaluation.

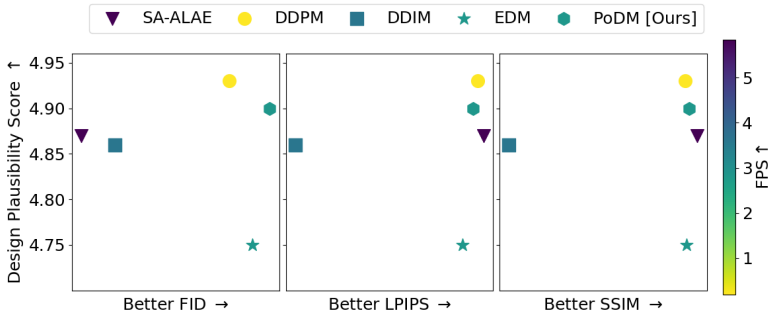
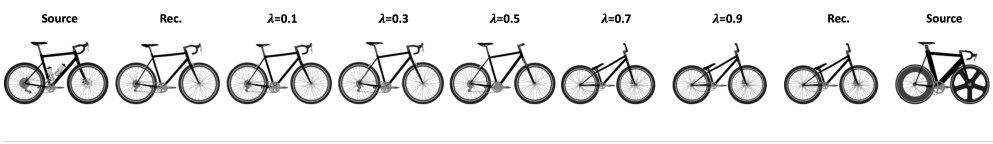


Figure 2.12: Alignment test. Here we test the alignment between human evaluation results (design plausibility score) and various metric results. The plot shows that non of them have a strong correlation to the human evaluation.

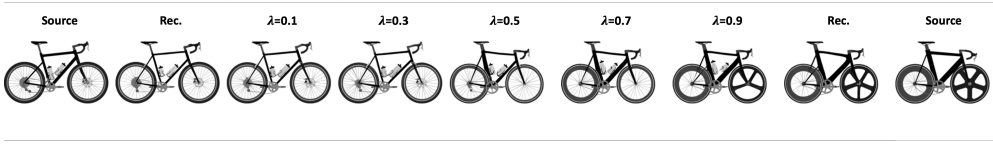
2.4 Controllable Generation and Design Editing

In this section, we test the PoDM's understanding of structural design space by applying cutting-edge image editing methods, e.g., interpolation via latent space, point-based dragging and inpainting, on bicycle designs.

Interpolation via latent space Interpolation via latent space can be quite useful



(a) Interpolation with latent space set at noise level $\sigma_{\max}$



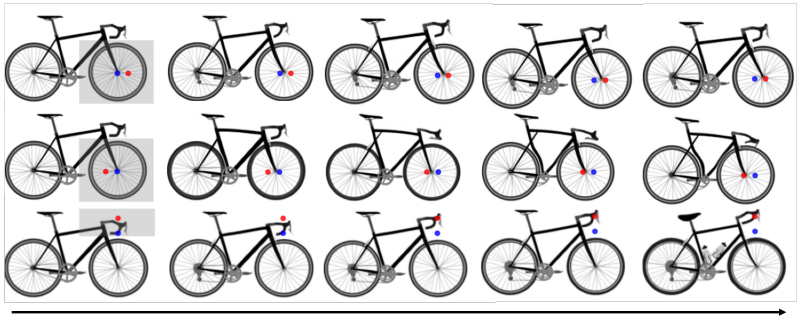
(b) Interpolation with latent space set at noise level σ_{start}

Figure 2.13: PoDM-driven structural interpolation via latent space set at various perturbation steps. In (a), the reconstruction has a poor accuracy, and interpolation fails to produce intermediate structures. In (b), the interpolation displays a smooth transformation between two source structures.

in exploring structural design space. After encoding a source data $\boldsymbol{x}(0)$ to pure noise $\boldsymbol{x}(T)$ via the forward process, the diffusion model is supposed to decode $\boldsymbol{x}(T)$ back to $\boldsymbol{x}(0)$ by utilizing a corresponding ODE [159]. However, in our implementation shown in Figure 2.13a, PoDM-motivated reconstruction has poor accuracy, which might be caused by the prioritizing strategy. We argue that it is unnecessary to conduct the forward process completely, instead, perturbed images at noise level σ_{start} retain good reversibility. Taking images at noise level σ_{start} as latent code allows well-performing reconstruction and interpolation, as shown in Figure 2.13b.

Point-based dragging As a novel image editing method, point-based dragging [122, 150] can precisely and iteratively "drag" the handle point to a target point and the remaining parts of the image will be correspondingly updated to maintain the realism. We implement DragDiffussion [150] on BIKED images and plot the results in Figure 2.14a. To the best of our knowledge, our work is the first to apply point-based dragging on structural design.

Extending the target design geometry is common in the day-to-day work of engineering, but it is still very time-consuming because engineers need to manually modify each related part to edit the geometry of the final design. Meanwhile, our work has shown that after the DGM is trained with historical design data and the geometric dependencies among parts are captured accordingly, the DGM can perform this dragging task. In Figure 2.15a, we showcase the advantages of using DGMs in geometric editing tasks. Inspired by this, we assume that with the same optimization pipeline,



(a) PoDM-driven structure editing with DragDiffusion [150]



(b) PoDM-driven inpainting with RePaint [97]

Figure 2.14: PoDM-driven structural interpolation via latent space set at various perturbation steps. In (a), from left to right, the handle point is iteratively dragged from the initial handle point (*blue*) towards the selected target point (*red*). In (b), the part enclosed by a *dashed line* is redesigned with RePaint.

DGMs can also modify the target design to achieve better functional performances, which will be detailed described in Section 6.1.

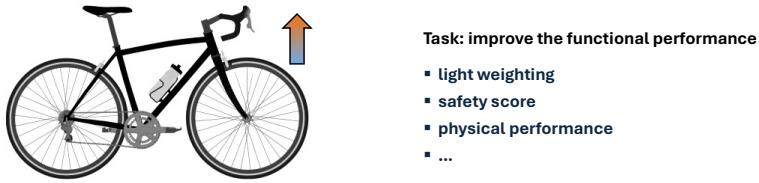
Inpainting In an inpainting task, the generative model is tasked to generate the inpainting area to match the known part. A DDPM-based inpainting mechanism, RePaint [97], has achieved the state-of-the-art performance on diffusion-based inpainting tasks by utilizing the known part as guidance at each step. We adapt RePaint to our PoDM and test it on BIKED images. The inpainting results are shown in Figure 2.14b.

2.5 Conclusion

When generating engineering designs with DGM, the primary task is to ensure that the generated design is reasonable, whereas the current DGM cannot achieve this well. We observe that the performance of the diffusion-based generative models exhibits a trade-



(a) Intuitive design editing task



(b) Function-oriented design editing task

Figure 2.15: Scenarios of engineering day-to-day operation tasks. Inspired by the geometric operation task shown in (a), we assume that with the same optimization pipeline DGMs can also edit the design to achieve a better functional performance.

off among visual quality, the plausibility of generated images, and sampling time. We assume that there is a range of noise levels, that is responsible for the plausibility of the outcome, especially in generating structures. Following this observation, we propose a plausible-oriented diffusion model (PoDM) that leverages a novel noise schedule to prioritize this range of noise levels in both training and sampling procedures. We observe that the well-known EDM has a poor performance in generating plausible structures. Our PoDM method significantly improves the plausibility of generated images over EDM and also achieves a satisfactory plausibility score comparable to DDPM but with a much-reduced generation time. Additionally, we demonstrate with convincing results that the improvement in the plausibility thanks to the prioritization of the determined noise range. Further implementations of PoDM-driven image editing tools showcase PoDM's ability to semantically manipulate complex structural designs, paving the way for future work in the field of generative design.

This chapter is inspired by, but not limited to, engineering design generation. We believe that our observations and determinations of the phases in the diffusion process are equally applicable to images from natural scenes and, therefore, beneficial for all diffusion-based synthesis tasks. In addition, we hope that our work will inspire more research on the tool for automatically evaluating the plausibility of generated images and the relevance between noise level and generated features.

Limitation The designs generated in this chapter are assessed through both qualitative and quantitative evaluations. However, we have found that the existing metrics do not align well with human judges' assessments in terms of plausibility, as detailed in Section 2.3.3. This misalignment raises concerns about the reliability of the current evaluation methods. To address this issue, we have developed a design plausibility score, which is manually evaluated as an assessment metric. Unfortunately, this process is not automated, making it time-consuming and less efficient. Therefore, it is crucial and urgent to create a novel metric that can effectively evaluate plausibility while aligning with human evaluation standards. Achieving this goal will enhance the accuracy and relevance of our assessments. In the next chapter, we will focus on developing such a metric for result plausibility, exploring innovative approaches that can bridge the gap between automated evaluations and human judgment.

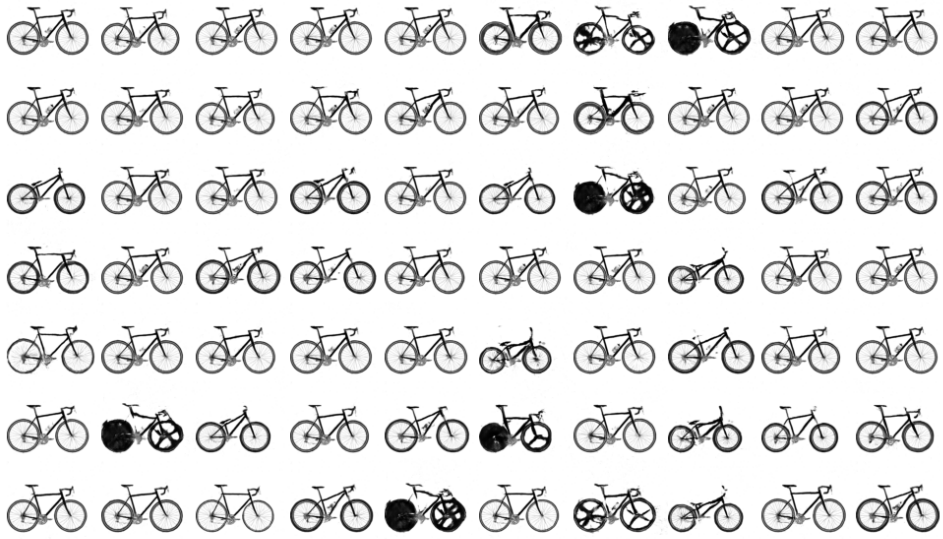


Figure 2.16: BIKED images randomly generated by SA-ALAE [165]. SA-ALAE shows an uneven performance over various classes bikes and a great portion of generated designs are of poor quality and present implausible structures.

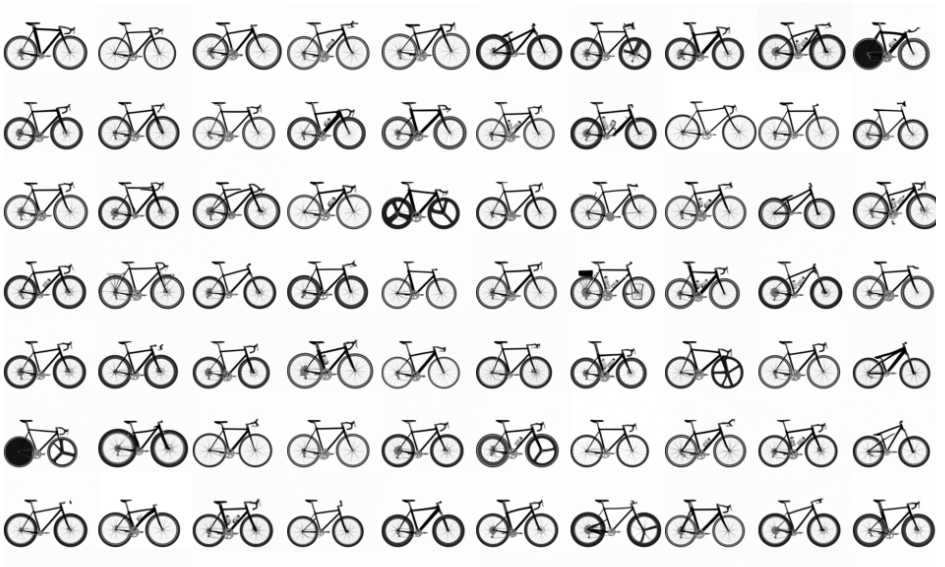


Figure 2.17: BIKED images randomly generated by DDPM [61]. DDPM presents a strong generative power in both visual quality and structural plausibility. However, DDPM requires always a tremendous number of denoising steps (i.e., 1 000), otherwise the results look like in Figure 2.18.

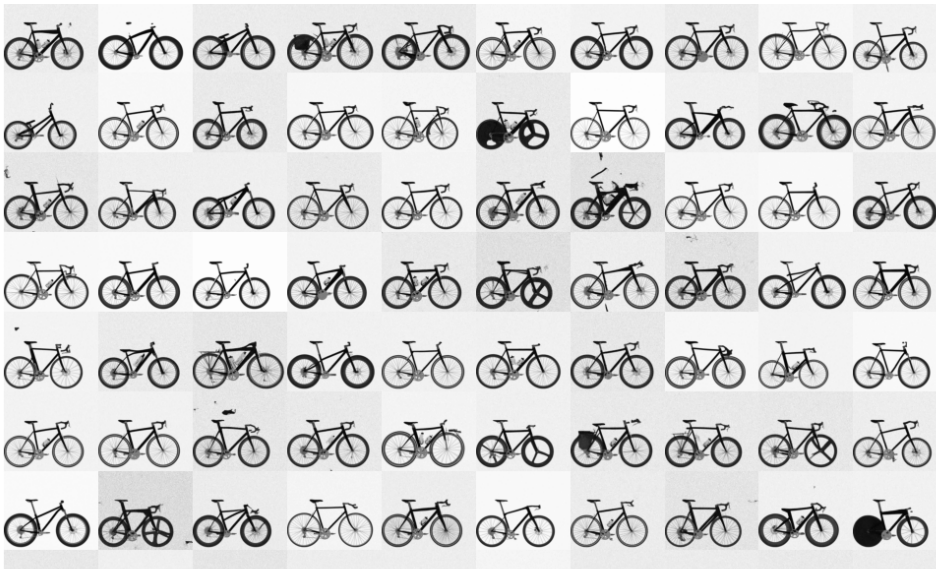


Figure 2.18: BIKED images randomly generated by DDIM [156]. DDIM leverages the same trained backbone model as DDPM, but attempts to break the Markov-chain of DDPM and to use a reduced number of denoising steps (i.e., 50). Hereby, the generated images present poor visual quality.

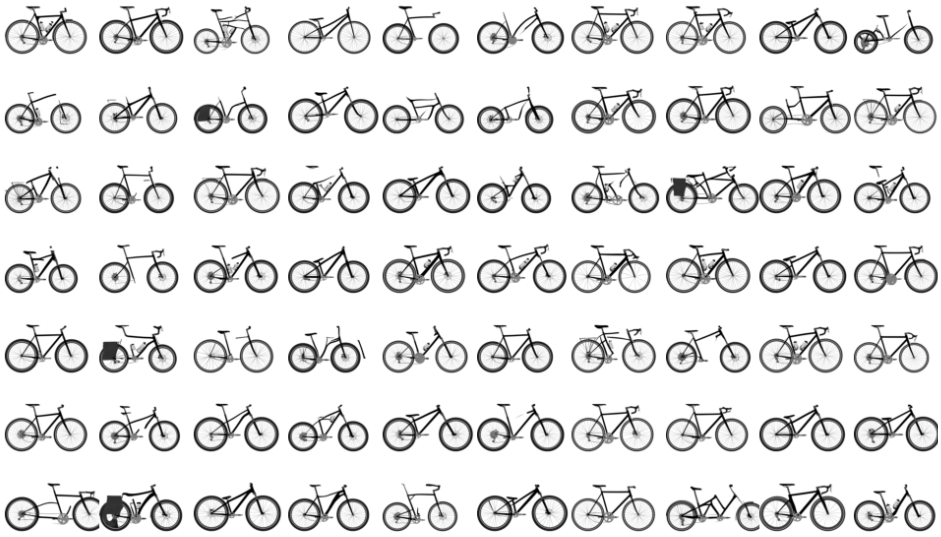


Figure 2.19: BIKED images randomly generated by EDM [71]. EDM is theoretically based on Score-matching models [159], but attempts to significantly reduce the sampling number by focusing on a pre-defined range of noise scales. Compared to DDIM, EDM has indeed deliver DDPM-like visual quality images, but we observe that it fails badly in design plausibility.

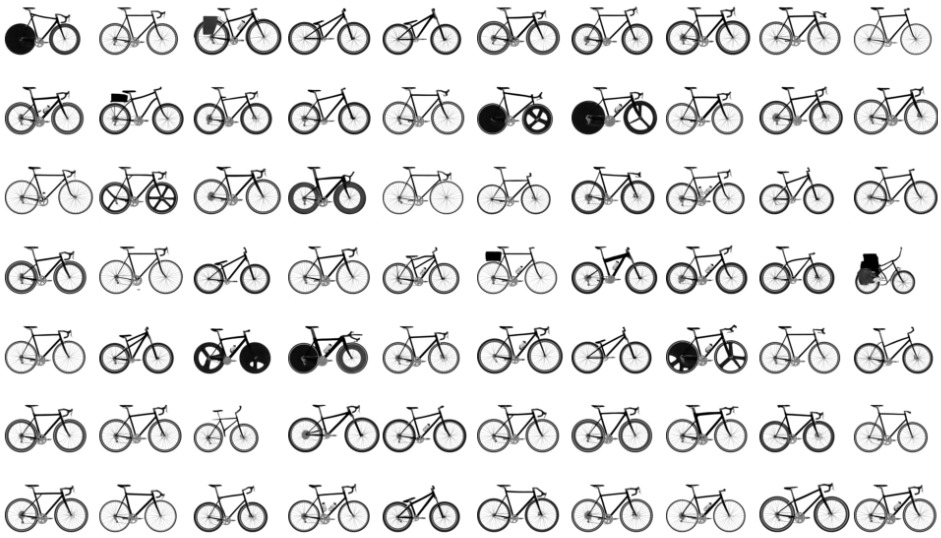
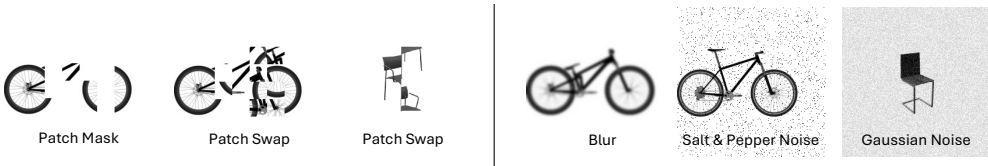


Figure 2.20: BIKED images randomly generated by PoDM. Based on the achievement of EDM, our work figures out a way of locating the focusing range of noise scales and hereby well-addresses the trio-trade-off among sampling time, visual quality and design plausibility.

Chapter 3

Evaluating the Plausibility with Deep Learning

We have pointed out the evaluation issue in the previous chapter, i.e., the evaluation of FID does not align well with human judge in terms of the design plausibility. In this chapter, we are addressing this issue as well as the research question 2: *How to automatically evaluate the plausibility of designs generated by DGMs?*. The content of this chapter has been published in the paper [164].



(a) Better scored by the SOTA metrics (e.g., FID, KID and $FD_{DINO\text{-}V2}$)

(b) Better scored by human designers and our FDD

Figure 3.1: From which side (*left* or *right*) are the design images more plausible? (a) Structural implausibility; (b) visual artifacts. Recent works [10, 46, 57] discover that the SOTA metrics (FID, KID and $FD_{DINO\text{-}V2}$) tend to penalize visual artifacts more than structural implausibility, which matters more to the human designers. In contrast, our FDD consists better with human designers and is able to focus on shapes.

3.1 Introduction

Design data is responsible for representing the design object with structural and geometric patterns, which are required to be recognizable and plausible. In order to rank models during the development of generative models, recent works rely on a subjective evaluation [167, 99], where human experts apply an established set of criteria to manually assess a significant quantity of generated data. This evaluation method yields reliable results, serving as "ground truth" for model ranking, but it is time-consuming and hard to reproduce [99]. Hence, for developing DGMs for design generation, it is necessary to have an automated metric, which is able to reliably quantify the goodness of the target DGM.

Meanwhile, the evaluation of generated images is still an unsolved challenge among other general tasks in the DGM domain [11, 112, 14]. DGM developers [70, 58, 72] are heavily relying on the Fréchet inception distance (FID) [58] metric, which extracts latent features from real and generated images with an Inception-V3 [163] model pre-trained on ImageNet [38] respectively and then quantifies their difference using Fréchet distance as the final FID score. As the primary metric in the DGM field, FID is able to measure the fidelity and diversity and present them in a single value. However, a lot of studies [18, 161, 83] disclose that FID does not always align with human evaluation and claim that this limitation is due to the reliance on the pre-trained Inception-V3 model. Hence, novel metrics are delivered by replacing the Inception-V3 model by other backbone networks, e.g., Clip [134], VQ-VAE [173] and DINOv2 [119], etc. According to the most recent work by Stein et al. [161], where they compared 17 metrics using encoders from 9 various networks, $\mathrm{FD}_{\text{DINO-V2}}$ has the most reliable performance in terms of consistency with human judgment in their experiments.

On the other hand, recent works have pointed out that the Inception-V3 model and Inception-powered metrics perform poorly on shapes [10, 46, 57, 167]. Our work investigates this finding and observes that the state-of-the-art (SOTA) metrics generally suffer from this issue: they are sensitive to visual artifacts like noises, yet they have a high tolerance towards semantic failures, e.g., part missing in a bicycle, as illustrated in Figure 3.1. Besides, human experts are able to recognize the same structural representation of the observed design image regardless of minor noise and they tend to penalize the evaluation based on the implausibility of the design more, rather than the presence of visual artifacts [86, 82]. Motivated by this, our work aims to create a novel metric for generative design that is robust to visual corruption of the observed images and biased towards the design plausibility.

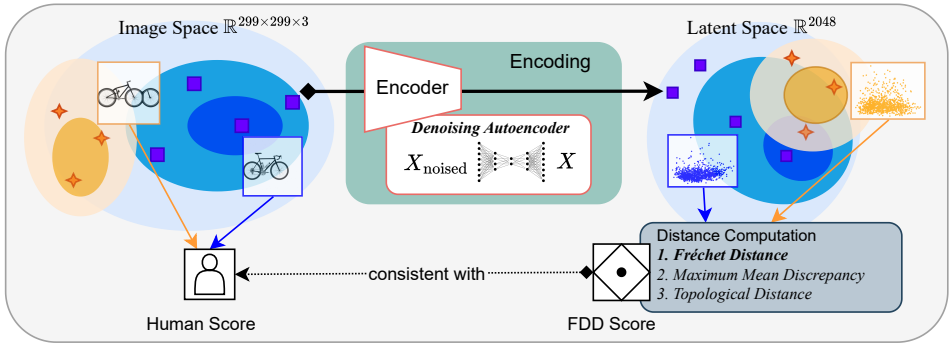


Figure 3.2: Plausibility evaluation using Fréchet denoised distance. *Blue area and squares* visualizing the distribution and samples of real data in the image space and in the DAE-encoded latent space; *Orange area and stars* illustrating the distribution and samples of generated data in the image space and in the DAE-encoded latent space.

Finally, we propose the Fréchet denoised distance (FDD) by replacing the Inception-V3 model within the FID framework with a denoising autoencoder (DAE) [176] that has been also pre-trained on ImageNet dataset and capable of encoding images into latent features with an Inception-comparable dimension of $\mathbb{R}^{2048}$. The DAE is able to observe the same structural representation in the image regardless of the noisy disturbances, which can be utilized as a strong method to extract the structural feature from the noisy input. Our work compares our FDD with other SOTA metrics, e.g., FID, $\text{FD}_{\text{DINO-V2}}$ and topology distance (TD) [62] (since their results show a similar bias to our intention), based on the following experiments: (1) sensitivity test over visual artifacts and structural failures; (2) consistency test with increasing disturbances; (3) consistency test with human judgment in model ranking. As a result, our FDD has the most stable performance among all the experiments. In order to explain the performance of FDD, we visualize the "focus" of our FDD metric compared to the FID using a GradCAM [161, 148, 83] test, hereby showcasing that the DAE model has a better assessment regarding the requirements of human designers. In addition, our work explores the potential for further improvement of DAE-based metrics, where we build-upon concepts from existing works, i.e., KID [16], TD [62] and training the network on structural images [46], to design new DAE-based metrics, i.e., kernel denoised distance (KDD), topology denoised distance (TDD), and FDD $(\cdot)$, respectively. We test these metrics on BIKED and the results show that these DAE-based metrics are highly correlated and FDD performs relatively best.

3.2 Related Work on DGM Evaluation

Accurately ranking generative models remains an unresolved challenge [11, 112, 14]. Humans are able to give the ground-truth evaluation in assessing a limited number of generated images, but quantifying the performance of a DGM requires an automated evaluation method [161]. Overcoming the flaws of previous metrics, e.g., SSIM [202], LPIPS [198] and IS [143], currently most reported evaluation methods, e.g., Fréchet inception distance (FID) [58] and kernel inception distance (KID) [16], have largely addressed the challenge of automated evaluation and are employed as the primary metric for model ranking in the field of DGM. They leverage a two-step procedure: encode real and generated images into latent features in a lower-dimensional space with a representation extractor and then use a distance critic to quantify the difference between their features. Both FID and KID utilize the Inception-V3 [163] model pre-trained on ImageNet, which has a 2048-dimensional latent space. Regarding the measurement of latent distance: FID fits the Inception features from real and generated images into a multivariate Gaussian before computing the Fréchet Distance (also known as the Wasserstein-2 distance) between them; whereas KID [16] uses the squared maximum mean discrepancy (MMD) [50] with a polynomial kernel [14].

Concerns about the over-reliance on the Inception-V3 model have been raised and researchers claim that an ImageNet [38] classifier like the Inception-V3 model brings a significant bias to the evaluation with FID [112, 119]. Furthermore, FID is proved to be vulnerable to manipulation [83], especially when there exists a significant domain discrepancy between the data set of interest such as BIKED [137] and ImageNet [38]. Consequently, the results measured by FID often show a poor correlation with human judgments. Similarly, KID [16] encounters the same issue as it also leverages the pre-trained Inception-V3 model. Most recently, in order to find a perceptual representation space superior to the inception manifold, Stein et al. [161] studied 17 metrics with 9 different encoders (e.g., CLIP [134], SwAV [24] and DINOv2 [119]). Their finding concludes that $\text{FD}_{\text{DINO-V2}}$ [161] demonstrates the most reliable performance over various perspectives, e.g., fidelity, diversity, rarity, and memorization of generative models. Previous works [10, 46, 57] shed light on the role played by the image attributes, e.g., edges, shapes, textures, and colors in various computer vision tasks, e.g., classification and segmentation. They revealed the limitation of ImageNet-trained CNNs in recognizing shapes. This flaw may explain the inconsistency of CNN-based metrics with human judgments when evaluating design images, where human experts prefer to use shape information for assessment [86, 82]. In other studies, new metrics have been

proposed to evaluate fidelity and diversity, including density and coverage [112], as well as precision and recall [141, 84].

Recent studies have introduced autoencoder-based metrics for evaluation purposes: for instance, Buzuti et al. [21] leveraged the VQ-VAE [173] and showed that their unsupervised model-based metric Fréchet autoencoder distance (FAED) outperforms FID in terms of consistency with increasing disturbance when evaluating on human and animal faces, i.e., CelebA HQ [95], FFHQ [72], and AFHQ [31]. By cross-comparing their measured values among various types of disturbance, their FAED noticeably penalizes visual artifacts more severely than structural implausibility with comparable intensity. This may still lead to unfair comparisons of DGMs for design synthesis, where human experts prefer to use shape information for assessment [86, 82]. Meanwhile, Horak et al. [62] proposed a more competitive shape-based evaluation metric by investigating the topological characteristics of potential flow shapes and proposed topological distance (TD) as a complementary metric for FID. Thus, we choose TD as for the later comparison.

3.3 Preliminaries

Fréchet Inception Distance (FID) The FID leverages the Inception-V3 model pretrained on ImageNet without its last fully connected layer. Hereby, it provides a lower-dimensional latent space. Real images $\boldsymbol{x}$ and generated images $\boldsymbol{x'}$ are embedded into the Inception features $\boldsymbol{w} \in \mathbb{R}^{2048}$ and $\boldsymbol{w'} \in \mathbb{R}^{2048}$, respectively, and then separately fitted into two multivariate Gaussian distributions, with $(\mu_{\boldsymbol{w}}, \Sigma_{\boldsymbol{w}})$ and $(\mu_{\boldsymbol{w'}}, \Sigma_{\boldsymbol{w'}})$ denoting the means and covariances thereof. The difference between the two latent manifolds will be quantified with Fréchet distance with:

$$\mathrm{FD} = \|\mu_{\boldsymbol{w}} - \mu_{\boldsymbol{w'}}\|_2^2 + \mathrm{Tr}(\Sigma_{\boldsymbol{w}} + \Sigma_{\boldsymbol{w'}} - 2(\Sigma_{\boldsymbol{w}}\Sigma_{\boldsymbol{w'}})^{\frac{1}{2}}), \tag{3.1}$$

where $\mathrm{Tr}(\cdot)$ computes the trace of a matrix.

Denoising Autoencoder (DAE) The DAE [176] is able to observe the same structural representation in the image regardless of the noisy disturbances, which demonstrates its robustness in assessing structural plausibility. The architecture of DAE is based on an expansion of the fundamental autoencoder model, consisting of two components: an encoder $(E_\theta \colon \boldsymbol{x} \to \boldsymbol{w})$ and a decoder $(D_\theta \colon \boldsymbol{w} \to \boldsymbol{x})$. In the training phase, source images $\boldsymbol{x} \in \mathbb{R}^{w \times h \times c}$ are corrupted with Gaussian noises $\boldsymbol{x}_\eta = \boldsymbol{x} + \eta$,

where $\eta \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$ and σ refers to the noise scale. The encoder (E_θ) embeds the noised image $\boldsymbol{x}_\eta$ into its lower-dimension latent representation $\boldsymbol{w} = E_\theta(\boldsymbol{x}_\eta)$, then the decoder restores the latent representation back into pixel-based image space $\hat{\boldsymbol{x}} = D_\theta(\boldsymbol{w}) = D_\theta \circ E_\theta(\boldsymbol{x}_\eta)$. The network is trained minimizing the following loss function:

$$\min_{E_\theta, D_\theta} \Delta(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - D_\theta \circ E_\theta(\boldsymbol{x}_i + \eta))^2 \,, \tag{3.2}$$

where n is the batch size. While minimizing the reconstruction error in the training, denoising autoencoder (DAE) in turn maximizes the mutual information between the original input $\boldsymbol{x}$ and its latent representation $\boldsymbol{w}$ [176]. More specifically, DAE bypasses the noisy corruption between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$. This allows the latent representation $\boldsymbol{w}$ to contain meaningful information about the source $\boldsymbol{x}$, even thought DAE only sees the corrupted input $\hat{\boldsymbol{x}}$.

3.4 Method

We implement the encoder $E_\theta(\boldsymbol{x}_\eta)$ of the denoising autoencoder (DAE) as the feature extractor. First, we design a DAE architecture (refer to Section 3.5.2 for more information on this architecture) and train it on the ImageNet [38] dataset with input shape of $299 \times 299 \times 3$. Second, similarly to the procedure of FID, we embed a certain number K of real images $\boldsymbol{x}$ and generated images $\boldsymbol{x}'$ into the latent features $\boldsymbol{w} \in \mathbb{R}^{2048}$ and $\boldsymbol{w}' \in \mathbb{R}^{2048}$, respectively. Note that the image is preprocessed into a shape of $299 \times 299 \times 3$ regardless of the original shape and color. Next, we follow the procedure of the Fréchet distance, introduced in Section 3.3, to quantify the difference between the two manifolds $\boldsymbol{w}$ and $\boldsymbol{w}'$. Hereby, we design the Fréchet denoised distance (FDD), illustrated with an explanatory diagram in Figure 3.2.

For exploration purpose, we simulate the design processes of KID [16] and TD [62] and replace the distance measures with maximum mean discrepancy (MMD) and topology distance (TD), hereby delivering more DAE-based metrics, e.g., kernel denoised distance (KDD) and topology denoised distance (TDD). We also notice the work of [46] that trains a ResNet-50 [55] model on an alternative dataset of ImageNet, i.e., Stylized-ImageNet, and hereby successfully develops a shape-biased classifier. Inspired by this proposal, we additionally train a DAE model from scratch on the BIKED [137] dataset. The DAE model trained on BIKED images has an input shape of $256 \times 256 \times 1$ and a smaller latent space with dimension $D_{\boldsymbol{w}} = 64$. Hereby, we design a FDD $(\cdot)$

metric, which is based on the DAE trained on the same target dataset. The evaluation of FDD $(\cdot)$ on BIKED images is demonstrated in Section 3.5.

3.5 Experiments

To evaluate the design plausibility of generated images, a useful metric should satisfy the following conditions: (1) bias toward design structure, (2) consistency with increasing disturbances, and (3) alignment with human judgment. Hence, we leverage correspondingly three experiments: sensitivity test, consistency test with increasing disturbances, and model ranking, over the the SOTA metrics and our metrics (see Table 3.1 for more detailed information). Note that in our work, the TD metric refers to TD-Inception [62] unless otherwise explained.

3.5.1 Datasets

We select a variety of datasets covering different aspects. For a fair comparison with the FID, we train the DAE on the ImageNet [38] dataset, ensuring that the learned feature manifold is similar to the one of the Inception-V3 model. We employ a subset of the ImageNet [38] dataset of 50 000 samples with dimension $299 \times 299 \times 3$, properly chosen to cover a wide range of 1 000 classes. The dataset is divided into 45 000 training samples and 5 000 test samples. Our comparative analysis and tests also incorporate two design datasets, BIKED [137] and Seeing3DChairs [7], to address the interests of human designers. Additionally, we incorporate the color-channeled FFHQ [72] dataset and the test samples of ImageNet [38] into our metric testing to confirm the metric's adaptability to general image generation tasks.

3.5.2 Experimental Settings

For the reproducibility of our work, this section documents all the essential details regarding the development of our FDD metric and the experimental setups. To justify the setting choices, we aim to align our DAE model's architecture with that of the Inception-V3 model, particularly in terms of input shape and latent dimension. The model architecture and training settings describe the DAE trained on ImageNet, whereas the configurations of the DAE trained on BIKED are correspondingly adjusted as shown in Table 3.1.

Table 3.1: A list of candidate performance metrics for measuring design plausibility. Below the *dashed line*, we also list other DAE-based metrics as a means of exploring further improvements. FDD (·) utilizes a DAE trained on the target dataset with the DAE architecture modified according to the dataset, e.g., FDD (BIKED) utilizes a DAE trained on the BIKED dataset.

Metric	Backbone Model	Input Dimension	Feature Dimension	Training Dataset	Distance Measures
FID [58]	Inception-V3 [163]	$299 \times 299 \times 3$	2048	ImageNet [38]	Fréchet distance
KID [16]					MMD
$FD_{DINO\text{-}V2}$ [161]	DINOv2 [119], ViT [78]	$224 \times 224 \times 3$	1024	LVD-142M [161]	Fréchet distance
TD-Inception [62]	Inception-V3 [163]	$299 \times 299 \times 3$	2048	ImageNet [38]	Topology distance
TD-ResNet [62]	ResNet18 [55]	$224 \times 224 \times 3$	512	Fashion-MNIST [183]	
FDD	DAE [176]	$299 \times 299 \times 3$	2048	ImageNet [38]	Fréchet distance
KDD	DAE [176]	$299 \times 299 \times 3$	2048	ImageNet [38]	MMD
TDD					Topology distance
FDD (·)		$256 \times 256 \times 1$	64	Target Dataset	Fréchet distance

Model architecture Our approach employs a DAE comprising 5 convolutional layers across both the encoder and the decoder. Here, the feature dimensions for the convolutional layers in the encoder are arranged in the following sequence [32, 64, 128, 256, 512]. For the decoder, these dimensions are applied in reverse order. Each layer employs a 3×3 kernel shape, a stride of 2, padding of 1, and the `Rectified Linear Unit (ReLU)` as the activation function, aligning with the Inception-V3 model. The last activation layer of the decoder uses a `Tanh` function to adjust the outputs to a pixel range of $[-1, 1]$. In alignment with the configuration parameters of the Inception-V3 model, the encoder's input shape is specified as $299 \times 299 \times 3$, and the latent vector dimension is established at 2048.

Training settings The training process uses a subset of 45 000 images from ImageNet [38], which are rescaled to the range $[-1, 1]$. For the DAE training set-up, input images are corrupted with Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$ with $\sigma = 0.1$ before being fed into the encoder. We utilize the `Adam` Optimizer with a learning rate of 1e−3 to train the DAE with a batch size of 128 and epochs of 1 000. The reconstruction loss is assessed by calculating the mean squared error (MSE) between the original and output images. Model performance is continuously assessed during training, and the best-

performing model is chosen from the saved checkpoints for further experiments. We also implement an early stop function, where the training stops if the reconstruction loss does not reduce within 20 epochs.

Disturbance procedures For conducting the sensitivity and consistency tests that exam metrics' performance in dealing with various disturbances, we design the perturbation methods, i.e., salt & pepper noise, Gaussian noise, patch mask, patch swap and a mixed disturbance of Gaussian noise and patch swap, and their respective intensity levels based on previous studies [58, 62]. The details of the disturbances are outlined below:

- **Pepper Noise.** Salt & Pepper Noise is characterized by the random conversion of image pixels to black or white. In our experiments, we specifically target pixels to turn black (i.e., pepper noise), considering the prevalent white backgrounds in most design images. The proportion of image pixels altered to black, effectively setting their value to 0, is determined by a factor α within the set $[0, 0.01, 0.02, 0.03]$.

- **Gaussian Noise.** We generate a random Gaussian noise in matrix form, $\boldsymbol{\eta} = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then we create noisy images $\boldsymbol{x}'$ by adding the defined Gaussian noise to the source image $\boldsymbol{x}$: $\boldsymbol{x}' = (1-\alpha)x + \alpha\mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\alpha \in [0, 0.1, 0.2, 0.3]$ refers to the intensity of the noise. The larger α is, the more intensive the disturbance of the source data is.

- **Gaussian Blur.** We apply a Gaussian blur to the images using a convolution operation with a Gaussian kernel. The standard deviation of the kernel, determined by α, varies from $[0, 1, 2, 3]$, resulting in progressively more blurred images.

- **Patch Mask.** For design images (BIKED and Seeing3DChairs), we evenly divide the focus area of each image (where the design object is usually located) into 16 patches. For the FFHQ-256 dataset, the entire image is segmented into 64 patches. Afterward, we randomly select a portion of patches denoted by $\alpha \in [0, 0.25, 0.5, 0.75]$ and apply a white mask to them.

- **Pepper Swap.** Using the same patch division approach as the Patch Mask, we randomly select a subset of patches, indicated by $\alpha \in [0, 0.25, 0.5, 0.75]$, and swap their positions pair-wisely.

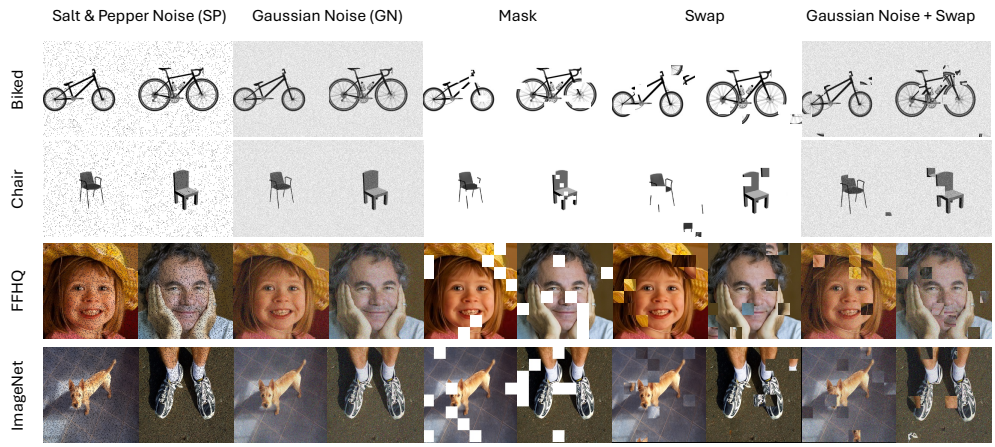


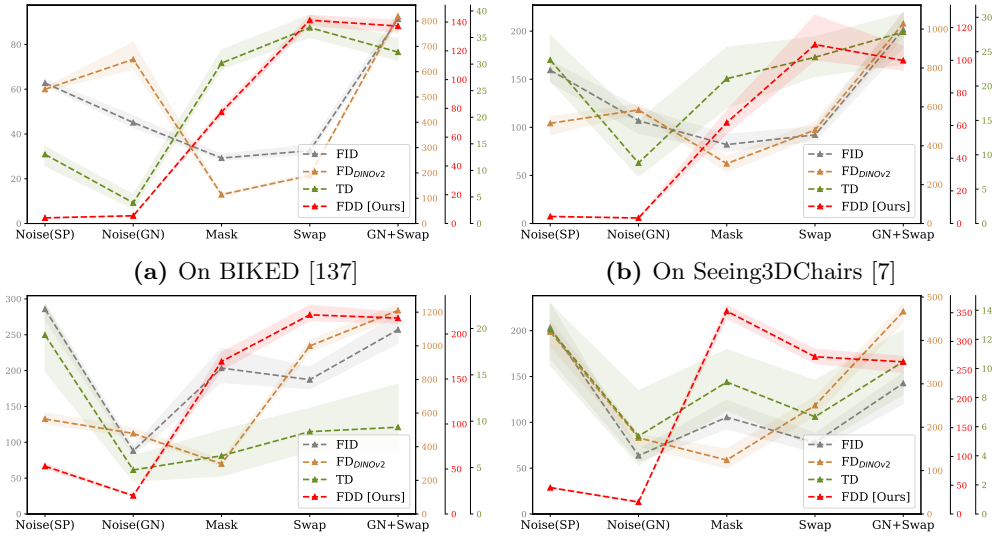
Figure 3.3: Examples of manipulated images for sensitivity test. We choose the intensity of the disturbances so that images with structural errors (i.e., mask and swap) are notably less plausible than ones with visual artifacts (i.e., salt & pepper noise and Gaussian noise).

- **Elastic Transformation.** The image is deformed by displacing a grid of control points. Each point is shifted randomly in both the horizontal and vertical directions, typically following a Gaussian distribution to determine the displacement magnitude. The degrees of the distortion are regulated by adjusting the standard deviation of the Gaussian filter $\alpha \in [0, 4, 5, 6]$.

3.5.3 Sensitivity Test

Despite the presence of noise, a human designer can still recognize the underlying structure in a design. However, designs with missing parts or structural errors are less usable. Thus, we design the sensitivity test with the anticipation that an appropriate metric for the design generation evaluation task should progressively demonstrate deteriorating scores from visual artifacts to structural deficiencies. Additionally, to prove the importance of structural integrity in the evaluation process, we expect that the score for a mixed disturbance of Gaussian noise and patch swap will be comparable to that of solely patch swap disturbance, thus remaining independent from the added visual artifacts. The aim of the sensitivity test is to cross-compare the metric performance in dealing with various disturbances and to see if the metric aligns with human designers.

This test involves four datasets: BIKED [137], ImageNet [38], FFHQ [72] and Seeing3DChairs [7]. For each dataset, we shuffle and split the samples into $n = 10$

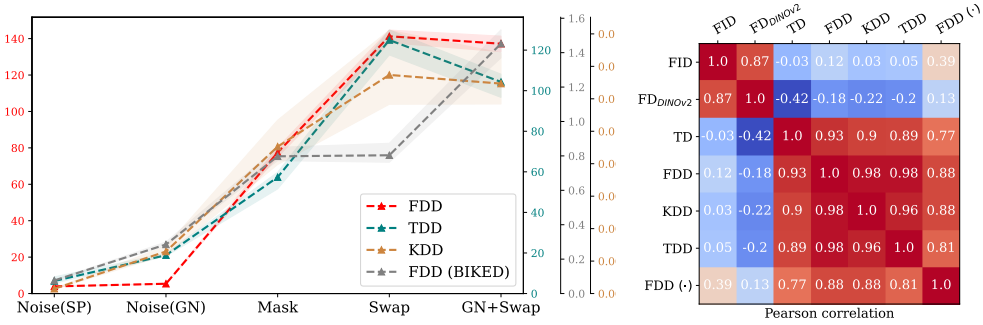


(a) On BIKED [137] **(b)** On Seeing3DChairs [7]

(c) On FFHQ [72] **(d)** On ImageNet [38]

Figure 3.4: Sensitivity Comparison. The y-axis represents the score value measured by each metric, where a lower value indicates a higher similarity to source data, i.e., better quality. A reliable plausibility metric should penalize more on the basis of structural errors (e.g., mask and swap) than visual artifacts (e.g., noise). For each metric, the *dashed line* shows the mean across the groups and the *shaded region* depicts the measured values from the groups.

groups, number of samples in each group varies from the dataset: $K = 300$ (for BIKED) and $K = 100$ (for Seeing3DChairs, FFHQ and ImageNet). We introduce five types of disturbances into source images and create five corrupted counterparts, i.e., pepper noise, Gaussian noise, patch mask, patch swap and a mix of Gaussian noise and patch swap. The introduced disturbances adhere to a rule where visual artifacts, such as pepper noise and Gaussian noise, are intentionally kept at levels that do not significantly impact the recognition of the design. On the other hand, structural failures, such as patch masking and patch swapping, lead to designs that are implausible and consequently receive worse human evaluation scores compared to visual artifacts. We choose one level from each disturbance described in Section 3.5.2: $\alpha = 0.01$ (pepper noise, Gaussian noise), and $\alpha = 0.25$ (patch mask, patch swap). Next, we measure the distance between each one of these corrupted image sets and the original image set, using FID, $\text{FD}_{\text{DINO-V2}}$, TD, and our FDD. Since they are measures of distance quantifying the dissimilarity between observed images and source images, a smaller value indicates greater similarity to the source data.



(a) Sensitivity test on BIKED [137] **(b)** Correlation test

Figure 3.5: Experiments with FDD and other DAE-based metrics. In (a), within all DAE-based metrics, FDD shows the best performance; (b) Pearson correlation of metrics over all distances measured during the sensitivity test with BIKED.

We plot several examples of disturbed images in Figure 3.3 and record the measured results in Figure 3.4. As expected, FID [58] and $\text{FD}_{\text{DINO-V2}}$ [161] show a great bias towards visual artifacts, with notably higher distance assigned to pepper and Gaussian noised images compared to those with patch mask and patch swap. In contrast to FID and $\text{FD}_{\text{DINO-V2}}$, TD and our FDD provide a distinct evaluation perspective by detecting structural faults and imposing penalties accordingly. One unanticipated result was that TD exhibits a poor performance with regard to pepper noise as illustrated in Figure 3.4b and Figure 3.4c. Furthermore, as the sample size decreases within each group from 300 (BIKED [137]) to 100 (for Seeing3DChairs [7] and FFHQ [72]), TD shows a significant increase in standard deviation across 10-times implementations. Interestingly, our FDD shows a better stability among various noises and gives significantly worse scores to images with structural failures.

Furthermore, we explore other possible metrics based on DAE by incorporating concepts from existing works such as KID [16], TD [62] and training the network on structural images [46]. This adaption yields new evaluation metrics, i.e., kernel denoised distance (KDD), topology denoised distance (TDD), and FDD (BIKED), respectively. Later on, we subject these metrics to the sensitivity test and present the results in Figure 3.5a. Our analysis reveals that FDD exhibits the most consistent performance across various criteria: the most stable result across different groups and excellence in distinguishing between visual and structural disturbances.

It is important to note that the different plausibility metrics presented in Figure 3.5a operate on inherently different scales due to the distinct formulations and normalization schemes of each method. As a result, the absolute numbers are not di-

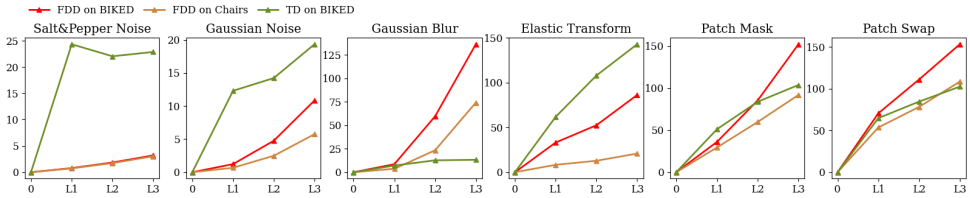


Figure 3.6: Metric comparison with increasing disturbances. The y-axis label represents the measured distance of each metric. The TD, as the most competitive metric to our FDD, performs however unstably with increasing disturbances.

rectly comparable across metrics. Despite this, the evaluation and comparison remain meaningful because the analysis focuses on relative trends and sensitivity patterns within each metric. In other words, the ability of a metric to distinguish between different types of disturbances, detect structural faults, and maintain consistency across sample groups is what informs its reliability, rather than the raw numerical range. Therefore, observing how each metric responds to noise, masking, or structural perturbations provides actionable insight, and differences in scale do not undermine the validity of the comparison.

Finally, we calculate the Pearson correlation coefficients pair-wisely among all candidate metrics, by taking the measured values from Figure 3.4a, and record the outcome in the table Figure 3.5b. Notably, the result reveals two categories among the metrics: FID and $\text{FD}_{\text{DINO-V2}}$ are grouped together, while TD and our designed metrics demonstrate a stronger correlation with each other. This is a promising finding since TD's main perspective is the topology and geometric behavior of the latent space, hereby we argue that the latent space of our DAE maintains the topological properties of the image space well and can be captured by Fréchet distance. Note that the KDD, TDD and FDD (BIKED) are experimental explorations. They are highly related to our FDD in the correlation test and our FDD outperforms them in the sensitivity test.

3.5.4 Consistency with Increasing Disturbances

In this section, we test the consistency of the FDD metric in response to escalating levels of disturbances outlined in Section 3.5.2. As a fundamental requirement, a performance metric should be able to accurately detect and respond to worsened image quality, including visual fidelity and structural plausibility. We start by adding various disturbances to a subset comprising $K = 1\,000$ images sourced from the BIKED [137] and Seeing3DChairs [7] datasets, respectively. Afterwards, we report the scores in Fig-

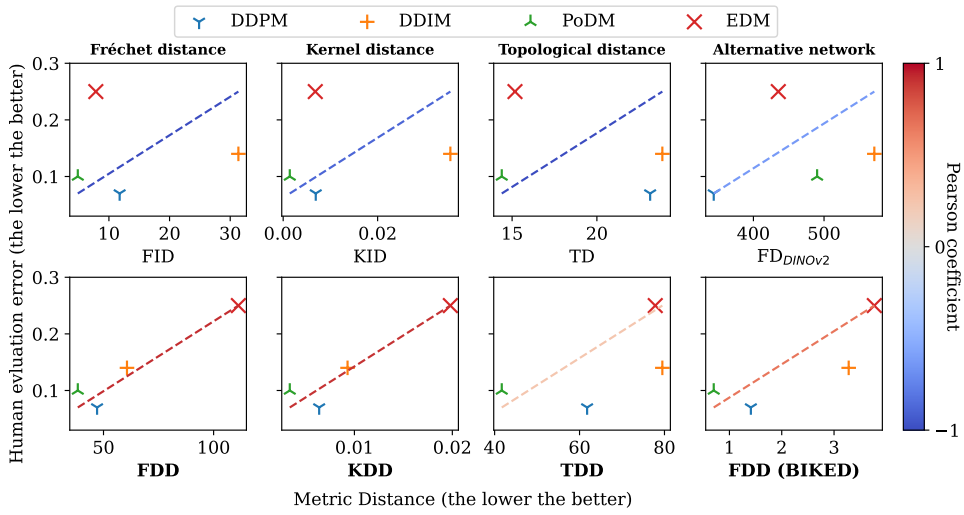


Figure 3.7: Metrics comparison in the task of model ranking. We color-code the *diagonal lines* after the measured Pearson correlation coefficient between metric results and human judgments, *dark red* refers to a strong positive correlation between metric distances and human judgments.

ure 3.6 and demonstrate the consistent performance of the proposed FDD metric. While FID has been noted to exhibit inconsistency in detecting the disturbance level induced by salt and pepper as documented in Heusel et al. [58], our FDD successfully measures the levels of various deformations, spanning from visual to structural distortions.

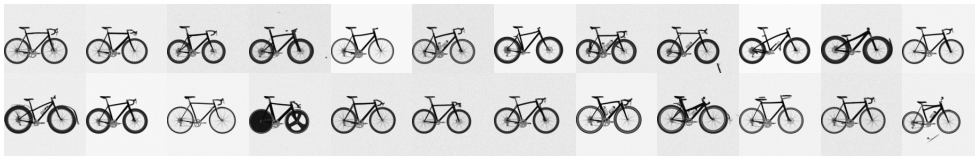
3.5.5 Model Ranking

In the model ranking, we employ five deep generative models, e.g., DDPM [61], DDIM [156], EDM [71] and PoDM [167], with the consideration that the models executed in model ranking should exhibit significant differences in visual quality and structural plausibility. These models are then trained on BIKED images with a resolution of 256×256. We generate 5k images from each model and manually evaluate them into plausible designs and implausible designs. We denote the ratio of implausible bicycle designs as human evaluation error, the lower the better, which serves as the "ground truth" in this model ranking experiment.

Meanwhile, we apply the candidate metrics, including FID, KID, $\text{FD}_{\text{DINO-V2}}$, TD, FDD, and other DAE-based metrics, to evaluate each generative model with their generated samples, with 1k images in each group. Subsequently, the distances measured



(a) DDPM [61] (FID: 11.77, $FD_{DINO\text{-}V2}$: 342.82, FDD: 48.08)



(b) DDIM [156] (FID: 31.35, $FD_{DINO\text{-}V2}$: 571.21, FDD: 60.66)

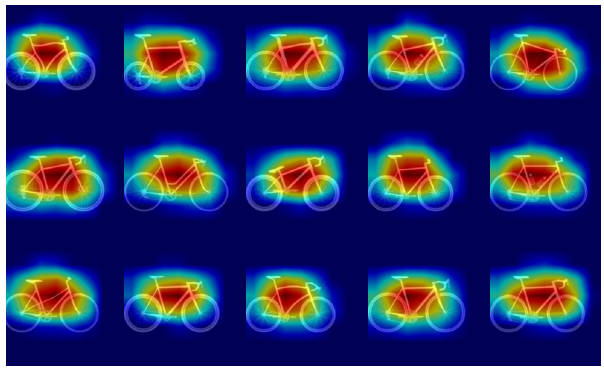


(c) EDM [71] (FID: 7.84, $FD_{DINO\text{-}V2}$: 435.25, FDD: 111.25)

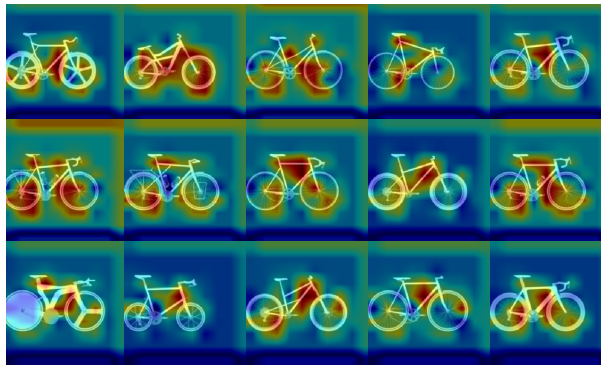
Figure 3.8: Qualitative evaluation of generated bicycle designs. DDPM and DDIM yield structurally more plausible results than EDM, but FID and $FD_{DINO\text{-}V2}$ fail to agree with human judgments, whereas our FDD ranks the models with the perspective of structural plausibility and penalizes the visual artifacts as well.

and human error rates are visualized in Figure 3.7. Note that the proximity of the plotted points (measured distances, human evaluation error) to the diagonal line signifies the consistency of the metric with human evaluation. Particularly, the expected behavior is seen in the FDD, KDD, and FDD (BIKED) measurements, which are highly associated and yield the same consistent ranking. This observation aligns with the notion proposed by [161], whereby provided a good encoder is chosen, all these metrics provide sensible ways of quantifying distances between probability distributions.

On the other hand, the absence of a significant link between the SOTA metrics and human evaluation suggests a deficiency of these most reported metrics in evaluating structural design images. In Figure 3.8, we plot the generated bicycles for qualitative evaluation of our FDD metric. EDM achieves the best FID of 7.84, but the generated bicycles contain a large portion of implausible designs; DDPM (FID 11.77) and DDIM (FID 31.35) are unfairly penalized, even though the results are significantly more plausible than those from EDM. Here, our FDD is able to rank the models more accurately.



(a) FID (Inception-V3)



(b) Our FDD (DAE trained on ImageNet)

Figure 3.9: Where does the metric look at? Heatmaps illustrating the perception of the Fréchet distance with Grad-CAM. The focus of an encoder can be demonstrated by both *bright red* and *deep blue*. The offset of the focusing area can be caused by upsampling the attention map to the image shape.

3.5.6 Grad-CAM Visualization

The Grad-CAM [148, 91] is designed to visualize the focus on the input image as perceived by the classifier/segmentation model up to the last fully connected layer. In our work, we use the Grad-CAM visualization to compare the observation fields of the FID and FDD metrics. We first transfer the test images into inception space and latent space via the Inception-V3 model and the DAE model, respectively, which have the same dimension of $\mathbb{R}^{2048}$. We compute the mean $\mu_{\boldsymbol{w}}$ and the covariance $\Sigma_{\boldsymbol{w}}$ of the extracted features. Then, we obtain the attention maps of FID and FDD by back-propagating the value of ${\mu_{\boldsymbol{w}}}^2 + \Sigma_{\boldsymbol{w}}$, to the last convolutional layer of the Inception-V3 model (i.e., *Mixed 7c.branch pool*) and the one of DAE (i.e., *encoder 8*), respectively.

The Grad-CAM generates a heatmap of reduced dimensions (e.g., 10×10 for DAE) which is then upsampled to match the dimensions of the original image for intuitive visual comparison. The heatmaps (seen in Figure 3.9) visualize the area observed by the corresponding metric in the BIKED [137] images, i.e., where the metric "looks at" when it calculates the distance.

The focus of the Inception-V3 model is simply the area around the center of the main object, often mismatching the object's shape and borders. As explained in previous works [161, 83], this phenomenon is caused by the model's classification training across 1 000 classes. Consequently, it prioritizes detecting the object's presence rather than its structure. On the other hand, even when also trained on ImageNet, the DAE generates an intensive attention map with positive and negative gradients surrounding the bicycle's structure, which efficiently assesses the complex details of the bicycle's shape. In Figure 3.14 and Figure 3.15, we provide also Grad-CAM analysis on general images, where Inception-V3 model tends to drop structural information while DAE captures it.

3.5.7 Reconstruction with Denoising Autoencoder

In this section, we demonstrate the restoration power of the DAE model trained on the ImageNet on noisy images from various datasets, e.g., the ImageNet [38] in Figure 3.10, BIKED [137] in Figure 3.11, Seeing3DChairs [7] in Figure 3.12, and FFHQ [72] in Figure 3.13. For the reconstruction, we apply Gaussian noise to the original images using the formula $\mathbf{x}_\eta = \mathbf{x} + \eta$, where $\eta \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$ and $\sigma^2 = 0.5$ and then restore the noised images with the DAE model. First, the DAE reliably filters out the additive Gaussian noise, which shows that the learned latent representation encodes the essential structural content of the image (rather than simply reproducing pixel-level noise). Second, because it is trained to ignore superficial perturbations (noise) and focus on the underlying image content, the encoder part of the DAE produces representations that are robust to visual artifacts.

Here, FID, based on the Inception-V3 network pretrained on ImageNet, lacks the explicit restoration objective that encourages latent representations to focus on structural coherence and clean image manifolds. FID therefore uses features optimized for classification (or generic image-recognition) rather than for denoising or explicitly modeling a clean-image manifold. As a result, FID's feature space may be overly sensitive to visual noise, artifacts, or superficial texture differences, but less sensitive to deeper structural implausibilities of generated designs (for instance, incorrect shape



Figure 3.10: DAE reconstruction of images from ImageNet. *Top*: original images, *Middle*: noised images, *Bottom*: reconstructed images.

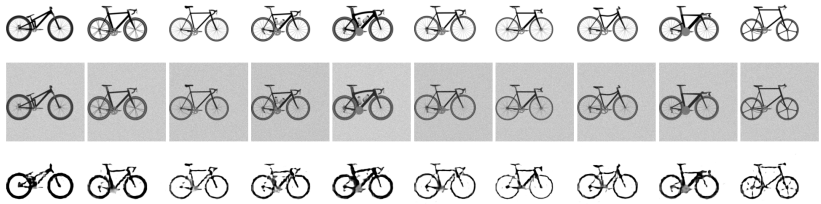


Figure 3.11: DAE reconstruction of images from BIKED. *Top*: original images, *Middle*: noised images, *Bottom*: reconstructed images.

combinations or missing functional components).

3.6 Conclusion

In this work, we approached the field of evaluating generated design images and proposed a structure-biased metric Fréchet denoised distance (FDD) by replacing the Inception-V3 model in the FID metric with a denoising autoencoder, unsupervised-trained on the same dataset (i.e., ImageNet) and with the same 2048-dimensional latent space. Through a series of experiments, including sensitivity test for various types of disturbance, consistency test with increasing disturbances, and alignment test with human judgment in model ranking, we found FDD to fulfill the quality requirements for serving as a metric and outperform other SOTA metrics, e.g., FID, $\mathrm{FD}_{\text{DINO-V2}}$ and TD, on design images such as BIKED and Seeing3DChairs, as well as real-world images such as human faces from FFHQ and general images from ImageNet. We explained the effectiveness of FDD with a Grad-CAM visualization, where the DAE is able to "focus" on the design structure of the observed shape.

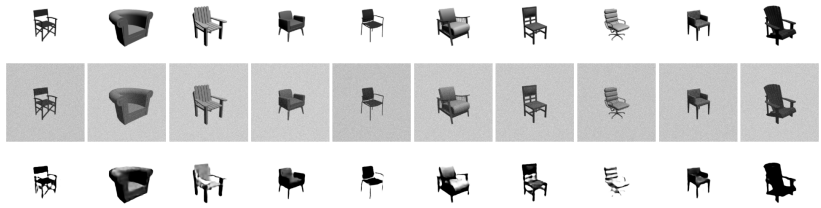


Figure 3.12: DAE reconstruction of images from Seeing3DChairs. *Top*: original images, *Middle*: noised images, *Bottom*: reconstructed images.

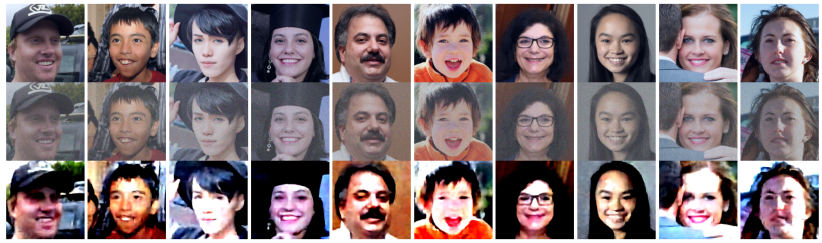


Figure 3.13: DAE reconstruction of images from FFHQ. *Top*: original images, *Middle*: noised images, *Bottom*: reconstructed images.

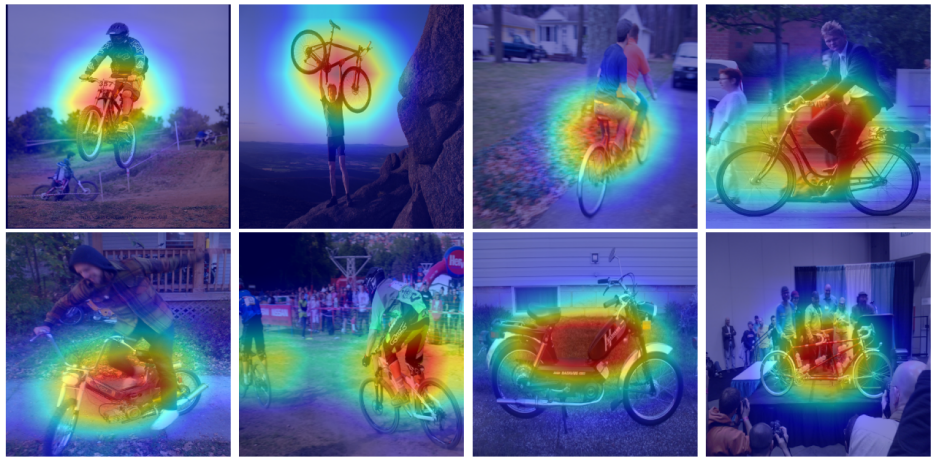


Figure 3.14: Heatmaps of the Inception-V3 model on ImageNet images from the bike class



Figure 3.15: Heatmaps of our DAE model on ImageNet images from the bike class. Inception-V3 focuses on the object from the top-classes, such as the bike, and hereby ignores the rest parts of the image, which is suboptimal for evaluating the image plausibility.

Chapter 4

Shape Generation with Learning-free Decomposition

Different from previous chapters, from this chapter on we start targeting on generation of 3D models. The AI community has made substantial progress in learning mesh data, however, introducing current mesh-based DGMs into industrial design processes is not trivial, because industrial meshes tend to be extremely high-dimensional. To address this, we present a new pipeline that enables learning on meshes by decomposing them into low-dimensional variables, without explicit training, answering the research question 3: *How to use DGMs to generate high-dimension designs more efficiently?* The content of this chapter has been published in the paper [168].



Figure 4.1: A gallery of 3D meshes generated by SpoDify.

4.1 Introduction

Today, 3D modeling has become a more convenient method in the industrial design process with the help of powerful CAD software, whereas traditional 2D blueprints have less advantages over 3D shapes, e.g., B-Reps and meshes, in terms of simulating usability, efficiency of design modification, and accuracy of representation. While B-Rep data remains a challenge in learning and generating, meshes [96, 154] are by far the most commonly used form in industry, as the native representation of many CAE software and finite element tools [138]. Besides, generating 3D shapes represents one of the major challenges of the deep generative modeling community, where the researchers have made substantial progress in directly learning on mesh data [135, 30, 199]. However, as a non-monotonous data representation, a mesh contains multiple modal data forms and their lengths vary with samples, where deep learning methods generally perform poorly. Most recently, research [32, 151, 64] in this field enabled the generation of meshes with signed distance field (SDF) [162], a powerful implicit representation that encodes the source mesh into a voxel, where the value of each voxel grid indicates a distance value from the grid position to the nearest surface of the source mesh. Here, a negative value indicates that the point is inside the shape, while a positive value indicates that the point is outside the shape. Representing a mesh in the voxel form allows the implementation of 3D convolutional neural networks (CNNs), which addresses the challenge of using deep learning on meshes. However, to produce high-fidelity shapes, a large dimension of the voxel-shaped SDF is often required, e.g., 256^3 [64, 123], which poses computational and temporal challenges for directly learning with deep generative models (DGMs).

Meanwhile, the trend in high-dimensional data generation has shifted toward encoding the source data into a low-dimensional latent space so that DGMs can efficiently learn from compact latent codes. Using a trained autoencoder is the most commonly used methodology. It has yielded powerful DGMs, e.g., latent diffusion models (LDMs) [139, 194], which can generate high-dimension data with much reduced computational cost. Encoding high-dimensional data has also been explored in the context of SDF representations [105, 90, 29]. However, the quality of the results generated by such pipeline largely depends on the performance of the autoencoder introduced, which remains challenging and computationally intensive to train. In fact, the amount of training samples needed to properly train an autoencoder drastically increases with the dimensionality and diversity of the target data, which tends to be impossible for real-world design cases.

Several existing approaches have used a learning-free encoding pipeline to obtain the latent variables of high-dimensional data [68, 22, 64]. Among them, neural wavelet-domain diffusion [64] (referred to as NWD in this chapter) achieves state-of-the-art (SOTA) performance in generating complex topology and structures with clean surfaces and fine details. However, the introduced diffusion model has to be trained on 3D voxels of dimension 130^3 for a single-level wavelet transformation. To fill the gap between mesh generation and neural latent learning, a fully deterministic approach can be used, such as singular value decomposition (SVD), which has been historically used for dimensionality reduction in deep learning tasks including data classification [74, 190] and image generation [73]. [73] leverages SVD eigenvalues as a loss regularization term for GANs training. Furthermore, SVD guarantees minimum information loss during the encoding, which is an essential characteristic for accurately reconstructing complex models. Even though SVD guarantees minimum information loss without the need for training, its application in generative 3d modeling remains understudied.

In this chapter, we propose to exploit this idea and design a novel mesh generation method, spectral-domain diffusion for high-quality shape generation (SpoDify), which uses a learning-free pipeline to encode meshes into low-dimensional spectral features that serve subsequently as the latent variables for training the diffusion-based DGM. We display our results in Figure 4.1, which are generated by learning on the ShapeNet dataset [26]. Compared to SOTA methods (3DShape2VecSe [195], NWD [64]) that rely on deep learning-based encoders or large data representations, our SpoDify can produce comparable results, and in some cases even superior, by using generative modeling in a 512-dimensional spectral space.

4.2 Method

Our method SpoDify is inspired by NWD [64], where we additionally introduce an SVD-based decomposition approach to achieve a more interpretable and computationally efficient encoding of mesh representations. We show a diagram of SpoDify in Figure 4.2.

4.2.1 Spectral Representation of Mesh

Clustering Our approach remains effective even with a limited number of training samples, provided the samples are representative of the larger dataset. We explain

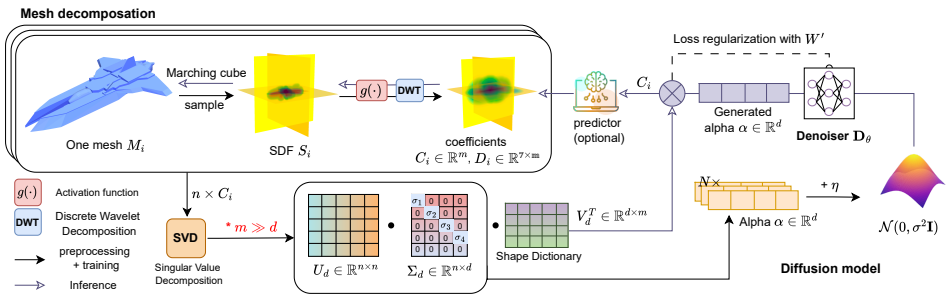


Figure 4.2: Diagram of SpoDify. We apply singular value decomposition on a set of the coefficients that are derived by applying a signed distance field and discrete wavelet transformation on source meshes, resulting in the dataset of spectral features. Here, the basis $V_d^\top$ will be stored for later generation; spectral features α will serve as one sample and will be used for training the diffusion model. To generate a new mesh, the trained diffusion model generates new α for a given random noise. The generated α will be denormalized and then multiplied with pre-computed and stored $V^\top$ to obtain new low-frequency coefficients C_i, which can be converted to new mesh M_i.

this in Section 4.2.3. We construct a diverse training set using fewer samples by introducing a clustering process. First, diffusion maps [35] are applied to the complete set of available meshes, using the Chamfer distance as the metric, embedding all meshes into a shared diffusion space that preserves geometric relationships. K-Means clustering is then used to partition the dataset into n clusters, capturing its diversity. From each cluster, one representative mesh is selected, ensuring broad representation even with a small subset. Pairwise Chamfer distances between all 3D meshes quantify shape similarity. The Chamfer distance is implemented using the GitHub repository [178]. The resulting distance matrix, capturing geometric similarities between shapes, is transformed into a similarity kernel matrix using an exponential function to enhance mesh relationships representation. Diffusion maps [35] are applied to reduce high-dimensional representation while preserving intrinsic geometric structure, extracting the top 64 eigenpairs for low-dimensional embedding of each mesh. This embedding captures significant modes of variation in the dataset. Clustering on the diffusion embeddings using K-Means identifies n cluster centroids as representative shapes encapsulating the dataset's diversity.

Implicit representation with SDF Next, we represent the geometry of the mesh with the SDF due to its differentiability and smoothness properties [45]. We sample from the n representative meshes $M_{1,..,n}$. Each mesh M_i is scaled to the range

$[-0.5, 0.5]^3$ to standardize its size and position and then represented as an SDF S_i of resolution 256^3, so that

$$f_{M_i}(x) = \begin{cases} d(x, \partial M_i) & x \in M_i, \\ -d(x, \partial M_i) & x \notin M_i, \end{cases} \tag{4.1}$$

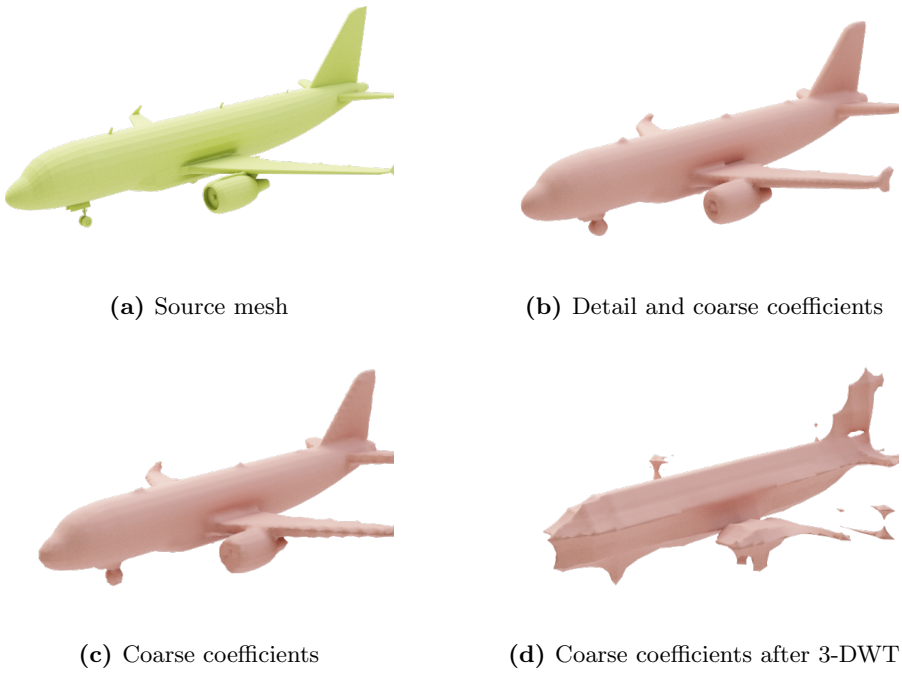
with d being a suitable point-surface distance. When points in the field are far away from the shape surface, their value becomes large and unstable (i.e., with increasing deviation). These points are often identified as irrelevant by the model during training and contribute the least to the prediction of the final shape. To maintain smoothness and avoid discontinuities, [64] truncates the distance values in the SDF to the range $[-0.1, 0.1]$. While this improves the learning, it does not emphasize accurate predictions along the shapes' contours. Unlike them, we introduce a limiting function

$$g(x) = \frac{1}{2} \tanh\left(f_{M_i}(x)\right) - \frac{1}{2}. \tag{4.2}$$

Through this bounding continuous function, SDF values far from the object's surface are constrained to approach zero. Since wavelets, applied in the following step, are inherently sensitive to local variations, this ensures that the resulting coefficients are more responsive to the shape boundaries rather than distant regions. As a result, with distant regions approaching zero, fewer wavelet coefficients are required to encode the surface, leading to a more efficient shape representation.

Pre-encoding with wavelet transformation Furthermore, we apply the 3D discrete wavelet transformation (DWT) on the preprocessed SDFs to extract localized features and patterns from the data. This step is essential for efficiently encoding localized spatial details while reducing redundancy in the representation. DWT can be considered as a particular type of convolutional layer with specific filter banks for extracting multi-scale features [120, 52]. Here, selecting an appropriate wavelet filter is crucial. While Haar wavelet is a popular choice for its simplicity, using it to encode smooth and continuous signals such as the SDF may introduce some voxelization artifacts [64]. For the data representation chosen in this approach, the `Coiflet` wavelet [15] is a suitable choice because, empirically, it provides a good balance between performance (in preserving important geometric features) and reconstruction accuracy. The application of the 3D DWT on each S_i results in two sets of coefficients, i.e., one low-frequency coarse coefficient $C_i \in \mathbb{R}^{130^3}$ (referred to as DWT coefficients in this chapter) and high-frequency detail coefficients $D_i \in \mathbb{R}^{7\times 130^3}$.

For the subsequent process, we drop the detail coefficients, as a single-level wavelet decomposition retains sufficient information in the coarse coefficients for reconstruction. The difference between Figure 4.3b and Figure 4.3c demonstrates this effect. However, training a generative model directly on these coefficients ($C_i \in \mathbb{R}^{130^3}$) remains computationally demanding. To mitigate this, [64] applied hierarchical wavelet transformation for further compression. In such cases, discarding detail coefficients is no longer viable, shown in Figure 4.3d, as they must be further predicted from the coarse coefficients.



(a) Source mesh

(b) Detail and coarse coefficients

(c) Coarse coefficients

(d) Coarse coefficients after 3-DWT

Figure 4.3: Effect of Wavelet Decomposition levels and the Dropping of High-Frequency Coefficients on Plane Mesh Reconstruction. (a) Original plane mesh; (b) Reconstructed plane after applying wavelet decomposition and reconstruction using all coefficients (both coarse coefficients and fine coefficients); (c) Reconstruction after one single-level wavelet decomposition level, keeping only low-frequency coefficients (coarse coefficients) and setting others to zero; (d) Reconstruction after **three** levels of wavelet decomposition, keeping only low-frequency coefficients (coarse coefficients) and setting others to zero.

Dataset decomposition with SVD We propose to encode the DWT coefficients with SVD, which can be conducted through the following steps: (1) for n meshes, flatten their DWT coefficients, denoted by $C_i \in \mathbb{R}^m, m = 130^3, i = 1, \ldots, n$, (2) stack

the coefficients, resulting in a matrix $X = [C_1, C_2, ..., C_n] \in \mathbb{R}^{n \times m}$, and (3) perform singular value decomposition (SVD) on X. Let $r \leq \min(m, n)$ denote the rank of X, the compact SVD is $X = U\Sigma V^\top$, where $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{m \times r}$ are semi-unitary matrices and $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ contains the singular values on it diagonal. We arrange the singular values in descending order, i.e., $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, for later truncation. Here, we address:

1. For the geometric information of each mesh (stored in rows of X), the rows of U compress it drastically when $m \gg n \geq r$, e.g., high-resolution meshes are used on a small 3D shape dataset with sample number n.

2. SVD can be used to obtain a low-dimensional rank approximation of X by keeping only the $d < r$ largest singular values, where the approximation error depends on the spectrum of X. Let $\widehat{X}_d = U_d \Sigma_d V_d^\top$, where $\Sigma_d = \mathrm{diag}(\sigma_1, \ldots, \sigma_d)$ and $U_d \in \mathbb{R}^{n \times d}, V_d \in \mathbb{R}^{m \times d}$ obtained from U and V by only keeping the first d columns, respectively. We have the approximation error: $\|X - \widehat{X}_d\|_{\mathrm{F}}^2 = \sum_{i=d+1}^{r} \sigma_i^2$. SVD achieves the optimal approximation error by the Eckart–Young–Mirsky theorem [103]. If the spectrum of X decays rapidly, then we can safely truncate off a large fraction of singular values and keep the error small.

In practice, we maximize computational efficiency by decreasing the approximation rank d to the lowest value, where the reconstructed mesh shows no visually recognizable error, and measured infinity error should be acceptable. See Figure 4.5 for an example of selecting d for the airplane dataset in ShapeNet [26]. Intuitively, each row of U_d is the low-rank representation of a mesh shape, and its corresponding singular value reflects its frequency in the entire data set. Thereby, we decide to define the *spectral features* of the mesh shape by scaling each row of U_d with its singular value, i.e., rows of matrix $U_d\Sigma_d$. We shall denote by α a row of $U_d\Sigma_d$. Also, the column space of V_d is a subspace of the original wavelet coefficients, serving as a "dictionary" or "basis" for representing the DWT data. Thus, we define V_d as *DWT basis* in this chapter. In this setup, each (flattened) DWT coefficient is approximated by

$$\widehat{C}_i = \alpha V_d^\top,$$

where $\widehat{C}_i = C_i$ iff. $d = r$. In the sequel, we shall train a generative model on the spectral feature α, and a new shape can be created by sampling a new α from the model and reconstructing the DWT coefficients with the matrix V_d. Note that the space where α lies maintains the same smoothness of the SDF space, as only

continuous functions are applied to those.

4.2.2 Spectral Domain Diffusion

Diffusion model architecture For the diffusion model, we adapt the denoising diffusion probabilistic model (DDPM) architecture proposed by [61]. However, we replace the 2D convolutional layers with fully connected dense layers to better handle the 1D sequence nature of the spectral features α. This modification is motivated by the fact that the values in α are ordered according to the weights of the corresponding eigenvectors, and dense layers are better suited to capture global patterns in such structured data. For training and inference, we utilize a score-matching generative model, specifically the elucidating diffusion model (EDM) [71, 167], due to its fast sampling and efficient training capabilities. The diffusion model is trained to predict the spectral features α from noisy inputs, enabling the generation of new α values that can be used to reconstruct novel meshes.

Training objective The spectral features α are normalized to the scale $[-3, 3]$ and used to train the diffusion model. The diffusion model is trained using a composite loss function designed to ensure accurate prediction of the spectral features α while preserving the structural integrity of the generated shapes. The loss function is defined as

$$L = (1 - \lambda)L_\alpha + \lambda L_C, \tag{4.3}$$

where L_α ensures the model accurately predicts the spectral features α, and L_C acts as a regularization term to incorporate the precomputed DWT basis $V^\top$. The individual loss terms are defined as

$$L_\alpha = \mathbb{E}_{\sigma,\alpha,\eta} \| D_\theta(\alpha + \eta; \sigma) - \alpha \|_2^2, \tag{4.4}$$

$$L_C = \mathbb{E}_{\sigma,\alpha,\eta} \| D_\theta((\alpha + \eta) \cdot V^\top; \sigma) - \alpha \cdot V^\top \|_2^2, \tag{4.5}$$

where, $\ln(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$, $\eta \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$, D_θ is the implemented neural denoiser and $V^\top$ is the DWT basis obtained from SVD. While $V^\top$ does not participate directly in the training process, it serves as a critical multiplication factor for shape reconstruction. The regularization term L_C ensures that the generated α values, when combined with $V^\top$, produce coherent and structurally valid low-frequency coarse coefficients, as demonstrated in Section 4.3.3.

4.2.3 Mesh Generation

Our proposed pipeline yields a set of basis $V^\top$ for storing shape elements, and features α that serve as "weights" that can be combined with the basis and form new shapes. Thus, at the beginning of our pipeline, we introduce clustering to maximize the shape elements obtained in the basis $V^\top$, ensuring that when the model generates new spectral features, they can be leveraged to explore the shape space of the larger dataset.

During generation, the trained diffusion model produces new α values from random noisy inputs. These generated α values are denormalized with pre-stored scaling parameters ($\alpha_{min} \in \mathbb{R}^d$ and $\alpha_{max} \in \mathbb{R}^d$ estimated from source α among all training samples) and multiplied with the pre-stored $V^\top$ to reconstruct the low-frequency coarse coefficients C_i. Finally, the inverse DWT and marching cube algorithms are applied to C_i to generate a new mesh M_i.

4.3 Experiments

4.3.1 Experimental Dataset and Setup

Evaluation is conducted on ShapeNet [26] to compare with previous SOTA models. We focus on the airplane and chair categories from ShapeNet. These categories are chosen due to their geometric complexity and relevance in benchmarking generative models for 3D surface reconstruction. For mesh decomposition, we consistently select a sample size of $n = 1k$ for each dataset and set the reduced dimensionality to $d = 512$ as the default configuration. An experiment on tuning d is presented in Section 4.3.3. Training is conducted on a single NVIDIA A10G GPU with a batch size of 32 and a learning rate of 5×10^{-4} for $100k$ steps. Using the EDM training pipeline [71], we retain the original hyperparameters: $P_{\text{mean}} = -1.2$ and $P_{\text{std}} = 1.2$. For inference, we set $\sigma_{\text{min}} = 0.002$, $\sigma_{\text{max}} = 80$, $\rho = 5$, and $T = 64$.

4.3.2 Evaluation Metrics

Evaluating the unconditional synthesis of 3D shapes is a different challenge different than the one we introduced in Chapter 3 due to the absence of direct ground truth correspondence. To address this, we use established metrics consistent with previous works which include:

Chamfer Distance (CD) : This metric measures the similarity between two point clouds. It computes the average distance between each point in one point cloud and its closest point in the other point cloud. The Chamfer Distance for two point clouds P and Q, respectively, is defined as:

$$\mathrm{CD}(P, Q) = \frac{1}{|P|} \sum_{p \in P} \min_{q \in Q} \|p - q\|^2 + \frac{1}{|Q|} \sum_{q \in Q} \min_{p \in P} \|q - p\|^2, \tag{4.6}$$

where p and q are points in the point clouds P and Q, respectively, and $\|\cdot\|$ denotes the Euclidean distance between two points.

Minimum Matching Distance (MMD) : This metric measures the mean Chamfer Distance between a sample in the test dataset and its closest sample in the generated dataset. Lower values indicate better performance. The MMD is given by:

$$\mathrm{MMD}(P, Q) = \frac{1}{|P|} \sum_{p \in P} \min_{q \in Q} \mathrm{CD}(p, q), \tag{4.7}$$

where P and Q represent the point clouds in the test and generated datasets, respectively.

Coverage (COV) : Coverage measures the percentage of test data that has at least one corresponding match in the generated data. After assigning every generated sample to its closest test data based on Chamfer Distance, the Coverage is computed as:

$$\mathrm{COV}(P, Q) = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{I} \left(\min_{p \in P} \mathrm{CD}(q, p) \leq \epsilon \right), \tag{4.8}$$

where $\mathbb{I}$ is the indicator function, which is 1 if the condition holds, and 0 otherwise. ϵ is a predefined threshold for matching.

1-Nearest-Neighbor Accuracy (1-NNA) : This metric computes the accuracy of the nearest neighbor search by measuring the percentage of generated point clouds that match the nearest ground truth point cloud. Ideally, the accuracy should be around 50%, indicating that the generated data is similar to the test data. The accuracy is computed as:

$$\text{1-NNA}(P, Q) = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{I} \left(\min_{p \in P} \mathrm{CD}(q, p) \leq \min_{q' \in Q} \mathrm{CD}(q', p) \right). \tag{4.9}$$

Jensen-Shannon Divergence (JSD) : JSD measures the divergence between the probability distributions of two datasets. In our context, we convert the point clouds into discrete voxel grids and compute the divergence between the test and generated data distributions. The JSD between two distributions P and Q is given by:

$$\mathrm{JSD}(P, Q) = \frac{1}{2} \left(D_{\mathrm{KL}}(P \| M) + D_{\mathrm{KL}}(Q \| M) \right), \tag{4.10}$$

where $M = \frac{1}{2}(P + Q)$, and $D_{\mathrm{KL}}(P \| Q)$ is the Kullback-Leibler divergence between distributions P and Q, defined as:

$$D_{\mathrm{KL}}(P \| Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}. \tag{4.11}$$

These metrics ensure a comprehensive evaluation of the generated meshes in both the 3D space and visual quality, addressing various aspects of geometric accuracy, distributional similarity, and perceptual quality.

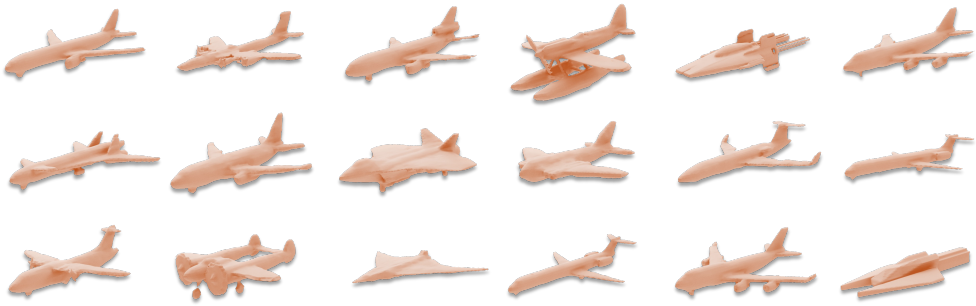
4.3.3 Ablation Study

Dimension of the spectral space Reducing the dimensionality of the spectral space d directly contributes to reducing data size, model size, and computational costs at the expense of reconstruction accuracy. To investigate this trade-off, we conducted a hyperparameter tuning experiment, varying d across several values. The goal is to identify an optimal dimension that balances these competing factors while achieving strong overall performance. Based on the results, we select $d = 512$ as the most suitable configuration for our method. Figure 4.5 illustrates the effect of different truncation levels on various evaluation metrics, while Table 4.1 provides a quantitative comparison of performance across various dimensions ($d \leq n = 1\,000$). Metrics used for evaluation include minimum matching distance(MMD), Jensen-Shannon divergence(JSD), coverage(COV), and the L_2-norm reconstruction error.

At $d = 512$, our method demonstrates a balanced performance across metrics, achieving the best trade-off between reconstruction accuracy and generative diversity. Specifically, the L_2-norm reconstruction error improves significantly compared to $d = 256$, dropping from 1.54×10^{-6} to 5.82×10^{-7}. While increasing d to 786 or 1 000 further reduces reconstruction error, this comes at the cost of higher computational demands and a marginal decrease in generative diversity metrics such as JSD and COV. Dimension $d = 512$ strikes an effective balance, delivering strong coverage



(a) Examples of 3D chairs.



(b) Examples of 3D airplanes.

Figure 4.4: Examples of 3D meshes generated by our SpoDify for qualitative evaluation.

(64.71%) and reconstruction quality without unnecessarily increasing model size or computational overhead.

Training loss Several techniques have been introduced in our method, i.e., the loss regularization L_C (Equation (4.5)) and the limiting function $g(\cdot)$ (Equation (4.2)). Thus, we evaluate the impact of each feature through an ablation study. Here, we consider the following ablated models:

(i) Ablation (w/o L_C): Removing the loss regularization of wavelet coefficients L_C;

(ii) Ablation (w/o L_α): Removing the loss term of spectral feature L_α;

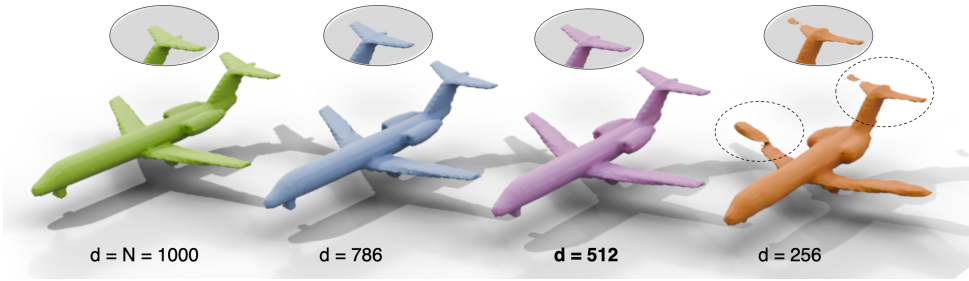


Figure 4.5: Truncation level. Changing the reduced length d of rows in α can impact the visual quality of final results and the computational power required to train the generative model. We notice that by truncating the row length until $d = 512$, no significant visual artifacts are brought to the reconstructed meshes, whereas with $d = 256$, reconstructed meshes show structural errors.

Table 4.1: Ablation study on the dimension of the spectral space out of $n = 1\,000$ possible.

d	MMD↓	JSD↓	COV(%) ↑	L2 Recons↓
256	1.68	3.28	50.06	1.54E-6
512	1.7	3.1	64.71	5.82E-7
786	1.81	3.05	69.55	1.49E-7
1 000	1.76	2.9	61.05	0

(iii) Ablation (w/o $g(\cdot), L_C$): Deactivating the limiting function on SDF $g(\cdot)$ and removing the loss regularization of wavelet coefficients L_C;

From the results presented in Table 4.2, we have the following observations: (1) The full model, which includes all three components (L_C, L_α, and $g(\cdot)$), achieves in average the best performance; (2) Removing the wavelet coefficient regularization (L_C) results in a slight increase in MMD and JSD, as well as a drop in COV, indicating that L_C helps in improving coverage and reduces discrepancy; (3) Removing the spectral-

Table 4.2: Ablation study on training and regularization losses with the airplane Dataset from ShapeNet. Configurations are formed by different combinations of parameters: $V^\top$ indicates the inclusion of regularization loss with respect to the wavelet domain, g denotes the use of a limiting function on SDF values, and α specifies the inclusion of loss with respect to the spectral space.

	$g(\cdot)$	L_C	L_α	MMD↓	JSD↓	COV(%) ↑
SpoDify	✓	✓	✓	1.7	**3.1**	**64.71**
Ablation (i)	✓		✓	1.776	3.27	61.05
Ablation (ii)	✓	✓		1.73	3.47	45.17
Ablation (iii)			✓	**1.64**	3.15	56.1

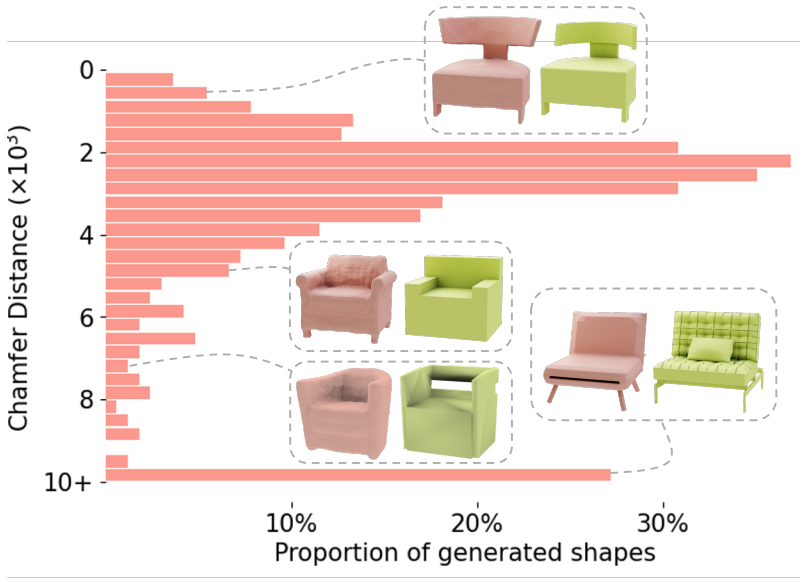


Figure 4.6: Shape novelty analysis on ShapeNet [26] chair category. We plot the distribution of 500 chair samples generated by our method and their closeness to the training dataset. Additionally, we display samples to visualize the similarity of various CD values, where *green chairs* are from the training dataset and *red chairs* are generated.

feature regularization loss (L_α) leads to a notable increase in JSD and a significant decrease in COV, confirming that L_α is key to maintaining the diversity and quality of the generated shapes; (4) When both $g(\cdot)$ and L_C are removed, the model performs better in terms of MMD compared to removing L_α alone, but it still lagged behind the full model in JSD and COV. This suggests that while the limiting function $g(\cdot)$ and the wavelet regularization term help in coverage, they do not fully compensate for the loss of spectral regularization.

In conclusion, our findings emphasize the importance of all three components in achieving the best performance. The wavelet coefficient regularization L_C and the limitation function $g(\cdot)$ contribute to model stability and coverage, while the spectral regularization loss L_α is essential for maintaining high-quality outputs and preventing overfitting. The full model, with all components, strikes the optimal balance between MMD, JSD, and COV, demonstrating the effectiveness of our design choices.

Table 4.3: Quantitative evaluation of our proposed pipeline and current 3D shape generators. Metrics are computed over ShapeNet classes airplane and chair using the Chamfer distance (CD). NWD provided generated meshes for evaluation, while UDIff did not release its ShapeNet meshes. Due to computational constraints, we did not retrain UDiFF to compute JSD, resulting in missing values.

Category	Method	1-NNA ↓	MMD $\times 10^{-3}$ ↓	COV (%) ↑	JSD ↓
Airplane	NWD [64]	97.24	1.69	71.2	2.9
	UDiFF [200]	**74.48**	**0.315**	**64.77**	-
	Ours	97.98	1.7	64.71	3.1
Chair	NWD [64]	53.4	1.180	45.19	0.027
	UDiFF [200]	65.96	1.167	52.58	-
	Ours	**46.69**	**1.114**	**53.5**	**0.022**

Table 4.4: Efficiency comparison. Our approach can use less than 90% GPU compared to TetraDiffusion [69] and be trained in a few hours compared to days needed for NWD [64] and UDiff [200]. For NWD and UDiFF, the (+12) indicates additional GPU memory used for training the detail predictor network, separate from the main network responsible for global coefficient training.

Method	**Representation**		**Training**			**Inference**
	Dimension	Compression rate	GPU (GB)	Speed (it/s)	Duration (h)	Speed (s/obj)
NWD [64]	256^3	$(46/256)^3 = 5.8‰$	5.3 (+12)	15.3	84	3.6
TetraDiffusion [69]	192^3	$(192/192)^3 = 100\%$	78.2	0.3	-	33.3
UDiFF [200]	256^3	$(46/256)^3 = 5.8‰$	4.5 (+12)	4.89	84	3.4
Our SpoDify	256^3	$512/256^3 = 0.03‰$	7.2	100	3	3.3

4.3.4 Results

Qualitative comparison To evaluate the performance of our method, SpoDify, we first conduct a qualitative evaluation against several leading 3D mesh generation models while focusing later on the trade-off between performance and model complexity. We display the qualitative comparison in Figure 4.7. The evaluation is performed on the ShapeNet categories airplane and chair, where we compare the results of SpoDify to those of NWD [64], UDiFF [200], and 3DShape2VecSet [195]. Our primary objective is not to surpass the SOTA generation models but to demonstrate that our method can achieve comparable or near-SOTA performance with a significantly less complex architecture and less computational power. While being able to fully capture the diversity of the dataset, our model maintains the structural properties of the underlying dataset. We notice that generated samples rarely present floating material or other structural artifacts that are physically incoherent in practice.



(a) 3DShape2VecSet [195]



(b) NWD [64]



(c) Our SpoDify

Figure 4.7: Qualitative comparison of chairs generated by different methods: (a) 3DShape2VecSet [195], (b) NWD [64], and (c) our SpoDify. The results from 3DShape2VecSet show a significant amount of artifacts, e.g., floating material and incomplete component; the results from NWD perform better in results plausibility but tend to be simple designs; meanwhile, our SpoDify is able to generate complicated designs and meantaining a high level of plausibility.

Quantitative comparison In the further quantitative comparison with SOTA methods, as results shown in Table 4.3, our SpoDify performs competitively across various metrics, including 1-nearest-neighbor accuracy (1-NNA), minimum matching distance (MMD), coverage (COV), and Jensen-Shannon divergence (JSD), indicating that it can generate high-quality 3D shapes similar to those produced by more complex models. In particular, our model outperforms the state of the art on the chair dataset. Moreover, COV values are consistently large, as the implicit representation adopted ensures the capability of reconstructing training samples.

Shape novelty analysis This study examines the capacity of our proposed method to generate novel shapes with respect to the ones in the training dataset. To do so, we build on the work by [152] and synthesize 500 chairs using SpoDify. All shapes (generated and from the training set) are preprocessed through normalization into a unit cube, ensuring a standardized and equitable framework for comparison. We employ Chamfer Distance (CD) as the metric to quantify the similarity between shapes. We use CD to determine the sample in the training dataset that is the most similar to the

generated chair. From the CD distribution shown in Figure 4.6, we observe that our model generates not only shapes that closely match those in the training set (low CD) but also produces realistic shapes that differ significantly from the training set shapes (high CD).

Efficiency comparison Table 4.4 highlights the comparative efficiency of various methods regarding training and inference requirements. Additionally, it reports the compression ratio between the input mesh and its corresponding representation used as input to the training model. Our proposed method demonstrates outstanding improvements in training speed and duration due to the impressive compression rate while requiring minimum GPU usage, while achieving comparable state-of-the-art results. Specifically, our method achieves the fastest training speed at 100 iterations per second (it/s), thanks to its streamlined model backbone, which significantly outpaces competing methods such as NWD [64] (15.3 it/s) and UDiFF [200] (4.89 it/s). This acceleration reduces the time required for large-scale training scenarios, making it feasible to handle extensive datasets without excessive computational cost. For inference, our method achieves a speed of 3.3 seconds per object, marginally surpassing UDiFF (3.4 s/obj) and vastly outperforming TetraDiffusion [69] (33.3 s/obj).

Moreover, our model leverages an efficient data compression strategy, achieving a compression rate of $512/256^3 = 0.03‰$, which contributes to reduced memory overhead during processing. Meanwhile, 3DShape2VecSet [195], while effective in certain scenarios, is hindered by the exceptionally large size of its data representation—approximately 300 GB—making it challenging to download and impractical to evaluate comprehensively under typical resource constraints.

4.4 Conclusion

Unlike prior works that rely on training to store shape information within the autoencoder parameters, our approach introduces a novel application of singular value decomposition to extract two components from the source data: (1) spectral features, a compact latent representation utilized to train the generative model, and (2) the basis, which provides foundational information for data reconstruction and is preserved during decomposition for reuse during the generation process. We are the first work to use the outputs of singular value decomposition as inputs for training generative models, we demonstrate that its outputs (spectral features and basis) can be effectively leveraged for generative modeling, enabling the synthesis of novel shapes.

Furthermore, our SpoDify not only maintains high-quality reconstruction but also achieves significant improvements in scalability, reducing training times from days to hours and compressing data dimensionality from gigabytes to megabytes. These advancements mark a crucial step toward making 3D generative models practical for handling high-dimensional data, especially when the amount of data is limited, and open up new avenues for research and applications.

Chapter 5

Learning Efficient Representations for 3D-Surfaces

Despite the advancements made in generating plausible engineering designs in the both forms of cross-section blueprints or meshes, utilizing natural CAD data remains our ultimate objective. However, as we initiated the study, we found that both the data representation and the generative capabilities of DGMs had not yet sufficiently advanced to address the generation of 3D B-Reps. With the growing interest and effort in this field, current advancements appear promising, as demonstrated in Section 1.4. Although these most recent works [66, 187] have already enabled the direct generation of B-Rep solids, they rely on UV-grids to represent the geometry of each surface, which is less optimal compared to the natural surface form, NURBS. Therefore, in this chapter, we propose NeuroNURBS, which addresses the research question 4: *How to enable DGMs to directly synthesize CAD native representation?* The content of this chapter has been published in paper [166].

5.1 Introduction

Boundary Representation (B-Rep) [177, 87] is commonly used to represent shapes and solids in computer-aided design (CAD) — widely applied in industrial design, simulation, and manufacturing. In a B-Rep, the solid boundaries are defined using a set of

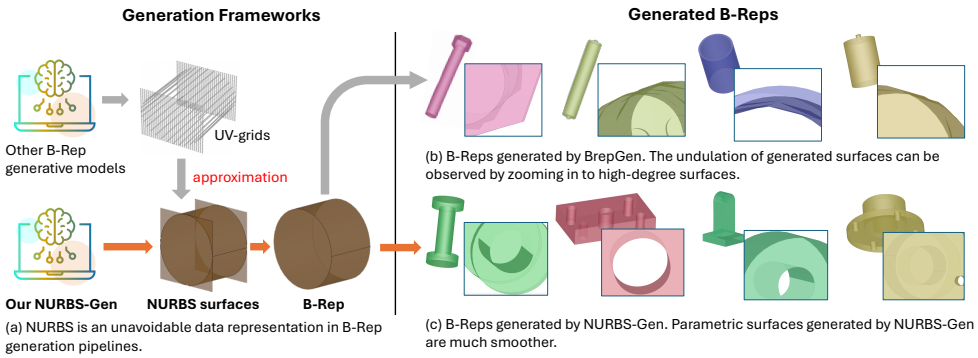


Figure 5.1: Why do we need UV-grids, if we could directly learn on NURBS. (a) Comparison of various generation frameworks. (b) Visualization of generated results. Our *NURBS-Gen* can directly generate valid NURBS parametrization and hence ensure the smoothness and regularity of the surfaces, in contrast to the BrepGen surfaces, which appear undulating.

surfaces [177, 39], which are, by default, parameterized by Non-Uniform Rational B-Splines (NURBS) [128]. Deep learning could offer solutions to several computational tasks important to the industry, e.g., B-Rep generation and solid segmentation. Aiming to solve these tasks, various works have been devoted to learning the geometry and topology of B-Rep data [67, 36, 66, 179, 187]. A major work, BrepGen [187], decomposes the B-Rep entities (faces, edges, vertices) into a tree data structure that encodes the topological information.

Despite the success of learning the topology of B-Rep, we find that these works [187, 67, 101] utilize a parametric approximation to surfaces, that is representing the surface with a certain number of 3D points uniformly distributed in the UV-domain, i.e., a UV-grid [67]. More details about UV-grids are introduced in Section 5.2. However, this approach has several drawbacks: (1) the approximation to the target surface is often imprecise unless a dense UV-grid is used to achieve an acceptable error range [67, 111, 109]. (2) The demand for high accuracy often incurs large data sizes, model sizes, and computational costs, which is sometimes unnecessary. For example, planar surfaces are represented by 32×32 3D points in [187]. (3) Generative models that use UV-grid-based surface approximation, e.g., BrepGen, can produce artificial undulating patterns on the surface. For instance, we showcase several solids generated by BrepGen in Figure 5.1 (a), where some sections of the curvy surface are not perfectly smooth.

Motivation Alternatively, it is more natural and advantageous to use the NURBS parametrization in the solids learning task: NURBS are more accurate [128, 109, 39], easier to manipulate [127, 63], and have less parameters [128, 44] compared to UV-grids. However, incorporating the parameters of a NURBS surface in deep neural networks is not trivial: (1) The parameters — control points, knot vectors, and weight matrix (see explanation in Section 5.2) — have varying sizes across surfaces in a B-Rep data set. It is challenging to unify them into a fixed-size input to a deep neural network. (2) These parameters are related (e.g., control points are only meaningful together with knot vectors and and the dimension of the weight matrix depends on the number of control points), and hence, they should be encoded and decoded jointly if we wish to learn a shared latent representation thereof.

Contribution We target learning *an effective representation of NURBS parameters*, for which we design a pipeline that preprocesses raw B-Rep solids into learnable NURBS parameters and propose the *NeuroNURBS model* to autoencode the heterogeneous sizes of NURBS parameters. We refer to the resulting latent representation as *NURBS features*. More precisely, *NeuroNURBS* is our proposed pipeline to convert *NURBS parameters* into *NURBS features* that can be used in downstream tasks, e.g., solid segmentation and generation. In Figure 5.1 (b), we illustrate some examples of surfaces generated with NURBS-Gen, which is obtained by replacing the preprocessing and autoencoder of surface entities with NeuroNURBS in the BrepGen [187] framework. As seen from Figure 5.1, the surfaces generated with NeuroNURBS and NURBS features are very smooth and regular. To evaluate NeuroNURBS, we conduct the following experiments:

1. We compare its performance with the UV-grid method for reconstructing surfaces on DeepCAD [182] in terms of accuracy, memory efficiency, and computation cost (Section 5.4.2). We see the memory required to store features of solids is reduced by **79.9%**, the surface autoencoder's size is reduced by **92.9%**, the GPU consumption for training the surface autoencoder is reduced by **86.7%**. We further test on the ABC [76] dataset, where our method NeuroNURBS shows a near-perfect surface reconstruction.

2. We test the NeuroNURBS on two downstream tasks: B-Rep generation on DeepCAD [182] and ABC [76] (Section 5.4.3) and solid segmentation on MFCAD [23] (Section 5.4.4). For the former, our NURBS-Gen improves the FID (Fréchet Inception Distance [58]) from 30.04 (achieved by BrepGen) to **27.24** on DeepCAD

dataset. For the latter, using NeuroNURBS achieves an accuracy of **99.65%**.

5.2 Preliminaries

UV-grids A 3D freeform surface $\boldsymbol{S}$ can be approximated by a parameterized function $\mathbf{F}(u,v)\colon \mathbb{R}^2 \to \mathbb{R}^3$, which is learned on a number of points evenly distributed in $[u_{\min}, u_{\max}] \times [v_{\min}, v_{\max}] \in \mathbb{R}^2$. $\mathbf{F}(u,v)$ is called UV-grid function (see the left side of Figure 5.2). To learn the grid function, one can sample $n \times m$ points (along the U and V dimensions, respectively) from the target surface. A large sample number can increase the approximation accuracy of the original surface but this comes at the cost of higher-dimensional representation, leading to increased training times and memory requirements.

Non-Uniform Rational B-Splines Non-Uniform Rational B-Splines (NURBS) are essentially B-splines applied in homogeneous coordinates, rather than 3D coordinates, i.e., $(x,y,z,w) \mapsto (x/w, y/w, z/w)$. Each NURBS of order n (polynomial of degree $n-1$) requires n control points and $2(n-1)$ knots. A NURBS curve $C(u)$ can be defined using coordinate u with the formula:

$$C(u) = \frac{\sum_{i=1}^{n} N_i^p(u) w_i p_i}{\sum_{i=1}^{n} N_i^p(u) w_i} \,, \tag{5.1}$$

where p_is are 3D control points and N_i^ps are basis functions of degree $p \leq n-1$, defined recursively on a knot vector $(u_1, \ldots, u_i, \ldots, u_{n+p+1})$:

$$N_i^p(u) = \frac{u - u_i}{u_{i+p} - u_i} N_i^{p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1}^{p-1}(u) \tag{5.2}$$

$$N_i^0(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \,, \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

A NURBS surface $\boldsymbol{S}(u,v)\colon \mathbb{R}^2 \to \mathbb{R}^3$ is the tensor product of two NURBS curves, $C_U(u)$ (on n control points with order p) and $C_V(v)$ (on m control points with order q):

$$\boldsymbol{S}(u,v) \coloneqq C_U(u) C_V(v) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} N_i^p(u) N_j^q(v) w_{ij} p_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{m} N_i^p(u) N_j^q(v) w_{ij}},$$

which takes the following parameters: a grid of control points $p \in \mathbb{R}^{n \times m \times 3}$, weights $\boldsymbol{W} \in \mathbb{R}^{n \times m}$, the U-direction knot vector $\boldsymbol{U} = (u_1, \ldots, u_{n+p+1}) \in \mathbb{R}^{n+p+1}$, and the

V-direction knot vector $\boldsymbol{V} = (v_1, \ldots, v_{n+q+1}) \in \mathbb{R}^{m+q+1}$.

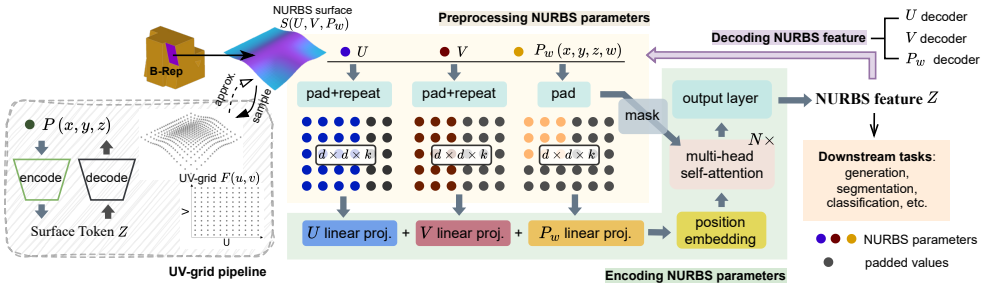


Figure 5.2: Diagram for NeuroNURBS. *Right:* two parts of NeuroNURBS, preprocessing and autoencoding NURBS parameters. *Left:* a simplified diagram for UV-grid, where the approximation from UV-grid back to NURBS surface is not deterministic.

5.3 Method

UV-grids enable the direct learning on B-Rep solids, but a UV-grid still serves as an approximation to the source surface, whereas using NURBS parametrization is a more natural and advantageous choice: NURBS are more accurate [128, 109, 39] and have less parameters [128, 44]. However, using modern neural networks to operate NURBS parameters is understudied and remains challenging (see Section 5.3). To address this, in this section, we describe NeuroNURBS which consists of a preprocessing pipeline for NURBS parameters and an autoencoder that is able to encode NURBS parameters into a low-dimensional feature space for downstream application.

To represent a surface, we propose to take its NURBS parametrization directly, which can be extracted from the B-Rep model, e.g., with `OpenCascade` functionalities. Compared to the UV-grid approach, the NURBS parametrization is (1) smooth and describes the surface precisely; (2) it can be memory efficient since it only requires storing a collection of 3D control points, a weight matrix, and two knot vectors. However, developing a deep learning model that takes as input the parameter of NURBS is challenging:

- NURBS parameters have different dimensions, e.g., a knot vector is a 1D object and control points are 3D objects and vary in size from one surface to the other.
- for a generative task, NURBS parameters have to be combined in the encoder and the decoder must map each latent point to a set of valid parameters.

We propose NeuroNURBS to resolve these challenges. It consists of two components: a preprocessing pipeline for the NURBS parameters (see below) and an autoencoder to learn an effective representation of NURBS parameters. We depict our method in Figure 5.2.

Table 5.1: Performance comparison of UV-grids and NURBS in surface representation tasks. All are evaluated on the DeepCAD dataset, except for the surface reconstruction, which is evaluated on the DeepCAD *(D)* and ABC *(A)*. We find that the reconstruction on NURBS data can achieve comparable accuracy as the one on UV-grids while offering significant advantages in representation efficiency.

	Input data size		Autoencoder size		Surface reconstruction		Construction speed
	Surface (MB/10k)	Solid (GB/10k)	VAE #Param.	GPU (GB)	CD ($\times 10^{-5}$) ↓	EMD ($\times 10^{-3}$) ↓	speed (NURBS/s)↑
UV-grids	245.8	37.7	84M	17.61	4.47 *(D)*, 3.20 *(A)*	1.90 *(D)*, 1.52 *(A)*	230
NURBS	**8.16** (96.7% ↓)	**7.6** (79.9%↓)	**6M** (92.9%↓)	**2.35** (86.7%↓)	4.23 *(D)*, 3.04 *(A)*	1.83 *(D)*, 1.67 *(A)*	**3230** (92.9%↑)

5.3.1 Preprocess NURBS Parameters

We consider the **control points** p, **weights** $\boldsymbol{W}$, and **knot vectors** $\boldsymbol{U}$ and $\boldsymbol{V}$ for the learning task, which we call the NURBS parameters. We exclude degrees p (degree in the U-direction) and q (in the V-direction) from parameters since they can be easily calculated: p is the length of knot vector $\boldsymbol{U}$ minus $n + 1$, where n is the number of control points in the U direction. The degree q can be determined in the same way.

Normalization For control points $p_{ij} = (x_{ij}, y_{ij}, z_{ij})$, we determine the coordinate axis with the largest range among three coordinates:

$$d = \max\left(x_{\max} - x_{\min}, y_{\max} - y_{\min}, z_{\max} - z_{\min}\right), \tag{5.4}$$

and then normalize the coordinates as $\hat{x}_{ij} = (x_{ij} - x_{\min})/d$ (same for y-axis and z-axis). The normalization ensures $(\hat{x}_{ij}, \hat{y}_{ij}, \hat{z}_{ij}) \in [0, 1]$. Note that weights w_{ij} also take values in the unit interval. Hence, we concatenate p and $\boldsymbol{W}$ together: $\boldsymbol{P_w} = [(\hat{x}_{ij}, \hat{y}_{ij}, \hat{z}_{ij}, w_{ij})]_{i\in[1..n], j\in[1..m]} \in \mathbb{R}^{n\times m\times 4}$.

Padding With a padding value of 0, we augment $\boldsymbol{P_w}$ to a tensor of size (d, d, 4), where d is the largest number of controls points across all surfaces in a data set. We also pad the knot vectors $\boldsymbol{U}$ and $\boldsymbol{V}$ to length k (the length of the longest knot vector in a data set). To combine the knot vector and control points, we duplicate the knot

vector to a tensor of shape (d, d, k). Also, we save the padding mask of $\boldsymbol{P_w}$ for the autoencoder. Although the padding seems to impair the efficiency of the NURBS representation, the padding parts are omitted when training the transformer-based autoencoder using the padding mask.

5.3.2 Learning NURBS Features with Autoencoder

We study recent works in multi-modal and multi-task autoencoding [113, 43, 8, 121] and design a VAE model to learn a latent representation of the NURBS parameters, which we call *NURBS features*.

Our autoencoder is inspired by MultiMAE [8]. We use separate dense layers for each preprocessed NURBS parameter (control points with weights $\boldsymbol{P_w}$, U-knot vector $\boldsymbol{U}$ and V-knot vector $\boldsymbol{V}$). The sum of the dense outputs is then processed with a sine-cosine position embedding and is then input into a transformer together with the control point padding mask. For the selection of a different transformer backbone from MultiMAE [8], we do not use the Vision Transformer (ViT) [78] as our data has a low dimension and does not require patch embedding, but the introduction of a padding mask would be helpful. Thus, we select a transformer backbone from BERT [40]. Each transformer block is eight layers of four-head self-attention blocks. Followed up with a fully connected dense layer, we obtain $\mu \in \mathbb{R}^{d_z}$ and $\sigma \in \mathbb{R}^{d_z}$, which will then be reparameterized into the surface token $Z \in \mathbb{R}^{d_z}$, like every VAE model. To reconstruct the NURBS parameters from the NURBS feature Z, we employ multi-task decoding, namely a separate decoder for each parameter. See Figure 5.2 for the visualization of the model architecture.

During the training, we compute the loss: $L = \sum_{i=1}^{3} \|x_i - x_i'\|_2^2$, where $x_1 = P_w$, $x_2 = U$, $x_3 = V$ are the NURBS parameters, and $x_1' = P_w'$, $x_2' = U'$, $x_3' = V'$ are the reconstructed NURBS parameters. Practically, adding a KL-divergence term could help convergency. For the VAE training and inference, inputs are normalized to $[-1, 1]$ and outputs of decoding will be denormalized to $[0, 1]$. Notably, for generated NURBS parameters, we perform the following post-generation checks: we remove control points with weight less than or equal to 0, and clip knot vectors once the value in the sequence reaches 1.

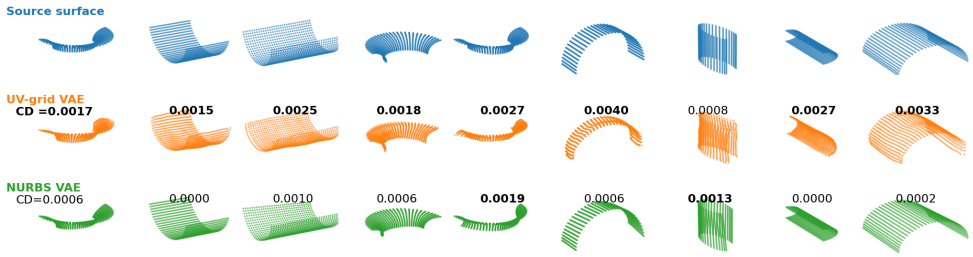


Figure 5.3: Qualitative evaluation of surface reconstruction on the ABC dataset. Considering that both pipelines work perfectly well on simple surfaces, we select surfaces of degree ≥ 5 for qualitative evaluation. We visualize results in evenly sampled points and marked measured CD (the higher the worse) on the sample. *First row*: source surfaces; *Second row*: surfaces reconstructed by UV-grid VAE of BrepGen; *Third row*: surfaces reconstructed by NURBS-based VAE of NeuroNURBS.

5.4 Experiments

We performed several experiments to investigate the effectiveness of our NeuroNURBS. NURBS parameters are more efficient than UV-grids for surface representation and can also be used as learning data. To demonstrate this, we compare its performance with UV-grids w.r.t. accuracy, memory efficiency, and computation cost (Section 5.4.2) on the DeepCAD dataset. After showing NURBS's utility in efficient representation, we introduce NURBS features encoded by NeuroNURBS into two downstream tasks: (1) The CAD (B-Rep) generation task on the DeepCAD dataset(Section 5.4.3), and (2) The solid segmentation task on the MFCAD data set (Section 5.4.4).

5.4.1 Datasets

DeepCAD We introduce the DeepCAD [182] and the ABC [76] datasets for the experiment of surface reconstruction and solid generation. The DeepCAD dataset originally contains CAD data in the form of a sequence of engineering operations that can be converted into B-Reps with their pre-defined construction algorithm. We follow the previous work [187], i.e., filtering out invalid B-Reps shapes (in case of multiple solids or being not watertight) and B-Reps with more than 30 surfaces. We use the train-val-test splitting of DeepCAD, resulting in 69 512 B-Reps for training and 7 083 B-Reps for testing; for ABC dataset, we randomly split the samples with the 90-5-5 ratio. Closed surfaces, e.g., cone surfaces, are cut into two separate NURBS surfaces as done in [66]. In representing surfaces as UV-grids, we set $n = m = 32$, the same as [187]. For training the autoencoder in NeuroNURBS, we list all

NURBS surfaces from the training solids and remove duplicated surfaces, obtaining 59 961 unique NURBS surfaces from the DeepCAD dataset (163 633 from ABC). The preprocessing of NURBS surfaces has been detailed in Section 5.3. Here, we set the NeuroNURBS hyperparameters $d = 10, k = 10, d_z = 48$. In Section 5.3.

For the solid segmentation experiment in Section 5.4.4, we introduce the MFCAD [23] dataset of 15 488 CAD solids in the STEP file format, which is the standard file format for B-Rep. MFCAD is designed for the machining feature segmentation task, where there are 16 various face labels and each face is labeled with a certain class. We split the MFCAD solids into train-val-test datasets with a ratio of 70-15-15, resulting in 10 842 solids for training. For all surfaces from training solids, we first remove duplicated surfaces and then split the surfaces into train-val-test datasets with a ratio of 90-5-5, where we collect 24 603 NURBS surfaces for training the NeuroNURBS. The surface embedding follows the pipeline described in Section 5.3, where we set $d = 2, k = 4, d_z = 48$.

5.4.2 Surface reconstruction

We present a comparative analysis between our proposed method and the UV-grid approach for surface reconstruction. For the UV-grid representation, we adopt the surface VAE and the weights from BrepGen [187] as a baseline. Our objective is to systematically evaluate the performance of our NeuroNURBS (the NURBS VAE, introduced in Section 5.3.2) relative to the UV-grid VAE from BrepGen. The comparison focuses on several key criteria: learning data size, model size, reconstruction error, distribution learning capability and NURBS construction speed. A summary of the results is provided in Table 5.1.

Input data size We randomly sample 10k surfaces from the source solids, collecting the corresponding NURBS parameters and sampling UV-grid on a 32×32 resolution for each surface. As shown in Table 5.1, representing surfaces with NURBS parameters reduces memory usage by 96.7% compared to UV-grid data. For solid representations, B-Rep entities are structured as trees and surfaces are encoded either as UV-grids or NURBS. In this setting, using NURBS for surface representation leads to a 79.9% reduction in memory cost.

Computational cost and reconstruction error The VAE model size and computational requirements differ significantly between representations. In Table 5.1, we observe the surface VAE from BrepGen has 84M parameters, which takes a UV-grid

of dimension 32×32. In contrast, our method requires only 6 million parameters. When training both VAEs with a batch size of 512, the UV-grid VAE requires 17.61 GB of GPU memory, while our NURBS-based method uses only 2.35 GB—an 86.7% reduction.

For reconstruction accuracy, we report the average Chamfer Distance (CD) and Earth Mover's Distance (EMD) between input and reconstructed surfaces, evaluated on 10k pairs with 32×32 uniformly sampled points per surface. In the early stages of this work, we evaluated the reconstruction using MMD, but it turned out to be unsuitable for reconstruction accuracy because it does not directly measure the distance between the source and the generated surface. The results of CD and EMD, summarized in Table 5.1, quantify the reconstruction error for both methods. For UV-grid representations, source and reconstructed UV-grids are directly used for evaluation. We note that the performance of NURBS VAE and UV-grid VAE is comparable in the case of sufficient training, while NURBS has a clear advantage in terms of representation efficiency. In Figure 5.3, we demonstrate the reconstruction of complex surfaces labeled with chamfer distances. Although the undulation problem is easily observed on surfaces reconstructed by UV-grid VAE, their measured CDs are concerningly better than the ones of NURBS VAE.

NURBS construction In B-Rep generation tasks, UV-grid representations require a surface fitting step (e.g., `GeomAPI_PointsToBSplineSurface` in `PythonOCC`) to convert grids to NURBS surfaces. In contrast, our NeuroNURBS approach only requires denormalizing the NURBS parameters (see Section 5.3.1) and passing them to a CAD tool (e.g., `Geom_BSplineSurface` in `PythonOCC`), which is computationally inexpensive. As shown in Table 5.1, our method increases NURBS construction speed by 92.9%.

Most importantly, as a main flaw of using the UV-grid representation and the biggest advantage of using NURBS directly, we investigate the degree evolution during surface reconstruction. Table 5.2 presents the degree distributions for the test set, the surface VAE, and NeuroNURBS. Our method precisely matches the degree distribution of the test data, as it directly learns a latent representation of the NURBS parameters. In contrast, UV-grid methods tend to produce surfaces with higher polynomial degrees, leading to increased data size of resulted NURBS and the introduction of artificial undulations, as illustrated in Figure 5.1. For a fair comparison, we have removed the lower degree constraint in the `PythonOCC` approximation function. In the Supplementary Material, we also add the surface degree distribution of real-world

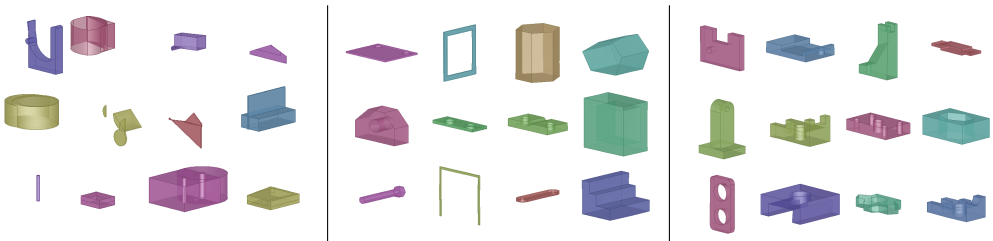
Table 5.2: The empirical distribution of the degree of polynomials along U(V) direction are measured in two tasks: surface reconstruction and generation. To reconstruct surfaces, BrepGen heavily leverages the UV-grid VAE and the UV-to-NURBS approximation function, which causes serious distortions in the surface properties (e.g., planes become uneven), as we highlight in red in the table. Note that, for this test, we remove the bottom limitation of degree in the `PythonOCC` approximation function. On the other hand, having the same degree distribution in both test and reconstructed surfaces means **a perfect reconstruction** of our NeuroNURBS. In addition, we added the surface degree distribution of real-world solids (car rims), where 30.41% of the high-degree surfaces (≥ 5) have a degree of 5.

NURBS source	Distribution of surface degrees			
	$d = 2$	3	4	≥ 5
Surface reconstruction				
Test data (surfaces)	88.59%	2.03%	0.00%	9.38%
UV-grid VAE [187]	0.00%	94.01%	5.92%	0.07%
NeuroNURBS	**88.59%**	**2.03%**	**0.00%**	**9.38%**
Solid generation (DeepCAD dataset)				
Test data (solids)	93.45%	2.32%	0.00%	4.50%
BrepGen [187]	0.00%	0.00%	99.99%	0.01%
NURBS-Gen	88.69%	0.88%	0.00%	10.43%
Solid generation (ABC dataset)				
Test data (solids)	82.26%	8.26%	0.04%	9.44%
BrepGen [187]	0.00%	0.00%	99.76%	0.23%
NURBS-Gen	80.26%	2.96%	0.26%	16.5%
Real-world solids	35.96%	28.51%	0.81%	30.41% (d=5) + 3.65%

solids (car rim), where which sharpens the drawbacks of using a UV-grid.

5.4.3 CAD (B-Rep) Generation

BrepGen [187] generates B-Rep solids by progressively creating the B-Rep entities (i.e., faces, edges, and vertices) following a hierarchical tree structure from top to bottom with four diffusion models, which are Transformer-based DDPM [61]. For generating faces and edges, each node in the tree structure contains a node feature representing the global bounding box of each face/edge and a latent code for local geometry, e.g., the shape of the surface. BrepGen represents all B-Rep surfaces as UV-grids, and fits UV-grids to NURBS surfaces after generation. Here, we substitute the surface VAE model, implemented by BrepGen for autoencoding UV-grids, with our NeuroNURBS (Section 5.3.2). Technically, we set the latent dimension of our NURBS VAE to $d_z = 48$, which is exactly the same as the latent dimension of surface VAE in BrepGen. Then, we fine-tune the four pre-trained diffusion models in BrepGen



(a) DeepCAD **(b)** BrepGen **(c)** NURBS-Gen

Figure 5.4: Qualitative evaluation. It is observe that DeepCAD often generates invalid (i.e., multiple and collapsed) solids; while BrepGen and our NURBS-Gen are able to synthesize clean and upstanding B-Rep solids. In Figure 5.1, we visualize generated samples from BrepGen and NURBS-Gen with zooming-in for a deep-diving qualitative evaluation.

(i.e., diffusion models for face bounding box, face geometry, edge bounding box, edge geometry, and vertex coordinates) on the DeepCAD data set. We call the resulting model NURBS-Gen. As for the details, face denoisers are fine-tuned with a batch size of 256 for 1 000 epochs, the edge denoisers with a batch size of 64 for 500 epochs, and the training is conducted using one NVIDIA A10G GPU. As for the inference, the sampling process of NURBS-Gen follows the pipeline of BrepGen and the surface decoding is replaced by the decoder in NeuroNURBS.

Results We generate 3 000 B-Rep solids with DeepCAD model [182] (only for DeepCAD data, because the ABC data contains no construction operations for training DeepCAD), BrepGen [187], and our NURBS-Gen and compare them to 1 000 test solids. For the DeepCAD model and BrepGen, we utilize their model checkpoints that are pre-trained on the target dataset. Between the generated and test solids, we compute Minimum Matching Distance (MMD), Coverage (COV), and Jensen-Shannon Divergence (JSD) metrics. To calculate these metrics, we convert each solid into a point cloud with 2 000 points, following the evaluation method in [182, 187]. We show the results in Table 5.3.

We argue that the above evaluation method, which takes as input point clouds, might be biased toward methods like BrepGen, which utilizes UV grids. Hence, we additionally measure the Fréchet Inception Distance (FID) [58] between the 3D solids rendered in 2D with a certain viewpoint. We repeat the FID calculation for 20 different viewpoints and report an average FID value, as proposed in [184, 196].

To check the validity of the generated solids, we first convert each generated B-Rep solid to a STEP file until 3 000 STEP files are obtained. Then, we use the

Table 5.3: In the CAD (B-Rep) generation task, we list the performance of NURBS-Gen and other related models in terms of the COV, MMD ($\times 10^{-2}$), JSD ($\times 10^{-2}$), and Valid ratio.

Model	MMD ↓	JSD ↓	COV (%) ↑	FID ↓	Valid (%) ↑
DeepCAD dataset					
DeepCAD	1.41	3.92	**77.9**	**14.36**	34.79
BrepGen	**1.02**	1.16	74.7	30.04	60.95
NURBS-Gen	1.10	**0.98**	73.9	27.24	**64.58**
ABC dataset					
BrepGen	1.68	3.1	46.4	24.28	48.1
NURBS-Gen	**1.51**	**3.0**	**52.3**	**21.12**	**50.7**

`BRepCheck_Analyzer()` and `IsValid()` functions from `pythonOCC` to test the watertightness of each solid. From this process, we calculate a valid ratio among all generated solids (including the ones not saved as STEP files) for each method.

Based on the results of the quantitative evaluation in Table 5.3, NURBS-Gen consistently outperforms BrepGen with a slight edge. Note that the main advantages brought by our approach are representation efficiency, better surface quality and a CAD-native approach. Also, we visualize some generated solids from DeepCAD, BrepGen, and NURBS-Gen in Figure 5.1 and Figure 5.4. In Figure 5.1, we observe that BrepGen solids are undulating, i.e., they exhibit the wobbly geometry described in [187]. In contrast, our NURBS-Gen can generate smooth and regular surfaces and accurately join the surfaces.

5.4.4 Segmentation

In the task of machining feature segmentation [23, 36], current state-of-the-art model [67] tends to leverage increasing information of geometry or topology (e.g., face normal vector, face geometry, trimming mask, face adjacent or edge features) from the source B-Rep solid to perform accurate machining feature recognition.

We design a NURBS-based segmentation model, NURBS-GAT, that leverages a Graph Attention Network (GAT) [175] to operate the graph data derived from B-Rep. Replacing the graph convolutional network (GCN) in CADNet with a GAT is due to constant memory issues when increasing the size of the node features. For the graph data, a stack of NURBS feature encoded with NeuroNURBS, face normal vector $\boldsymbol{n}$ and coefficient d calculated from a planar equation serves as the $(48+3+1)$-dimensional node feature V and the face adjacency serves as edge connection, as shown

Table 5.4: Ablation study on machining feature segmentation task. Although UV-GAT shows a comparable results to our NURBS-GAT, but their model size is significantly larger.

Ablated model	Acc. (%)	#Param.
CADNet (with GAT)	92.18	0.02M
UV-GAT	99.63	34.2M
NURBS-GAT (only NURBS feature)	97.03	1.28M
NURBS-GAT	**99.65**	1.29M

in Figure 5.5. Now, we use the ablation study to show the importance of each input feature by conducting the ablation study on NURBS-GAT.

NURBS-GAT (only NURBS features) We remove the face normal vector $\boldsymbol{n}$ and coefficient d from the node features of NURBS-GAT, resulting in node feature $V \in \mathbb{R}^{48}$.

CADNet (with GAT) We remove the NURBS feature from the node of NURBS-GAT, resulting in node feature $V \in \mathbb{R}^{4}$.

UV-GAT We replace the NeuroNURBS embedding with UV-grids embedding in node features of NURBS-GAT, resulting in node feature $V \in \mathbb{R}^{52}$. The UV-grids embedding is obtained by training a surface VAE model, same as in Section 5.4.2, on MFCAD surface training data.

The evaluation in ablation study is conducted by reporting the per-face accuracy in Table 5.4, following the usual process in the machining feature segmentation task [36, 67]. Observed from the results, utilizing the NeuroNURBS embedding is able to improve the per-face accuracy from 92.18% to 99.65%, which shows a comparable result to UV-grid embedding. In Figure 5.6, we visualize the segmentation task and its results: since the per-face accuracy has already achieved 99.65%, the predicted labels are the same as the ground truth in the qualitative evaluation.

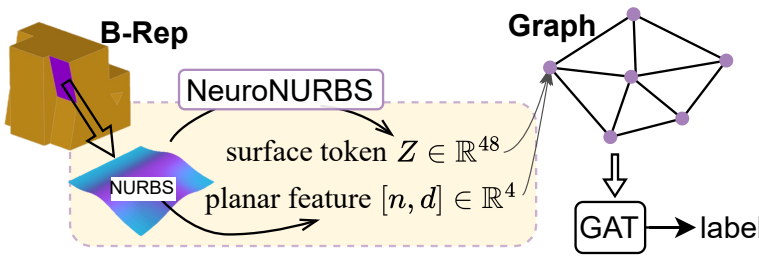


Figure 5.5: NURBS-GAT diagram.

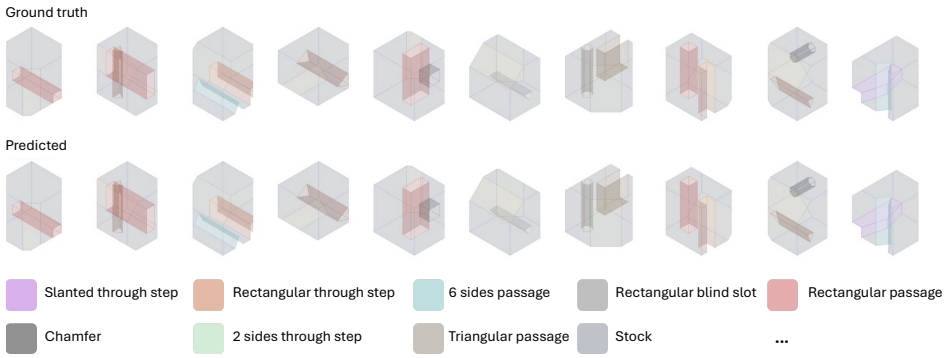


Figure 5.6: Qualitative evaluation of NURBS-GAT in machining feature segmentation task.

5.5 Implementation Details

Get NURBS parameters. In a B-Rep solid, the surfaces usually exist in the form of trimmed NURBS surfaces. To convert them into NURBS parameters, i.e., control points, weights, and knot vectors, we design the following pipeline with functions from `OpenCascade` and `pythonOCC` [124]:

1. Convert the input with form of `TopoDS_Face` into a untrimmed NURBS surface with function `BRepBuilderAPI_NurbsConvert()`, followed up with `Surface()` from `BRep_Tool`.

2. Use `SurfaceToBSplineSurface()` from `geomconvert` to convert the NURBS surface into the from of `Geom_BSplineSurface`.

3. Finally, list the NURBS parameters from the `Geom_BSplineSurface` surface with `Pole()`, `Weight(u,v)`, `UKnotSequence()` and `VKnotSequence()`.

Note that the `BRepBuilderAPI_NurbsConvert` function can be applied to planes, resulting in 4 corner points and U/V knot vectors $[0, 0, 1, 1]$.

NURBS construction from NURBS parameters After removing the padded values from NURBS parameters in Section 5.3.1, the parameters need to be converted into certain formats, so that they can be used for defining the `Geom_BSplineSurface`: Every control point is converted in the form of `gp_Pnt` and the tensor of control points is presented in the form of `TColgp_Array2OfPnt`, serving as `Poles`; Weights will be seperated and stored in the form of `TColStd_Array2OfReal`, serving as `Weights`; We

round the values in the reconstructed U(V)-knot vector to 1 decimal and store its sorted set (a ordered list of unique values) in the form of `TColStd_Array1OfReal()`, which serves as `U(V)Knots`; and the number of times each variable repeated is recorded in `TColStd_Array1OfInteger`, serving as `U(V)Mults`. The constructed NURBS is then `Geom_BSplineSurface`(Poles, Weights, UKnots, Vknots, UMults, VMults, UDegree, VDegree).Surface().

One may concern that our postprocessing brings damage to the surface geometry. As we already pointed out in Table 5.2, using UV-grids brings real damage to the surface geometry, whereas using natural parameters helps. Besides, our evaluation results will point it out if the postprocessing brings damage to the geometry. Processing the NURBS parameters is to make it valid for the PythonOCC API and easier to remove padding. This does not cause significant geometric damage, which would otherwise destroy the shape and the evaluation results. In early training we need this for model validation, otherwise the generated NURBS parameters cannot form a NURBS surface with PythonOCC, after the model being well-trained, the postprocessing won't bring much difference to the predict values.

NURBS approximation from UV-grids In Section 5.4.2, we conduct an experiment to study the distribution of U(V) degree in NURBS construction with NURBS parameters and UV-grids. The results displayed in Table 5.2. As mentioned in Section 5.4.2, to obtain UV-grids the function `GeomAPI_PointsToBSplineSurface` is used. In the BrepGen pipeline, they constrain the range of resulting U(V) degree to be $[3, 8]$. In this experiment, we use $[0, 8]$, i.e., removing the minimum limitation, for a fair comparison.

5.6 Conclusion

With data representation becoming one of the major challenges for solid learning, our work directly operates the native representation of B-Rep surfaces, i.e., NURBS parameters, by proposing NeuroNURBS. Our NeuroNURBS can embed NURBS parameters into a low-dimension NURBS feature. Using NeuroNURBS has demonstrated its ability of significantly reducing storage and GPU consumption, while delivering a comparable performance to state-of-the-art method on both solid generation and segmentation. We also observe that NeuroNURBS can remarkably help with the undulation problem in generated surfaces, which could be caused by using an approximate representation (such as UV-grids) to represent source surfaces.

As a first attempt to integrate NURBS parameters into solid learning tasks, there is still much room for improvement in neural networks, e.g., directly learning on NURBS parameters without using an autoencoder. For the solid generation task, the edge geometry in NURBS-Gen is still represented by sampling points in the parametric domain, and it is also technically possible to replace it with NURBS parameters.

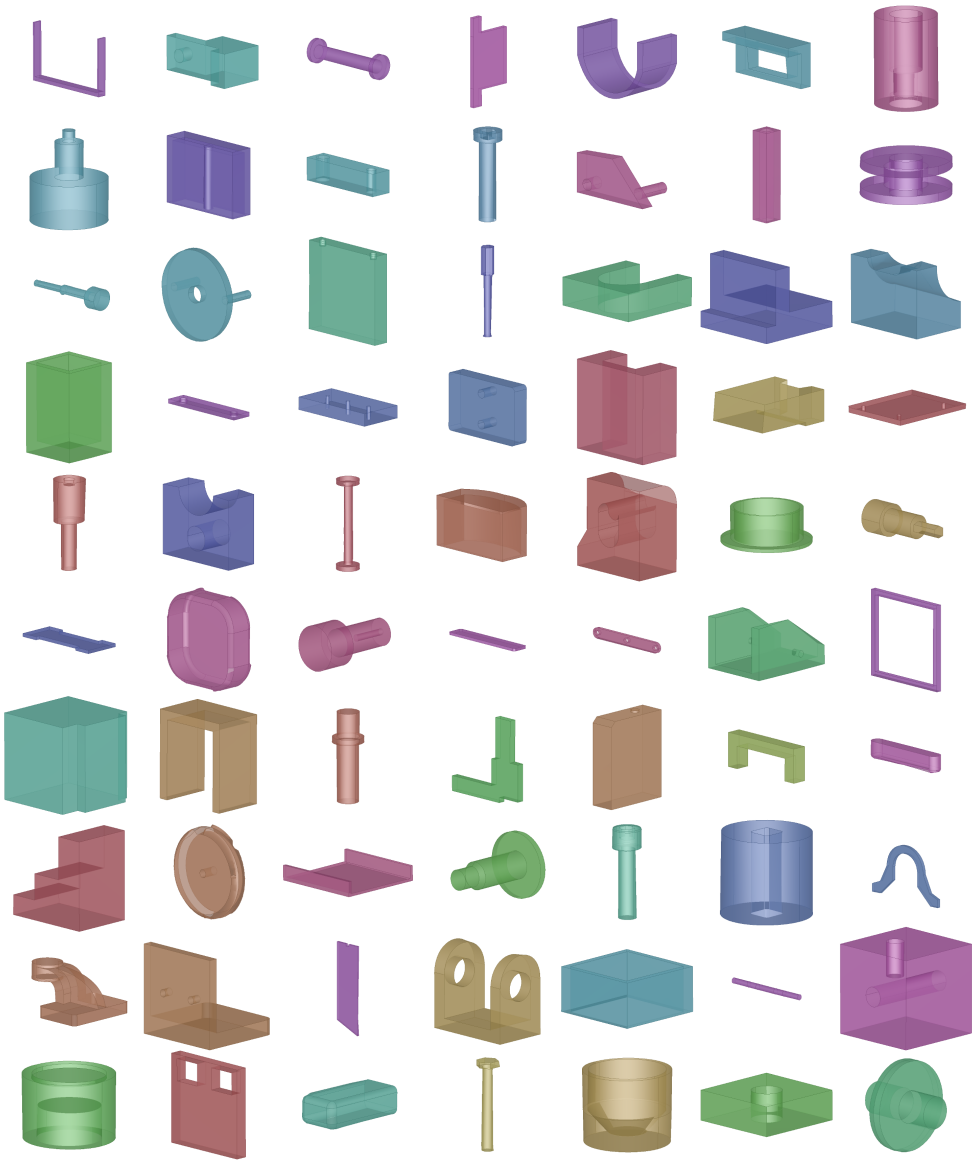


Figure 5.7: Qualitative evaluation of NURBS-Gen on the ABC dataset.

Chapter 6

Application in Industrial Development Processes

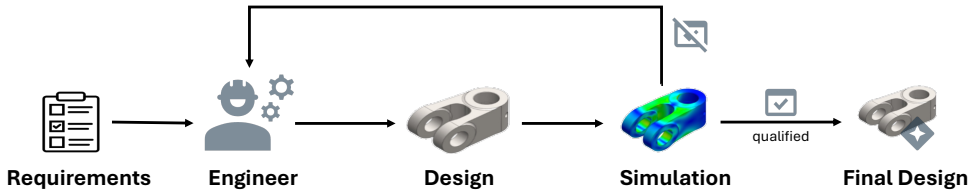
To demonstrate the industrial value of our work, it is essential to apply our methods in real-world industrial development processes, such as car design. This chapter focuses on two specific car components: A-pillars and rims. By evaluating with these components, we aim to assess the potential and capacity of our DGM-based approaches in aiding engineers with developing complex parts. This chapter is correspondingly aiming to address the research question 5: *How can DGMs be beneficial for real-world industrial design applications?*

6.1 DGM-based Generative Engineering Design

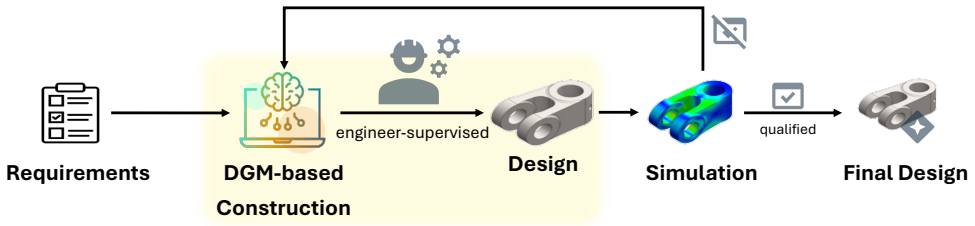
In the current industrial development process, while engineers can rely on a limited range of intelligent design tools, the initial design is often constructed manually based on the engineer's professional knowledge and predefined requirements (e.g., geometric constraints, cost limits, and other functional requirements). The initial design must be tested through certain simulations, and if the performance of certain functions is substandard, the engineer must modify the design, which happens in most cases. This iterative design process continues until the design has passed all simulations and meets the predefined requirements. This process is demonstrated in Figure 6.1a.

The current industrial development process has several key disadvantages:

- Limited design space exploration: The manual design approach constrains en-



(a) Traditional engineering design



(b) DGM-driven generative engineering design

Figure 6.1: Diagrams about traditional engineering design and DGM-driven generative engineering design.

gineers to a relatively narrow design space based on their own expertise and predefined requirements, limiting the potential for innovative or disruptive designs.

- Inefficient and error-prone design process: The heavy reliance on manual design construction by engineers, based on their own knowledge and experience, can be time-consuming, inefficient, and prone to human error, leading to suboptimal designs and longer development timelines.

- Underutilization of historical design knowledge: With the current manual design process, engineers are largely reliant on their own professional knowledge, rather than effectively leveraging the wealth of historical designs and accumulated organizational knowledge. This represents a significant missed opportunity, as the insights and experiences from past design efforts are not being efficiently captured and reapplied, leading to a wasteful duplication of effort.

This work focuses on enabling DGMs to directly generate engineering designs, with a certain level of supervision from engineers (e.g., design editing and generation conditioning). This is represented by the *yellow area* highlighted in Figure 6.1b. Unlike the traditional manual design construction process, the DGM-based design generation

is differentiable. This means the output design is numerically connected to the input parameters, allowing for gradient-based optimization. As demonstrated in the design editing experiments on BIKED (Section 2.4), introducing a gradient-based optimizer enables the iterative modification of the target design until it meets specific geometric requirements. When the simulation is differentiable (some of current simulation pipeline is already differentiable and modern AI-based simulation predictors are also differentiable) and the gradient can be calculated, the same optimization pipeline can be used to optimize the design to meet desired functional targets. Hereby, we come to a novel pipeline, shown in Figure 6.1b.

The key advantage of this DGM-based approach is that it allows for a more efficient and automated design exploration process, leveraging the differentiable nature of the generative model to guide the design towards optimal solutions, with the ability to incorporate both geometric and functional performance objectives. In this thesis, the whole pipeline would not be evaluated on real-world simulation, for the following reasons: the differentiable simulation is not available, simulation data differs from design data and the amount of trainable design data is not sufficient to enable this pipeline. But for the proof of the concept, we refer to the dragging edition in Section 2.4.

6.2 Automotive A-pillar Design

The current progress of DGM can generate feasible designs. Inspired by these results, we see the value of applying this method in actual industrial design cases. Therefore, this section will focus on the introduction of PoDM in the design process of a car A-pillar, using two-dimensional engineering blueprints as design data.

6.2.1 Background and Motivation

Federal Motor Vehicle Safety Standard known as FMVSS No.201U [171], which is provided by the National Highway Traffic Safety Administration (NHTSA) and Office of Vehicle Safety Compliance (OVSC), is the law guiding automobile manufacturers in designing the car interior as an “energy absorber” in crash. The law FMVSS No.201U requires that automobiles have to provide impact protection for passengers, which applies to passenger cars and trucks, buses and multipurpose passenger vehicles with a gross vehicle weight rating of 4 536 kilograms or less. For the impact test, Free Motion Headform (FMH) is implemented as the dummy head, which is presented in the middle of Figure 6.3. The FMH impactor is capable of propelling the FMH at

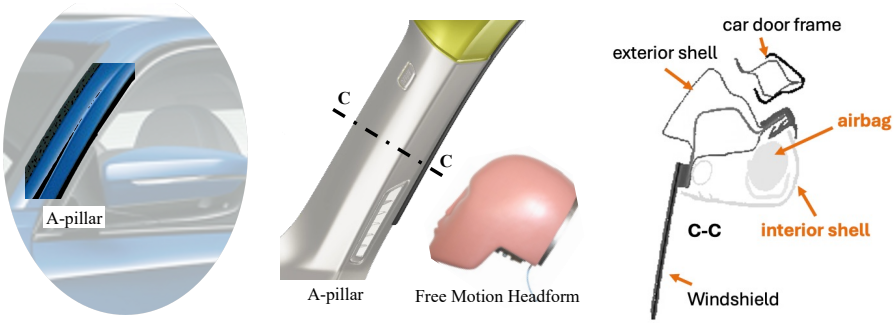


Figure 6.2: A vehicle A-pillar (*left*), an illusion of the slicing on the A-pillar 3D model and the free motion headform (*middle*), a cross-section blueprint for A-pillar design (*right*). In the cross-section blueprint for A-pillar design, the given structures are drawn with dark black lines and the target design parts are drawn with light black lines (e.g., airbag and interior shell).

a desired impact speed with ensuring the impact angle between the FMH and the target location. The impact test measures the value of the Head Injury Criterion (HIC), which is used to measure head injury potential in car crash test. There are various target locations required by FMVSS No.201U on one automotive product. According to FMVSS No.201U, the HIC value measured under specific impact speed (FMH Impact Speed — 24 km/h, Reduced FMH Impact Speed — 19 km/h for vehicles equipped with dynamically deployed upper interior head protection system) shall not exceed 1 000.

While developing a new car model, the designed A-pillar (a car component shown in the *left side* of Figure 6.2) needs to meet the requirement of FMVSS No.201U, i.e., the measured HIC value shall be lower than 1 000. Traditionally, engineer conduct the construction-simulation loop, explained in Figure 6.1a in Section 6.1. Notably, the design data of A-pillar comes often in the form of 2D engineering blueprint, i.e., cross sections from the 3D model, which will be detailed described in next section. According to the given geometric constrains, i.e., structural design from exterior parts that are less relevant to the measured HIC value (shown in the *right side* of Figure 6.2 with dark black lines), engineers are tasked to design the target parts of the A-pillar that are the key structure deciding the measured HIC value (shown in the *right side* of Figure 6.2 with light black lines) and optimize the design until it satisfies the HIC-value requirement in a crush simulation based on the expertise of the designers. However, it is not trial, how to modify the structure so that it performs better in the HIC value while following the geometric constrains (e.g., the given structures).

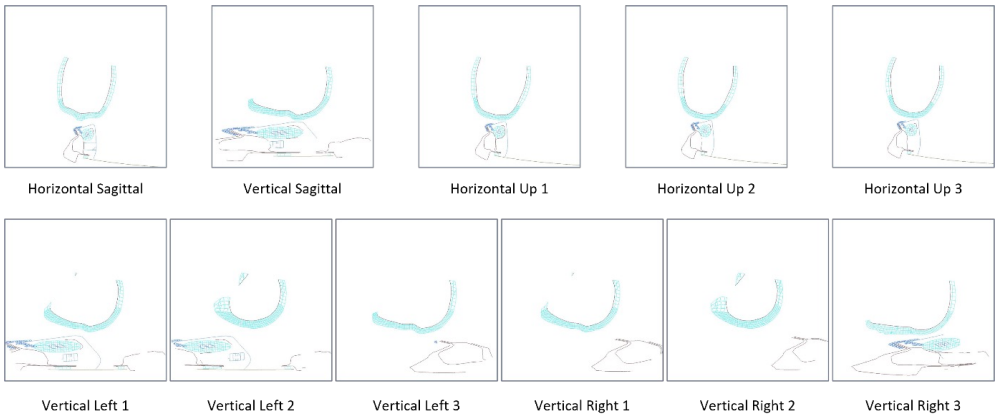


Figure 6.3: One sample of the cross-sectional blueprints of a vehicle A-pillar 3D model.

To this end, we follow the new designed pipeline, DGM-based Generative Engineering Design, as shown in Figure 6.1b. In this proposed pipeline, DGMs is tasked to generate geometric constrain-satisfied designs while the function requirement is met via gradient optimization. Here, the DGM-Simulation loop is designed to be differentiable, so that the gradient can be calculated and used to guide the DGM to synthesize designs that perform better in the target function. However, it has not yet been fully addressed how to use DGM to generate designs that follow the geometric constrains. We aim to tackle this in this section.

6.2.2 Cross-Section Blueprints

According to FMVSS No.201U [171], the HIC is measured for each target locations. These target locations stand for various automotive interior structures. The cross-sectional images, shown in Figure 4.1, are slices, produced by cutting the structure and Free Motion Headform (FMH) together with 11 various planes. These planes all go through the center of the FMH, and the slices are named after the cutting directions, which are Horizontal Sagittal 1, Horizontal Up 1, Horizontal Up 2, Horizontal Up 3, Vertical Sagittal 1, Vertical Left 1, Vertical Left 2, Vertical Left 3, Vertical Right 1, Vertical Right 2, Vertical Right 3. In the cross-sectional images, one sample is produced by slicing one target location of specific car model, and always contains 11 images, Figure 6.3 shows one sample as instance. In the example the curved shape is the sliced part of FMH, while the rest part is the cross-section of the target automotive interior structure. There exist 12 876 blueprints from 3D A-pillars models. We first

convert these blueprints into grayscale pixel-based images with a resolution of 256 × 256.

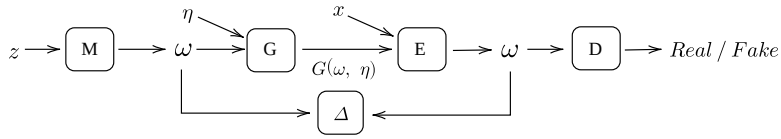
6.2.3 Generation of Cross-Section Designs with GANs

In this work, we first test GANs, dominant the DGM field before diffusion models, as a baseline framework on generating real-world industrial designs. Here, we select Adversarial Latent Autoencoder (ALAE) [126] for its ability of mapping between latent space and image space. We enhance the model robustness and develop an improved version by introducing modern techniques, e.g., spectral normalization (SN) [106], ResNet [55], etc. We further implement the self-attention mechanism [186, 174] to enable the generation of structural patterns. The content of this section has been published in the paper [165].

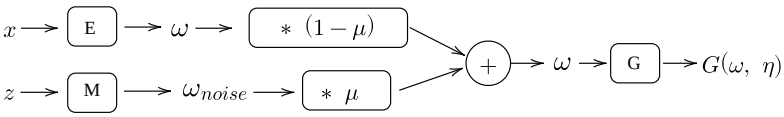
Improved ALAE The vanilla ALAE faces a major issue in its training for the following reasons. First, the ALAE implements the adversarial training strategy in image space, which exposes it to the same training instability commonly encountered by GANs [19, 140, 48]. We implement the following techniques to improve the model robustness and propose a novel version of the ALAE called ALAE$_{improved}$. Firstly, to ensure that generator G receives effective gradients for the optimization, we need to prevent the computed gradients from vanishing or exploding. For this purpose, we apply Spectral Normalization (SN) [106] in encoder E and discriminator D. Spectral Normalization is a powerful method that constrains the Lipschitz continuity of the functions optimized by E and D through layer-wise control of the spectral norm, thereby constraining the scale of the gradients. Spectral Normalization replaces every given weight matrix W by W_{sn} with the formula:

$$W_{\text{sn}} = \frac{W}{\sigma(W)}, \tag{6.1}$$

where $\sigma(W)$ is the greatest singular value of W. Moreover, to stabilize the adversarial training, we implement batch normalization in generator G, allowing for a larger range of learning rates and learning the data distribution more efficiently. Lastly, to address the degradation caused by increasing the depth of the neural network, we modify generator G and encoder E of ALAE to become Residual Networks (ResNet). ResNet features in the so-called residual blocks of convolutional operators contain a "skip connection" that can bypass the error information in back-propagation, hence alleviating the vanishing gradient problem in deeper architectures.



(a) Training



(b) Sampling

Figure 6.4: SA-ALAE in directed graphs. ALAE and SA-ALAE share an identical training strategy. The sampling procedure of SA-ALAE utilizes the off-the-shelf mapper M to sample additional noisy latent variables ω_{noise}, hereby introducing randomness in the sampling.

Self-Attention Adversarial Latent Autoencoder (SA-ALAE) Convolution-based DGMs generate high-quality real-world images. However, DGMs have difficulty in modeling structural objects in engineering design images. The main cause is possibly that the objects in structural images can be sparsely located but are still strongly geometrically connected, e.g., in bicycle design, both wheels should be connected by an axle and lie on the ground. In this simple example, the pixels describing each wheel depend on one another but are far away in the image. However, as a local operator on images, the convolution mechanism cannot capture long-range dependencies, suggesting that using convolutional layers alone is unsuitable for generating engineering design images. Self-Attention Generative Adversarial Network (SAGAN) [197] leverages the self-attention mechanism to gain a global understanding of the input features, which enables the model to capture and model long-range dependencies efficiently. Following the success of SAGAN, we introduce the same mechanism respectively into generator G and encoder E in the ALAE$_{improved}$ framework and propose a new model, Self-Attention Adversarial Latent Autoencoder (SA-ALAE) [165].

To achieve the stochastic generation of a novel image from a source image in the vanilla ALAE, the latent variables ω encoded from the original design are perturbed with sampled noise $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We note that this approach compromises the quality of the encoded latent variables, which generates blurry images. To avoid this, SA-ALAE leverages the trained map M to sample random latent variables as a noisy



(a) Source blueprints (A-pillar)



(b) Blueprints reconstructed by SA-ALAE



(c) Blueprints randomly generated by SA-ALAE

Figure 6.5: Examples of source blueprints of A-pillar and blueprints generated by SA-ALAE.

perturbation. The novel sampling procedure of SA-ALAE is displayed in Figure 6.4b. Adding randomly sampled latent variables as an independent noise protects the quality of perturbed latent variables so that the stochastic encoder-decoder inference does not harm the quality of the generated image. The calculation of the latent variables, used to explore the variants of existing designs, is based on the following formula:

$$\boldsymbol{\omega} = (1 - \mu)E(\mathbf{x}) + \mu M(\mathbf{z}), \tag{6.2}$$

where $\mu \in [0, 1]$ is a tunable parameter, $\mathbf{x}$ refers to an original design and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is sampled noise.

Results of SA-ALAE We evaluate the SA-ALAE in both image reconstruction and random generation on the A-pillar dataset. As shown in Figure 6.5, SA-ALAE generates high-quality blueprints corresponding to the input designs and containing recognizable structural details. In addition to the reconstruction function, we evaluate the ability of SA-ALAE in randomly generating A-pillar designs, which yields a

sufficient diversity of novel designs with detailed structures displayed in Figure 6.5. While the results of SA-ALAE is compelling, training of SA-ALAE is still challenging due to the instability of the adversarial training. To bypass this, in next section, we will focus on using diffusion models instead to generate A-pillar designs.

6.2.4 A-pillar Blueprint Autocompletion

As we describe in Section 6.2.1, the engineers responsible for the design of the interior parts of the A-pillar initially receive a predefined structure (i.e., the given parts) and then complete the design by constructing the remaining structural parts (i.e., the target parts), such as the airbag and interior shell, which we drew using light black lines in the right side in Figure 6.2.



(a) A-pillar exterior structures (the given parts)



(b) DGM-completed A-pillar designs



(c) Comparison between engineer- and DGM-completed A-pillar designs

Figure 6.6: Example of automatic blueprint completion. In (c), we have overlaid the engineer's completed design with the DGM's completed design to show that our DGM performs a conditional generative design process, the results of which follow the given geometric constraints well but is clearly different from the blueprint designed by engineers.

The introduction of inpainting technique (i.e., RePaint [97]) with PoDM (developed in Chapter 2) on bicycle designs has shown compelling results in Figure 2.14b. Consid-

ering that the completion of 2D A-pillar cross-section blueprints is a similar scenario, we test the PoDM-driven inpainting in designing A-pillar cross-section blueprints in this section. We train a PoDM on the cross-section images using the configurations defined in Section 2.3.1. During model training, we use FDD (developed in Chapter 3) to select the best-performing model. Afterwards, we implement the inpainting pipeline (explained in Section 2.4) on the given structure of exterior A-pillar design, shown in Figure 6.6a. The DGM-completed design structures are displayed in Figure 6.6b, while we also show the engineer-design A-pillar blueprints in Figure 6.6c. This shows that our PoDM does not just copy the designs of engineers, but explores new solutions for a given structure — a sign of good generative engineering design, the results of which are still feasible and usable and has been approved by engineers in expert reviews.

6.3 Vehicle Rim Design

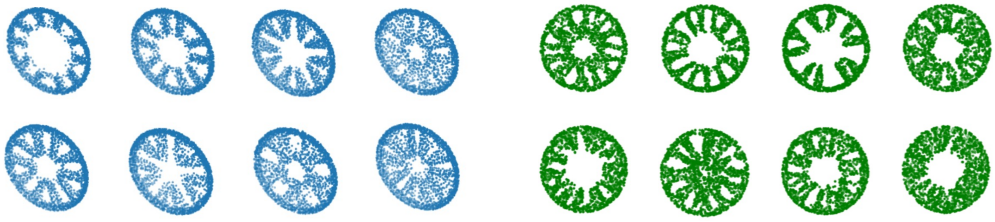
The design of car rims is a crucial aspect of the overall vehicle design process, as it not only contributes to the aesthetic appeal of the car but also plays a significant role in its aerodynamic performance. The rim design data in the car industry is closely tied to the mesh form, for its compatibility with aerodynamic simulations.

The shape and design of the car's rims have a significant impact on the overall aerodynamic performance of the vehicle. The rim's design can affect the airflow around the wheels, which in turn can influence the car's fuel efficiency, stability, and handling characteristics [146].

Designers and engineers in the car industry use Computational Fluid Dynamics (CFD) simulations to analyze the airflow patterns around the rims and the rest of the vehicle. By analyzing the mesh data form of the rims, they can identify areas of high pressure, turbulence, and drag, and then refine the design to optimize the aerodynamic performance. We show a basic pressure measurement example from [13] that demonstrates how the geometric design of the wheel rim affects the pressure distribution in the car.

To this end, we claim that DGM-based engineering can accelerate the rim design process and has great potential, but there is a major challenge: DGMs used for 3D shapes recently have focused on point clouds, generating results that are neither recognizable to engineers nor usable for simulation. We tried latent Wasserstein GAN (LWGAN) [131] and point-voxel diffusion (PVD) [201], details can be found in Section 1.4, and demonstrate the generated results in Figure 6.7. As the example shows,

the resulting point clouds are of poor quality and do not show the clear structures that engineers can construct after. In addition, we have consulted with engineers, and they suggest that even the source point clouds (Figure 6.8c) cannot represent the design in the simulation and give accurate results.



(a) Rims generated with LWGAN [131] **(b)** Rims generated with PVD [201]

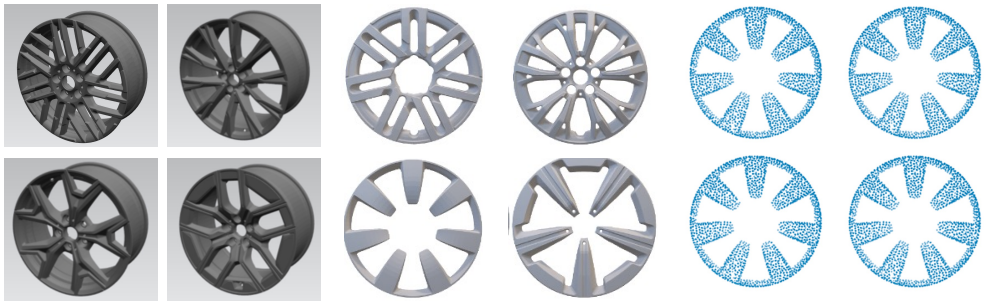
Figure 6.7: Rims generated by state-of-the-art point-based DGMs.

6.3.1 Rim Design Data

For the study in this section, we have 497 samples of BMW rims, which are designed in real engineering processes with multiple objectives, e.g., outstanding aerodynamic performance, low cost and excellent appearance. This dataset consists of 497 high-quality 3D models of rims, providing a representative example of industrial 3D surfaces that demand precision and efficiency. Source data is the in the form of STEP files, as we display in Figure 6.8a, where the design relevant part is only the front part of the whole rim. Thus, we trim the source rim data, so that the design can be represented with a much reduced model size, as shown in Figure 6.8b which are in the form of mesh. In developing DGM-based models for rim design, we also convert the rims to the form of point cloud (see Figure 6.8c), so that we can test state-of-the-art point-based generative models.

6.3.2 Design Inspiration

In this section, we implement the method (SpoDify) we developed in Chapter 4 for design inspiration of industrial rims. Note that, the advantage of our method is not only the compelling generation of realistic mesh shapes, but also the efficiency in reducing the dimension of source data (a rim mesh can have 15k vertices, and using SpoDify can reduce each rim mesh to 64 values) and saving computational power. In Figure 6.9b, we display the generated rims and the source rims for visual qualitative

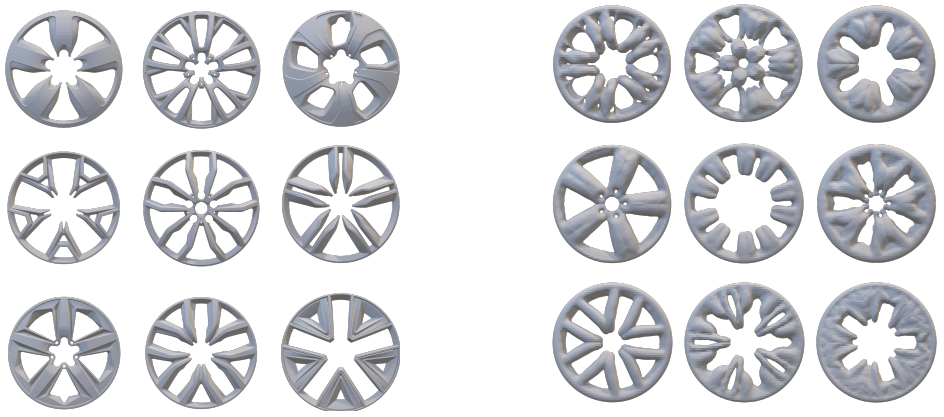


(a) Source rims without trimming **(b)** Source rims after trimming **(c)** Source trimmed rims represented in point clouds

Figure 6.8: Examples of BMW rims in various form, i.e., B-Rep, mesh, and point cloud.

comparison. In Figure 6.10, we additionally display the generated rims and their closest samples from source data to show that our method is able to generate novel shapes.

Here, the generated rims maintain structural integrity and meaningful design elements, showcasing that the model is able to synthesize good quality rim designs by training on source data. Additionally, we observe many novel rim designs that do not come from the source dataset, which can help exploration in the design process. In conclusion, this capacity to interpolate and generate novel shapes inspired by the original dataset is a promising indicator of the model's robustness and utility.



(a) Source rims **(b)** Rims generated by SpoDify

Figure 6.9: Comparison of generated rim data: (a) source data, (b) rims generated by our approach.

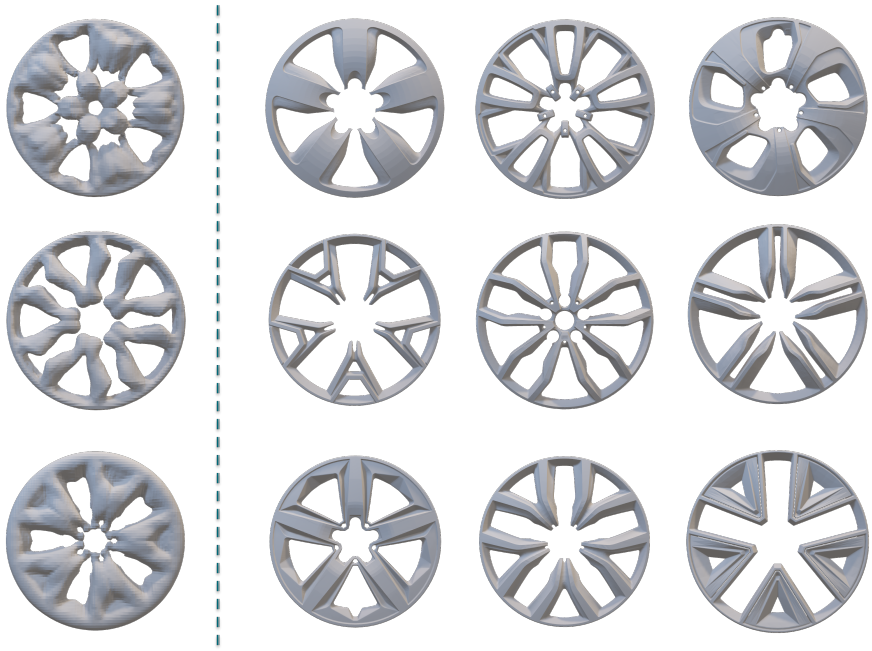


Figure 6.10: Generated rims (*left*) and their top 3 nearest meshes (*right*) from the source data. Determining the nearest mesh leverages Chamfer distance as the metric.

The idea behind applying spectral decomposition, as previously motivated, is to represent an object as a combination of basic shapes in the spectral space. These basic shapes act as the primary features that capture and assemble the structure of the dataset, grouping and describing it in terms of these key characteristics. In Figure 6.11, we visualize the right eigenvector V^T matrix. This is achieved by taking an eigenvector, applying the inverse discrete wavelet transform (IDWT), and then using marching cubes to map it back to the mesh domain. Here we highlight the data-dependent nature of these basic shapes. The rim dataset shows clear and distinct patterns in the basic shapes, making it easier to interpret the meaning of each eigenvector. This visualization process provided valuable insights into the underlying structures of the datasets.

6.4 Conclusion

In practical industrial design processes, designs are more complex and higher-dimensional, which makes training and reasoning difficult. Our work introduces two scenarios, de-

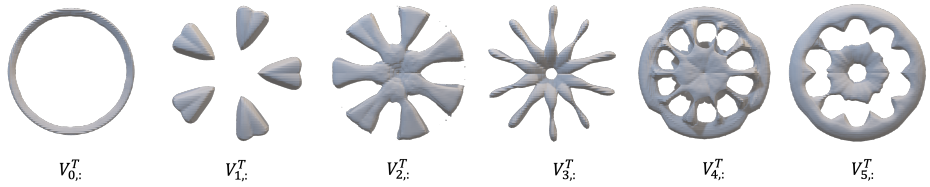


Figure 6.11: Visualization of the spectral features. The spectral bases present in the $V^\top$ matrix resulted from the SVD.

sign completion and design inspiration, to demonstrate how the Design Guidance Model (DGM) can assist engineers in developing complex practical designs within a carefully designed framework. First, completing a preliminary design based on given geometric requirements is a common task that requires some knowledge of the target design component and can be quite time-consuming. In situations where a large amount of historical design data is available, our work in this section demonstrates how DGM can facilitate the design process in a reliable manner. Secondly, generating designs that do not exist for design inspiration is a very valuable function for design teams. We apply our SpoDify developed in Chapter 4 to the wheel rim and obtained compelling designs that are not existing in real wheel rim data.

Chapter 7

Conclusion and Outlook

This thesis aims to bridge the gap between modern deep generative models and generative engineering design. This chapter summarizes all the content of the preceding chapters, responds to the main research questions raised in Chapter 1, and points out limitations and future directions.

7.1 Research Question Revisited

Chapter 1: In the introduction, this paper provides an overview for this study, including industrial design processes, generative engineering design, and the latest advances in deep generative modeling, and explains the motivation for using deep generative models to achieve generative engineering design. The main research question addressed in this paper is:

How to enable deep generative models to synthesize engineering designs?

To answer this question, we have broken it down into several sub-questions, which are discussed separately in different chapters. More precisely, this paper addresses the main question by tackling a series of challenges, namely data representation, model architecture, model training/sampling, and evaluation. A secondary objective is to apply the newly developed methods to industrial design processes and demonstrate their applicability.

Chapter 2: Although recent diffusion models have achieved impressive results on general images, their generated engineering blueprints are structurally unrecognizable,

i.e., poor plausibility. We assume that this can be caused by their prioritizing strategy for better visual quality. Herein, the challenge is:

How to prioritize plausibility in generation with deep generative models?

We observe that there is a range of noise levels associated with structure formation. Therefore, we design a new noise scheduling scheme for the diffusion model for training and sampling to focus on this selected noise level range. Our evaluation results show that this helps improve the rationality of the generated results. This proposed noise scheduling method is called plausibility-oriented diffusion model (PoDM).

Chapter 3: In the previous chapter, we discover the issue of poor plausibility in the generated designs, which are evaluated manually. This raise the following research question:

How to automatically evaluate the plausibility of designs generated by deep generative models?

Leveraging the advantage of denoising autoencoders in focusing on capturing underlying structures during training, we propose Fréchet Denoised Distance (FDD) as a new evaluation metric, which shows reliable performance in aligning with human judge on plausibility.

Chapter 4: High dependence on training autoencoders to enable deep generative models on high-dimensional data leads to significant computational costs and imperfect decoding results. Thus, the research question is:

How to use deep generative models to generate high-dimension designs more efficiently?

Several learning-free decomposition algorithms can achieve the same encoding-decoding function with adjustable information loss, such as singular value decomposition (SVD). Inspired by this, we design an SVD-based generation framework, SpoDify, which achieves performance comparable to state-of-the-art methods without training an autoencoder.

Chapter 5: Given that CAD is the most widely used data format in engineering design processes, enabling DGM for generative engineering design must address the

following research question:

How to enable deep generative models to directly synthesize CAD-native representation?

Existing work has directly generated B-Rep data (native CAD representation), but this progress relies on learning on structured point clouds sampled from geometries. Here, we address this limitation by proposing NeuroNURBS, a method that decomposes NURBS-based geometries into latent features that can be used for downstream tasks. This method demonstrates significant efficiency advantages in terms of storage, computation and speed.

Chapter 6: Industrial design encompasses many complex structures and knowledge. When using generative models to simulate engineer construction, our major concern is:

How can DGMs be beneficial for real-world industrial design applications?

Here, we use the car A-pillars and rims as examples to demonstrate how the methods we developed (SA-ALAE, PoDM, and SpoDify) can assist engineers in designing complex structures at certain stages of the design process. Through the attempts in Chapter 6, our research provides insights into the application of DGMs to real-world engineering design processes, thereby answering the core research question of this thesis.

7.2 Limitations and Future Work

While DGMs develop in a swift speed, applying the technologies for generative engineering design still faces several challenges. We list the limitations and future directions below, for more specific discussions, we refer to the conclusion sections of corresponding chapters.

Available dataset. Currently, despite the publication of numerous engineering design datasets, e.g., 2D design images (BIKED [137], Seeing3DChairs [7], etc.), sequential construction operations in sequence (DeepCAD [182], Fusion 360 Gallery [180], etc.), 3D B-Reps (ABC [76], DeepCAD in B-Rep [187], etc.), the existing open-source datasets for generative design are insufficient. More specifically, in industrial design

scenarios, design datasets often exhibit more complex structures (e.g., B-Rep with over 3 000 NURBS surfaces) but less in diversity. In this context, the challenge of achieving DGM shifts from generating diverse simple designs to effectively learning the complex data distribution of a single target. We note that the most advanced DGM currently used for B-Rep generation has been evaluated as having excellent performance, but the entities it generates have been deemed to be unreasonable designs. Here, we ask whether this problem is caused by poor DGM performance or excessive diversity in the training data set. Additionally, when we are developing the NeuroNURBS framework, these datasets cannot provide sufficient high-degree NURBS surfaces to challenge the model. Therefore, the potential of NeuroNURBS is underestimated. Considering that during the historical design phase, a significant amount of designs were created in a verifiable digital format, the release of new datasets containing these historical designs may be helpful in further advancing this field.

Generating feasible designs. Our work on PoDM is just a small step towards understanding the structure generated by DGM. We refer to the performance of generating reasonable designs as "plausibility". However, in actual industrial design scenarios, engineering design needs to present feasible, manufacturable structures. In order for a design to be feasible and manufacturable, it must meet a large number of critical requirements, such as ease of assembly, sufficient strength, compliance with safety standards, etc. Currently, research in this field still struggles to explain why and how DGM generates realistic samples. To address this, we believe that the future direction is to iteratively generate designs, evaluate feasibility and design performance at each step, and feed the evaluation results back into the next iteration. On this, we have presented our insights in Section 2.4.

Conditional design generation. This thesis evaluates proposed models often under unconditional generation, as we aim to improve the generation efficiency and to enable the generation of special design representations for research purposes. However, in real-world industrial design processes, conditional design generation is required. Recent studies [88, 92] have achieved the generation of desired designs under specific condition (e.g., text descriptions, sketches, or mesh inputs), but industrial users prefer more intuitive inputs (e.g., dragging), which we have demonstrated in Section 2.4 but only for 2D images. To address this limitation, we believe that the future direction is to enable more intuitive input and evaluate the accuracy of the editing.

Design representation. This work begins with two-dimensional blueprints and delved into three-dimensional meshes and B-Reps, fully exploring the representational possibilities of design. However, to use DGM in real-world design cases, the current representation types are far from sufficient. More specifically, representing B-Reps with hierarchical structures leads to unnecessary large data volumes and poses challenges for the reversal process. Although recent work has achieved promising results [187], this framework suffers from high inefficiency during generation and poor performance when generating complex structures with thousands of surfaces. To address this issue, we believe that developing new representation methods or new concept of how B-Reps are formed, such as introducing interactions and combination functions between solids to form B-Rep volumes, may be helpful.

7.3 The Future of DGMs in Generative Engineering Design.

The limitations mentioned previously expose a significant gap but also reveal the enormous potential for growth in the field of DGM-based generative design. Our work is completed during a period when generative models began to shine, with researchers in the field continuously surpassing previous methods. It may take only a short time for the future directions discussed in this section to be surpassed, but we will still offer our insights.

Firstly, due to the development of large language models (LLMs), current LLMs can operate on a high diversity of data by treating it as "language", such as code, tabular data, and even images, referred to as large multimodal models (LMMs). In fact, in historical industrial design processes, there is a rich repository of design data in various forms, e.g., code (JT and STEP files), engineering construction operations in sequence (from CADPART files), blueprint images (DWG files), or tabular data, which can be understood by LMMs according to recent research [185, 42, 9]. However, this concept remains in its early stages.

At the same time, there is great potential for research into using DGMs to directly learn from design data, such as B-Reps, as this enables a differentiable modeling process, which is useful for further applications such as numerical simulation. We have already provided this concept in Section 2.4. Most of the work in this thesis has been done in this direction, but it seems that there is still a long way to go. To enable more effective modeling of design data like B-Reps, DGMs need to access the natural pro-

cesses of construction, including but not limited to sketching, extruding, chamfering, drilling, and interaction. We believe that these construction operations are based on engineering expertise, and it is essential for DGMs to learn this knowledge in addition to structural information. It should be noted that representing these operations in text form does not necessarily enable the DGM to understand the construction knowledge and structural dependencies. DGMs also need to consider the changes these operations make to the structure and be penalized when constructing any infeasible entities. However, how to achieve this and how to represent it remains an open question.

Moreover, the current main trend in generative modeling applications is to deepen models exponentially. This trend has led to the emergence of giant, even mega models. Designers have observed that simply increasing the size of the model can effectively force it to learn hidden information better. However, we hope this does not become the future direction of research, as recent DGM frameworks have not yet been fully developed. Researchers in this field should strive forward to bring more advancements to DGM frameworks to address remaining challenges, such as low-sample learning, out-of-distribution learning or machine reasoning.

Finally, incorporating industrial applications into consideration is an inevitable future direction for DGM research. In our work, we have observed that the development of DGMs for industrial purposes has not garnered much attention in the DGM community, while most recent work has focused on more general data representations such as point clouds, meshes, and implicit representations like neural radiative fields (NeRFs). This focus has led to DGM research often attracting widespread attention but struggling to find corresponding application cases in the real world. Meanwhile, generating engineering designs is a valuable application scenario where DGMs can make significant contributions. By bridging the gap between theoretical advancements and practical applications, we can unlock the full potential of DGMs, paving the way for innovative solutions that enhance efficiency and creativity in industrial applications.

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.

[3] Md Ferdous Alam, Austin Lentsch, Nomi Yu, Sylvia Barmack, Suhin Kim, Daron Acemoglu, John Hart, Simon Johnson, and Faez Ahmed. From Automation to Augmentation: Redefining Engineering Design and Manufacturing in the Age of NextGen-AI. *An MIT Exploration of Generative AI*, mar 27 2024. https://mit-genai.pubpub.org/pub/9s6690gd.

[4] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[5] Silvia Ansaldi, Leila De Floriani, and Bianca Falcidieno. Geometric modeling of solid objects by using a face adjacency graph representation. *SIGGRAPH Comput. Graph.*, 19(3):131–139, July 1985.

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[7] Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic. Seeing 3d chairs: Exemplar part-based 2d-3d alignment using a large dataset of cad models. In *2014 IEEE CVPR*, pages 3762–3769, 2014.

[8] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. MultiMAE: Multi-modal multi-task masked autoencoders. *European Conference on Computer Vision*, 2022.

[9] Akshay Badagabettu, Sai Sravan Yarlagadda, and Amir Barati Farimani. Query2cad: Generating cad models using natural language queries, 2024.

[10] Nicholas Baker, Hongjing Lu, Gennady Erlikhman, and Philip J. Kellman. Deep convolutional networks do not classify based on global object shape. *PLoS Computational Biology*, 14, 2018.

[11] Shane T. Barratt and Rishi Sharma. A note on the inception score. *ArXiv*, abs/1801.01973, 2018.

[12] Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning*, pages 226–234. PMLR, 2014.

[13] Henrik Berg and Adam Brandt. Investigation of aerodynamic wheel design. 2018.

[14] Eyal Betzalel, Coby Penso, Aviv Navon, and Ethan Fetaya. A study on the evaluation of generative models. *CoRR*, abs/2206.10935, 2022.

[15] Gregory Beylkin, Ronald Coifman, and Vladimir Rokhlin. Fast wavelet transforms and numerical algorithms i. *Communications on pure and applied mathematics*, 44(2):141–183, 1991.

[16] Mikolaj Binkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD gans. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[17] Andrii Blyndaruk and Olena Shapovalova. End-to-end differentiable nurbs–geom-gnn–tgn pipeline for spatiotemporal identification of moving objects. In *Scientific and Practical Journal" Materials of Scientific Conferences of the Petro Mohyla Black Sea National University"*, number 1, pages 16–19, 2025.

[18] Ali Borji. Pros and cons of gan evaluation measures: New developments. *Computer Vision and Image Understanding*, 215:103329, 2022.

[19] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[21] Lucas F Buzuti and Carlos E Thomaz. Fréchet autoencoder distance: A new approach for evaluation of generative adversarial networks. *Computer Vision and Image Understanding*, 235:103768, 2023.

[22] Daniel R Canelhas, Erik Schaffernicht, Todor Stoyanov, Achim J Lilienthal, and Andrew J Davison. An eigenshapes approach to compressed signed distance fields and their utility in robot mapping. *arXiv preprint arXiv:1609.02462*, 2016.

[23] Weijuan Cao, Trevor T Robinson, Yang Hua, Andrew Colligan, and Wanbin Pan. Graph representation of 3d cad models for machining feature recognition with deep learning. In *ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference: Proceedings*, November 2020. International Design Engineering Technical Conferences & Computers and Information in Engineering Conference ; Conference date: 17-08-2020 Through 19-08-2020.

[24] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[25] Amaresh Chakrabarti, Kristina Shea, Robert Stone, Jonathan Cagan, Matthew Campbell, Noe Vargas Hernandez, and Kristin L. Wood. Computer-Based Design Synthesis Research: An Overview. *Journal of Computing and Information Science in Engineering*, 11(2):021003, June 2011. _eprint: https://asmedigitalcollection.asme.org/computingengineering/article-pdf/11/2/021003/5566343/021003_1.pdf.

[26] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015.

[27] Wei Chen and Faez Ahmed. Padgan: Learning to generate high-quality novel designs. *Journal of Mechanical Design*, 143:031703, 11 2020.

[28] Wei Chen, Kevin Chiu, and Mark D Fuge. Airfoil design parameterization and optimization using bézier generative adversarial networks. *AIAA Journal*, 58(11):4723–4735, 2020.

[29] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5939–5948, 2019.

[30] Shiyang Cheng, Michael Bronstein, Yuxiang Zhou, Irene Kotsia, Maja Pantic, and Stefanos Zafeiriou. Meshgan: Non-linear 3d morphable models of faces. *arXiv preprint arXiv:1903.10384*, 2019.

[31] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8185–8194, 2020.

[32] Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2262–2272, 2023.

[33] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer, 2016.

[34] CIMdata. Generative design: What's that? https://www.cimdata.com/en/news/item/8402-generative-design-what-s-that, April 2023. Accessed: 2023-04-24.

[35] Ronald R Coifman, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the national academy of sciences*, 102(21):7426–7431, 2005.

[36] Andrew R. Colligan, Trevor T. Robinson, Declan C. Nolan, Yang Hua, and Weijuan Cao. Hierarchical cadnet: Learning from b-reps for machining feature recognition. *Computer-Aided Design*, 147:103226, 2022.

[37] Corinna Cortes and Vladimir Vapnik. *Support-vector networks*. Number 3. Springer, 1995.

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[39] Anjana Deva Prasad, Aditya Balu, Harshil Shah, Soumik Sarkar, Chinmay Hegde, and Adarsh Krishnamurthy. Nurbs-diff: A differentiable programming module for nurbs. *Comput. Aided Des.*, 146(C), May 2022.

[40] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[41] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[42] Yuhao Du, Shunian Chen, Wenbo Zan, Peizhao Li, Mingxuan Wang, Dingjie Song, Bo Li, Yan Hu, and Benyou Wang. Blenderllm: Training large language models for computer-aided design with self-improvement, 2024.

[43] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[44] G Farin. Handbook of computer aided geometric design. *Elsevier science google schola*, 2:577–580, 2002.

[45] Robert L Foote. Regularity of the distance function. *Proceedings of the American Mathematical Society*, 92(1):153–155, 1984.

[46] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *ArXiv*, abs/1811.12231, 2018.

[47] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[48] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[49] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[50] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[51] Jana Gulanová, Ladislav Gulan, Michal Forrai, and Mario Hirz. Generative engineering design methodology used for the development of surface-based components. *Computer-Aided Design and Applications*, 14(5):642–649, 2017.

[52] Florentin Guth, Simon Coste, Valentin De Bortoli, and Stephane Mallat. Wavelet score-based generative modeling, 2022.

[53] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):90:1–90:12, 2019.

[54] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[56] Negar Heidari and Alexandros Iosifidis. Geometric deep learning for computer-aided design: A survey. *arXiv preprint arXiv:2402.17695*, 2024.

[57] Katherine L. Hermann, Ting Chen, and Simon Kornblith. The origins and prevalence of texture bias in convolutional neural networks. *arXiv: Computer Vision and Pattern Recognition*, 2019.

[58] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2018.

[59] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[60] Mario Hirz, Patrick Rossbacher, and Jana Gulanová. Future trends in cad–from the perspective of automotive industry. *Computer-Aided Design and Applications*, 14(6):734–741, 2017.

[61] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020.

[62] Danijela Horak, Simiao Yu, and Gholamreza Salimi Khorshidi. Topology distance: A topology-based approach for evaluating generative adversarial networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 7721–7728. AAAI Press, 2021.

[63] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. Ak Peters Series. Taylor & Francis, 1996.

[64] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[65] Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[66] Pradeep Kumar Jayaraman, Joseph George Lambourne, Nishkrit Desai, Karl Willis, Aditya Sanghi, and Nigel J. W. Morris. Solidgen: An autoregressive model for direct b-rep synthesis. *Transactions on Machine Learning Research*, 2023. Featured Certification.

[67] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D.D. Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[68] Mark W Jones. Distance field compression. *The Journal of WSCG*, 12(2):199–204, 2004.

[69] Nikolai Kalischek, Torben Peters, Jan D Wegner, and Konrad Schindler. Tetradiffusion: Tetrahedral diffusion models for 3d shape generation. *arXiv e-prints*, pages arXiv–2211, 2022.

[70] Tero Karras. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[71] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

[72] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[73] Mohamed Kas, Abderrazak Chahi, Ibrahim Kajo, and Yassine Ruichek. Eigengan: An svd subspace-based learning for image generation using conditional gan. *Knowledge-Based Systems*, 293:111691, 2024.

[74] Katherine Keegan, Tanvi Vishwanath, and Yihua Xu. A tensor svd-based classification algorithm applied to fmri data. *arXiv preprint arXiv:2111.00587*, 2021.

[75] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[76] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[77] B. Kolarevic. *Architecture in the Digital Age: Design and Manufacturing*. Taylor & Francis, 2004.

[78] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

[79] Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

[80] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5(4):1, 2010.

[81] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[82] Sarah C. Kucker, Larissa K. Samuelson, Lynn K. Perry, Hanako Yoshida, Eliana Colunga, Megan G Lorenz, and Linda B. Smith. Reproducibility and a unifying explanation: Lessons from the shape bias. *Infant behavior & development*, 54:156–165, 2019.

[83] Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of imagenet classes in fréchet inception distance. In *The Eleventh International Conference on Learning Representations*, 2023.

[84] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Neural Information Processing Systems*, 2019.

[85] Joseph George Lambourne, Karl Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshan. Reconstructing editable prismatic cad from rounded voxel models. In *SIGGRAPH Asia 2022 Conference Papers*, SA '22, page 1–9. ACM, November 2022.

[86] Barbara Landau, Linda B. Smith, and Susan Scanlon Jones. The importance of shape in early lexical learning. *Cognitive Development*, 3:299–321, 1988.

[87] Sang Hun Lee and Kunwoo Lee. Partial entity structure: a compact non-manifold boundary representation based on partial topological entities. In *International Conference on Smart Media and Applications*, 2001.

[88] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Sketch2cad: Sequential cad modeling by sketching in context. *ACM Trans. Graph. (Proceedings of SIGGRAPH Asia 2020)*, 39(6):164:1–164:14, 2020.

[89] Jiahao Li, Yunpeng Bai, Yongkang Dai, Hao Guo, Hongping Gan, and Yilei Shi. Autoregressive generation with b-rep holistic token sequence representation. *arXiv preprint arXiv:2601.16771*, 2026.

[90] Muheng Li, Yueqi Duan, Jie Zhou, and Jiwen Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12642–12651, 2023.

[91] Wenqian Liu, Runze Li, Meng Zheng, Srikrishna Karanam, Ziyan Wu, Bir Bhanu, Richard J Radke, and Octavia Camps. Towards visually explaining variational autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8642–8651, 2020.

[92] Yilin Liu, Duoteng Xu, Xingyao Yu, Xiang Xu, Daniel Cohen-Or, Hao Zhang, and Hui Huang. Hola: B-rep generation using a holistic latent representation, 2025.

[93] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in neural information processing systems*, 32, 2019.

[94] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[95] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[96] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

[97] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.

[98] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2837–2845, 2021.

[99] Antoine Maiorca, Youngwoo Yoon, and Thierry Dutoit. Evaluating the quality of a synthesized motion with the fréchet motion distance. In *ACM SIGGRAPH 2022 Posters*, SIGGRAPH '22, New York, NY, USA, 2022. Association for Computing Machinery.

[100] Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. Interacting particle solutions of fokker-planck equations through gradient-log-density estimation. *Entropy*, 22(8):802, 2020.

[101] Peter Meltzer, Hooman Shayani, Amir Khasahmadi, Pradeep Kumar Jayaraman, Aditya Sanghi, and Joseph Lambourne. Uvstyle-net: Unsupervised few-shot learning of 3d style similarity measure for b-reps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9690–9699, 2021.

[102] Victoria Miles, Stefano Giani, and Oliver Vogt. Approaching step file analysis as a language processing task: A robust and scale-invariant solution for machining feature recognition. *Journal of Computational and Applied Mathematics*, 427:115166, 2023.

[103] Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59, 1960.

[104] Mehdi Mirza. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[105] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 306–315, 2022.

[106] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[107] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[108] Sergey Moiseev, Andrey Kuznetsov, Viktor Gamayunov, Anton Ezhov, Sergey Serebryakov, and Sergey Kolesnikov. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[109] Jens-Dominik Mueller, Xingchen Zhang, Siamak Akbarzadeh, and Yang Wang. Geometric continuity constraints of automatically derived parametrisations in cad-based shape optimisation. *International Journal of Computational Fluid Dynamics*, 33:1–17, 11 2019.

[110] Phillip Mueller, Talip Uenlue, Sebastian Schmidt, Marcel Kollovieh, **Jiajie Fan**, Stephan Günnemann, and Lars Mikelsons. Geodiffusion: A training-free framework for accurate 3d geometric conditioning in image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6374–6384, October 2025.

[111] Orest Mykhaskiv, Mladen Banovic, Salvatore Auriemma, Pavanakumar Mohanamuraly, Andrea Walther, Herve Legrand, and Jens-Dominik Mueller. Nurbs-based and parametric-based shape optimization with differentiated cad kernel. *Computer-Aided Design and Applications*, 15:1–11, 04 2018.

[112] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7176–7185. PMLR, 13–18 Jul 2020.

[113] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[114] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[115] Sangeun Oh, Yongsu Jung, Seongsin Kim, Ikjin Lee, and Namwoo Kang. Deep Generative Design: Integration of Topology Optimization and Generative Models. *Journal of Mechanical Design*, 141(11):111405, September 2019. _eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/141/11/111405/6578473/md_141_11_111405.pdf.

[116] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[117] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[118] OpenAI. Chatgpt. https://www.openai.com/chatgpt/, 2022. Accessed: [date accessed].

[119] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[120] Edouard Oyallon, Stéphane Mallat, and Laurent Sifre. Generic deep networks with wavelet scattering, 2014.

[121] Emanuele Palumbo, Imant Daunhawer, and Julia E Vogt. MMVAE+: Enhancing the generative quality of multimodal VAEs without compromises. In *The Eleventh International Conference on Learning Representations*, 2023.

[122] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your GAN: interactive point-based manipulation on the generative image manifold. In Erik Brunvand, Alla Sheffer, and Michael Wimmer, editors, *ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023*, pages 78:1–78:11. ACM, 2023.

[123] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[124] T. Paviot. pythonocc, 2022.

[125] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[126] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [to appear].

[127] L. Piegl. On nurbs: a survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, 1991.

[128] L. Piegl and W. Tiller. *The NURBS Book*. Monographs in Visual Communication. Springer Berlin Heidelberg, 1996.

[129] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[130] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[131] Yixuan Qiu, Qingyi Gao, and Xiao Wang. Adaptive learning of the latent space of wasserstein generative adversarial networks. *Journal of the American Statistical Association*, pages 1–13, 2024.

[132] J. Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[133] Alec Radford. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[134] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[135] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European conference on computer vision (ECCV)*, pages 704–720, 2018.

[136] J Redmon. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[137] Lyle Regenwetter, Brent Curry, and Faez Ahmed. BIKED: A Dataset for Computational Bicycle Design With Machine Learning Benchmarks. *Journal of Mechanical Design*, 144(3), 10 2021. 031706.

[138] Lyle Regenwetter, Amin Heyrani Nobari, and Faez Ahmed. Deep generative models in engineering design: A review. *Journal of Mechanical Design*, 144(7):071704, 2022.

[139] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[140] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. *Advances in neural information processing systems*, 30, 2017.

[141] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lučić, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[142] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR, 2009.

[143] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *ArXiv*, abs/1606.03498, 2016.

[144] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[145] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation, 2023.

[146] Thomas Schütz. *Hucho-Aerodynamik des Automobils: Strömungsmechanik, Wärmetechnik, Fahrdynamik, Komfort*. Springer-Verlag, 2013.

[147] Adam Sebestyen, Albert Wiltsche, Milena Stavric, and Ozan Özdenizci. From nurbs to neural networks: Efficient geometry encoding for generative ai in architectural design. *International Journal of Architectural Computing*, 23(3):720–741, 2025.

[148] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[149] S. S. SHAPIRO and M. B. WILK. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611, December 1965. _eprint: https://academic.oup.com/biomet/article-pdf/52/3-4/591/962907/52-3-4-591.pdf.

[150] Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. *arXiv preprint arXiv:2306.14435*, 2023.

[151] Jaehyeok Shim, Changwoo Kang, and Kyungdon Joo. Diffusion-based signed distance fields for 3d shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20887–20897, June 2023.

[152] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. *arXiv preprint arXiv:2311.15475*, 2023.

[153] Wojciech Skarka. Application of moka methodology in generative model creation using catia. *Engineering Applications of Artificial Intelligence*, 20(5):677–690, 2007.

[154] Colin Smith. On vertex-vertex meshes and their use in geometric and biological modeling. *Retrieved March*, 13:2017, 2006.

[155] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[156] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, October 2020.

[157] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[158] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

[159] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[160] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[161] George Stein, Jesse C. Cresswell, Rasa Hosseinzadeh, Yi Sui, Brendan Leigh Ross, Valentin Villecroze, Zhaoyan Liu, Anthony L. Caterini, J. Eric T. Taylor, and Gabriel Loaiza-Ganem. Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models. *CoRR*, abs/2306.04675, 2023.

[162] Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.

[163] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2015.

[164] **Jiajie Fan**, Amal Trigui, Thomas Bäck, and Hao Wang. Enhancing plausibility evaluation for generated designs with denoising autoencoder. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 88–105, Cham, 2025. Springer Nature Switzerland.

[165] **Jiajie Fan**, Laure Vuaille, Thomas Bäck, and Hao Wang. Adversarial latent autoencoder with self-attention for structural image synthesis. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 119–124, 2024.

[166] **Jiajie Fan**, Babak Gholami, Thomas Bäck, and Hao Wang. NeuroNURBS: Learning efficient surface representations for 3D solids. *CoRR*, abs/2411.10848, 2024.

[167] **Jiajie Fan**, Laure Vuaille, Thomas Bäck, and Hao Wang. On the noise scheduling for generating plausible designs with diffusion models. *CoRR*, abs/2411.10848, 2023.

[168] **Jiajie Fan**, Amal Trigui, Andrea Bonfanti, Felix Dietrich, Thomas Bäck, and Hao Wang. A mesh is worth 512 numbers: Spectral-domain diffusion modeling for high-dimension shape generation. *arXiv preprint arXiv/2503.06485*, 2025.

[169] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

[170] University of Illinois at Urbana-Champaign. UIUC Airfoil Database. http://m-selig.ae.illinois.edu/ads/coord_database.html, 2022. Accessed on 18.10.2022.

[171] U.S. Department of Transportation National Highway Traffic Safety Administration. Laboratory Test Procedure For FMVSS No.201U. Technical Report TP201U-01, U.S. Department of Transportation National Highway Traffic Safety Administration, 1998.

[172] Muhammad Usama, Mohammad Sadil Khan, Didier Stricker, and Muhammad Zeshan Afzal. Nurbgen: High-fidelity text-to-cad generation through llm-driven nurbs modeling, 2025.

[173] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[174] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[175] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[176] Pascal Vincent, H. Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, 2008.

[177] Kevin J Weiler. Topological structures for geometric modeling (boundary representation, manifold, radial edge structure). In *Rensselaer Polytechnic Institute*, 1986.

[178] Francis Williams. Point cloud utils, 2022. https://www.github.com/fwilliams/point-cloud-utils.

[179] Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Joinable: Learning bottom-up assembly of parametric cad joints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[180] Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4):1–24, 2021.

[181] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016.

[182] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6772–6782, October 2021.

[183] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*, abs/1708.07747, 2017.

[184] Bojun Xiong, Si-Tong Wei, Xin-Yang Zheng, Yan-Pei Cao, Zhouhui Lian, and Peng-Shuai Wang. OctFusion: Octree-based diffusion models for 3d shape generation. *arXiv*, 2024.

[185] Jingwei Xu, Zibo Zhao, Chenyu Wang, Wen Liu, Yi Ma, and Shenghua Gao. Cad-mllm: Unifying multimodality-conditioned cad generation with mllm, 2024.

[186] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaodong He, and Jianfeng Gao. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1316–1324, 2018.

[187] Xiang Xu, Joseph G Lambourne, Pradeep Kumar Jayaraman, Zhengqing Wang, Karl DD Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *arXiv preprint arXiv:2401.15563*, 2024.

[188] Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. In *International Conference on Machine Learning*, pages 24698–24724. PMLR, 2022.

[189] Shaoliang Yang and Jun Wang. Triple-parametric autoencoder for 2d reparameterization via bézier, b-spline, and nurbs representations. *Computer-Aided Design*, 189:103936, 2025.

[190] Isabela M. Yepes and Manasvi Goyal. Image classification using singular value decomposition and optimization, 2024.

[191] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.

[192] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[193] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M. Hospedales, and Chen Change Loy. Sketch me that shoe. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 799–807, 2016.

[194] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[195] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023.

[196] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023.

[197] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[198] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 586–595. Computer Vision Foundation / IEEE Computer Society, 2018.

[199] Zheyan Zhang, Yongxing Wang, Peter K Jimack, and He Wang. Meshingnet: A new mesh generation method based on deep learning. In *International conference on computational science*, pages 186–198. Springer, 2020.

[200] Junsheng Zhou, Weiqi Zhang, Baorui Ma, Kanle Shi, Yu-Shen Liu, and Zhizhong Han. Udiff: Generating conditional unsigned distance fields with optimal wavelet diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[201] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5826–5835, 2021.

[202] Wang Zhou. Image quality assessment: from error measurement to structural similarity. *IEEE transactions on image processing*, 13:600–613, 2004.

List of Abbreviations

ALAE	Adversarial Latent Autoencoder
B-Rep	Boundary Representation
CAD	Computer-Aided Design
CAE	Computational-Aided Engineering
CD	Chamfer Distance
DAE	Denoising Autoencoder
DDIM	Denoising Diffusion Implicit Mode
DDPM	Denoising Diffusion Probabilistic Model
DGM	Deep Generative Model
DPS	Design Plausibility Score
FDD	Fréchet Denoised Distance
FID	Fréchet Inception Distance
FMH	Free Motion Headform
FPS	Frames per Second
GAN	Generative Adversarial Network
GED	Generative Engineering Design
GenAI	Generative Artificial Intelligence
HIC	Head Injury Criterion
KDD	Kernel Denoised Distance
LDM	Latent Diffusion Model
LLM	Large Language Model
LMM	Large Multimodality Model
MMD	Maximum Mean Discrepancy
MSE	Mean Squared Error
NeRF	Neural Radiance Fields

NURBS	Non-Uniform Rational B-Splines
NWD	Neural Wavelet-domain Diffusion
ODE	Ordinary Differential Equation
PDR	Plausible Design Rat
PoDM	Plausible-oriented Diffusion Mode
ReLU	Rectified Linear Unit
SA-ALAE	Self-Attention Adversarial Latent Autoencoder
SDE	Stochastic Differential Equation
SDF	Signed Distance Field
SVD	Singular Value Decomposition
SVM	Support Vector Machine
VAE	Variational Autoencoder

Summary

In this thesis, we have systematically studied deep generative models to better generate engineering designs in terms of plausibility, efficiency, reliability and dimensionality for industrial purposes. We discuss the background of this paper, the development and evolution of DGMs, and the latest advances in the application of DGMs in engineering design in Chapter 1. In Chapter 2, our research on noise scheduling of diffusion models highlights the importance of the focusing range of noise levels during both training and sampling in improving the plausibility of generated designs. We propose a data analysis method that is able to help with determining the focusing range for a given dataset. In developing diffusion models for better plausibility, most widely-implemented metrics (e.g., FID, KID, LPIPS) do not align well with human judgments in evaluating the plausibility, which makes it challenging to rank and select target DGMs. To address this issue, in Chapter 3, we subsequently build a new evaluation method driven by denoised autoencoder for better assessing the generated structural information. After successfully studying the performance of DGMs on 2D structural designs, our work moves on to the 3D design domain. In Chapter 4, we first delve into the field of mesh generation and we design a SVD-based encoding pipeline, with which we are able to reduce the source high-dimension shape into low-dimension spectral features in a learning-free manner. These spectral features are proven to be trainable with DGMs, hereby enabling DGMs on high-dimension data without training an autoencoder. In Chapter 5, we move to the field of CAD generation tasks and tackle the challenge of direct generation of B-Rep solids with surfaces represented in NURBS. Finally in Chapter 6, we bring insights of the application of our developed methods in real-world industrial design cases.

This thesis presents our pioneering attempts at bridging the gap between modern DGM research and industrial applications. While our work addresses the performance of DGMs in terms of efficiency and reliability in generating engineering designs, our

proposed methods have also brought great contribution to the general DGM field. More precisely, our methods are evaluated on open-source datasets widely implemented in the general DGM field and compared to current state-of-the-art with sufficient assessments. However, a gap persists for industrial applications, with design data in industrial cases presenting exponentially more complicated structural information than general datasets and a much better representation and generation framework are required. In the following, we identify current limitations, suggest potential ideas to address the limitations, and consider future research directions of DGMs, particularly in industrial generative design field.

Samenvatting

In dit proefschrift hebben we diepgaande generatieve modellen systematisch bestudeerd om technische ontwerpen beter te kunnen genereren in termen van plausibiliteit, efficiëntie, betrouwbaarheid en dimensionaliteit voor industriële doeleinden. We bespreken de achtergrond van dit artikel, de ontwikkeling en evolutie van DGM's en de laatste ontwikkelingen in de toepassing van DGM's in technisch ontwerp in Chapter 1. In Chapter 2 benadrukt ons onderzoek naar ruisplanning van diffusiemodellen het belang van het focusbereik van ruisniveaus tijdens zowel training als bemonstering voor het verbeteren van de plausibiliteit van gegenereerde ontwerpen. We stellen een methode voor gegevensanalyse voor die kan helpen bij het bepalen van het focusbereik voor een bepaalde dataset. Bij de ontwikkeling van diffusiemodellen voor een betere plausibiliteit sluiten de meest gebruikte meetcriteria (bijv. FID, KID, LPIPS, enz.) niet goed aan bij het menselijk oordeel bij de beoordeling van de plausibiliteit, wat het moeilijk maakt om doel-DGM's te rangschikken en te selecteren. Om dit probleem aan te pakken, hebben we in Chapter 3 vervolgens een nieuwe evaluatiemethode ontwikkeld op basis van een denoised autoencoder voor een betere beoordeling van de gegenereerde structurele informatie. Na succesvol onderzoek naar de prestaties van DGM's op 2D-structuurontwerpen, gaan we verder met het 3D-ontwerpdomein. In Chapter 4 verdiepen we ons eerst in het genereren van mesh en ontwerpen we een op SVD gebaseerde coderingspijplijn, waarmee we de bronvorm met hoge dimensies op een leerloze manier kunnen reduceren tot spectrale kenmerken met lage dimensies. Deze spectrale kenmerken blijken trainbaar te zijn met DGM's, waardoor DGM's kunnen worden toegepast op hoogdimensionale gegevens zonder een auto-encoder te trainen. In Chapter 5 gaan we over naar het gebied van CAD-generatietaken en pakken we de uitdaging aan van het direct genereren van B-Rep-solids met oppervlakken die worden weergegeven in NURBS. Ten slotte geven we in Chapter 6 inzicht in de toepassing van onze ontwikkelde methoden in praktijkvoorbeelden van industrieel ontwerp.

Dit proefschrift presenteert onze baanbrekende pogingen om de kloof tussen modern DGM-onderzoek en industriële toepassingen te overbruggen. Hoewel ons werk zich richt op de prestaties van DGM's in termen van efficiëntie en betrouwbaarheid bij het genereren van technische ontwerpen, hebben onze voorgestelde methoden ook een grote bijdrage geleverd aan het algemene DGM-veld. Meer precies worden onze methoden geëvalueerd op open-source datasets die op grote schaal worden geïmplementeerd in het algemene DGM-veld en vergeleken met de huidige state-of-the-art met voldoende beoordelingen. Er blijft echter een kloof bestaan voor industriële toepassingen, waarbij ontwerpgegevens in industriële gevallen exponentieel complexere structurele informatie bevatten dan algemene datasets en een veel beter representatie- en generatieframework vereist is. Hieronder identificeren we de huidige beperkingen, suggereren we mogelijke ideeën om deze beperkingen aan te pakken en bekijken we toekomstige onderzoeksrichtingen voor DGM's, met name op het gebied van generatief ontwerp in de industrie.

Acknowledgment

At the outset, this work began as a blank slate. Long before GenAI became part of mainstream discourse, we embarked on experimenting with machine learning methods to generate recognizable engineering designs. What began as a bold idea soon revealed itself to be a transformative endeavor—one of profound significance, yet accompanied by uncertainty and formidable challenges. This journey, demanding both perseverance and belief, became possible only through the unwavering support, guidance, and encouragement of many exceptional individuals.

I first express my deepest gratitude to my supervisors: Prof. Thomas Bäck, Dr. Babak Gholami, Dr. Hao Wang, and Laure Vuaille. Your vision laid the foundation for this research, and through moments of doubt, complexity, and discovery, your guidance continuously illuminated the path forward. I am profoundly grateful for your patience, trust, and belief in this work. I also extend heartfelt thanks to my co-author Amal Trigui, with whom I collaborated tirelessly over countless days. Our shared passion, curiosity, and determination to push the boundaries of this research transformed challenges into milestones and made this journey truly rewarding. Furthermore, I sincerely thank the BMW Group for its generous support, which provided the essential infrastructure, resources, and funding that enabled both my research and international conference participation.

This journey has been enriched by joy, learning, and unforgettable memories: traveling to Singapore with Dr. Roy de Winter and Dr. Anna Kononova to attend the CAI 2024 conference and present my first publication; joining my co-author Amal Trigui in Milan for the ECCV 2024 conference to demonstrate our newly accepted work; leading 30 BMW doctoral students on an educational tour of Shenyang, Shanghai, and Nanjing; and being invited to the BMW 2024 Summer School on Fraueninsel, Chiemsee. These experiences not only shaped my academic growth but also left lasting personal impressions. I am deeply thankful to my colleagues Gabriela Barrón Loeza, Andrea

Bonfanti, Fabia Schito, Duarte Madeira, Mohammad Rashed and Srishti Dasgupta for the precious moments we shared in our “lab” (BMW office), for moments of laughter and respite amid intense work, and for the mutual support that carried us through this demanding adventure.

Last but certainly not least, I owe my deepest gratitude to my parents for their unconditional love and unwavering support, and to my friends in Germany. Their presence gave me strength during the pandemic and comfort during moments of distance, uncertainty, and self-doubt while living in a foreign country. This PhD journey has been one of the most meaningful and fulfilling periods of my life thus far. They kept me grounded, celebrated my joys, and listened patiently when the journey felt overwhelming. My dear friends and family, you fill my life with joy and remind me to make time for my passions, friendships, and well-deserved moments of rest. To everyone mentioned above, thank you for being an inseparable part of this journey.

Curriculum Vitae

Jiajie Fan was born in 1996 in Wenzhou, China. After completing his undergraduate studies in Automotive Engineering at Wuhan University of Technology, he pursued a Master's degree in Mechanical Engineering at Technische Universität Darmstadt, which he obtained in 2022. Following his master's studies, he began his doctoral studies in Computer Science at Leiden University's Leiden Institute of Advanced Computer Science (LIACS), in cooperation with BMW. There, he conducts research on generative design and machine learning methods under the supervision of Prof. Thomas Bäck, Dr. Babak Gholami, Dr. Hao Wang, and Laure Vuaille. During his PhD studies he took several courses, including the Scientific Conduct.

Recently, Jiajie joined the newly established startup Kyrall as a founding engineer, where he works on bringing AI for CAD from zero to one, enabling generative artificial intelligence for engineering CAD generation. In this role, he contributes to the development of core technologies that bridge cutting-edge research in generative models with practical, industry-ready engineering design workflows. His goal is to advance the integration of generative AI into real-world engineering processes, making design more efficient, creative, and accessible.