Generative Models for Feature-Based Product Development as a Basis for Hybrid Decision-Making

Valentin Schemmann[1], Thorsten Schmidt[1,*], Niels Demke[1], Frank Mantwill[1]

[1] Institute of Machine Elements and Computer Aided Product Design (MRP), Helmut Schmidt University Hamburg

* *Corresponding author:*
Thorsten Schmidt
MRP, Helmut-Schmidt-Universität Hamburg
Holstenhofweg 85
22043 Hamburg
☎ *040/6541-3794*
✉ *thorsten.schmidt@hsu-hh.de*

Abstract

This paper investigates the general possibility for applying generative models in the early phase of product development. For this purpose, the fundamentals of feature-based product development are introduced and related to the development methodology VDI 2221 alongside a brief overview of deep generative models. Based on this, a conceptual framework is developed that combines the methods and proposes a collaborative approach. In conclusion, a prototypical implementation is performed by training a StyleGAN2 based on vehicle profiles followed by executing a GANSpace principal component analysis. Finally, the various results are presented and the possibilities of manipulating the generated images based on identified features are discussed and transferred back into the product development process.

Keywords

Feature-Based Product Development, Deep Generative Models, Principal Component Analysis, StyleGAN2, GANSpace

 | DOI: 10.35199/dfx2023.18

1. Introduction

The design field is increasingly emerging as the next application domain of *Artificial Intelligence* (AI) [1]. The recent development of a whole group of generative models like Text-to-Text (e.g., ChatGPT), Text-to-Image (e.g., Midjourney) or even Text-to-Video (e.g., Synthesia) has opened the possibility to support product managers and designers in creative tasks by providing powerful tools and support. Especially for an early-stage design, they offer the possibility to generate new solutions within a defined solution space, for example by recombination of existing features, and thus to investigate the design space systematically and successfully [1, 2]. As machine creativity is still limited, such an approach represents a form of support for limited creative tasks in the context of assistance systems [3].

While a developer's proposed solution space is rooted in their individual experiences, the solution space of a generative model is shaped by the diversity determined within its training dataset. This leads to challenges in the formalization of engineering problems and development of assistance systems. Features and their combination represent a possible field for considering the derivation of explicit design knowledge in the context of hybrid decision processes. Generative models can be used to generate proposed solutions that are evaluated by the developer so that features can be recognized. In computer science, product creation is an application area. Features refer to data and their measurable properties in different domains [4]. In product development, there is a strong focus on explainability, which leads to different approaches compared to generative models.

This paper examines the technical capabilities of generative models and feature recognition within the context of assistance systems in collaboration with developers, proposing a novel framework to integrate these elements. Through a prototypical implementation, key aspects of the framework are practically applied, and the outcomes are thoroughly discussed. The proposed framework aims to streamline and expedite the development process, offering potential benefits in terms of efficiency and simplification.

2. Research Problem & Research Question

The addressed research problem investigates the formalization of implicit design knowledge through the recognition of relevant design features by hybrid assistance systems in the early stages of product development. These decision systems describe possible applications for the internalization of knowledge, for example by evaluating the conditionability of design parameters of a *Deep Generative Model* (DGM). Through the hybrid use of DGMs, development work is focused on the evaluation of proposed solutions. This raises questions about the interaction between the developer and the assistance system as well as the AI methods used to derive features.

Three research objectives are addressed to support formalizing the design process through generative models: First, description of the product development process considering the formalization of design knowledge about features. Second, description and development of a framework to methodical support the designer in his creative tasks. Third, application of a *Generative Adversarial Network* (GAN) to generate artificial images and subsequent performing of a *Principal Component Analysis* (PCA) to identify design parameters in a prototypical application as a tool. These overall research objectives lead to the following research question: How can generative models be used to derive features to support implicit design knowledge for purposeful influencing the solution in the context of hybrid decision processes?

By addressing the research question, the objective is to contribute to the acceleration of prototyping through the systematic extraction of requirements and structural features at an early stage and, based on this, to extend the existing understanding of feature technology and knowledge-based engineering [5, 6].

3. Methodology

In order to answer the research question, this paper employs a four-step approach to establish a fundamental understanding of utilizing generative models in the context of feature-based product development, which is described in the following.

First, a literature review of related work and neighboring fields is presented. In this paper, a specific effort is made to focus on the early phase of product development, which is crucial for the later success of a product [7]. The literature review describes in particular the term "feature" in the traditional sense of feature-based product development in the VDI guideline 2218 [6] as well as in the context of the application within generative models, e.g., in the form of so-called “feature extraction”.

Second, using the identified Knowledge-, Processing- and Information-Layers, a conceptual framework is developed and presented, which describes the use of generative models in the early phase of product development. This framework emphasizes the VDI guideline 2221 from the clarification of requirements to the elaboration of a first initial system design [8]. Influences on the process model by the presented technologies and their focus on the evaluation of proposed solutions are shown.

Third, the actual DGM is trained and analyzed for its main components. The observation of the principal components allows identification of the characterizing features in the generated images. The images are then used to enrich the set of transformed training data or used in a design exploration loop to evaluate partial design ideas in the detail design process.

Based on the core of this framework, a prototypical implementation is exemplarily performed to validate the described capabilities of generative models. In this paper, the prototypical implementation is limited to the use of images from an available training dataset of side profiles of passenger cars [9]. The prototype includes a *StyleGAN2* implementation and a PCA of the generated latent space. The solution space is required to consist of difficult-to-distinguish images of the same size and style as the original dataset.

Fourth, the results are interpreted and the possibilities to transfer the gained knowledge in the sense of feature-based product development in terms of hybrid decision support for the derivation of design parameters or the incorporation of planning processes are discussed.

4. Related Work

4.1. Feature-Based Product Development

Formerly, feature technology was used to map manufacturing instructions for specific geometric sub-areas in work planning [6]. Based on this, features were further developed as information and integration objects over the entire life cycle of a product, contributing to a simplification and acceleration of product development [10]. The term “feature” is not standardized. In general, features can be understood as information elements representing areas (not exclusively geometry) of special (technical) interest of single or multiple products (physically realizable object created by a natural process or by manufacturing) [11]. With this definition, non-geometric product features are included as well, as they are important for following processes and data analysis (cf. [10]).

Features are also used for further enhancement of the product model by providing information and allowing relevant aspects of the Design for X to be included [6]. In this context, features represent a specific view of the product description, which are associated with characteristic classes and life cycle phases [6]. Accordingly, different types of features are used, such as geometry features, design features, information features, manufacturing features, assembly features, functional features, calculation features, and measurement features [10].

In order to create capacity for creative and innovative activities, feature technology forms the basis for achieving automation potentials in routine tasks by representing knowledge as

well as processing it [6]. Feature combinations provide a way to combine feature elements and create new knowledge [10]. In this context, however, there are risks that models are overloaded with too much information and no longer achieve the desired benefit in terms of their usability [10]. Thus, in addition to the design activity, further skills are needed for the development of information technologies in product development. Another issue results from the increasing availability and use of historical product data for deriving design knowledge in product development. With the different understanding of the feature term in the informatic science and the use of generative models, the following section will deal with the necessary basics.

4.2. Deep Generative Models

Deep Generative Models (DGMs) are a subclass of machine learning algorithms that can learn underlying distributions of large-scale datasets and generate new samples that resemble the training data. Generative modeling can be broadly classified into four main categories *Autoregressive Models*, *Flow-based Models*, *Latent Variable Models* and *Energy-based Models* [12] as shown in Figure 1. *Generative Adversarial Network* (GAN) and *Variational Autoencoders* (VAE) belong to the Latent Variable Models and can learn and represent complex data distributions in multi-dimensional spaces. Originally developed in the field of computer vision, these models increasingly attract researchers in the engineering design community as these models are able to synthesize new samples. As part of new product creation, four common approaches of DGM can be identified with applications within the design stage. These include the leverage of *Deep Neural Networks* (DNNs), GANs, VAEs, and *Reinforcement Learning* (RL) which are commonly used for design synthesis [2, 13].

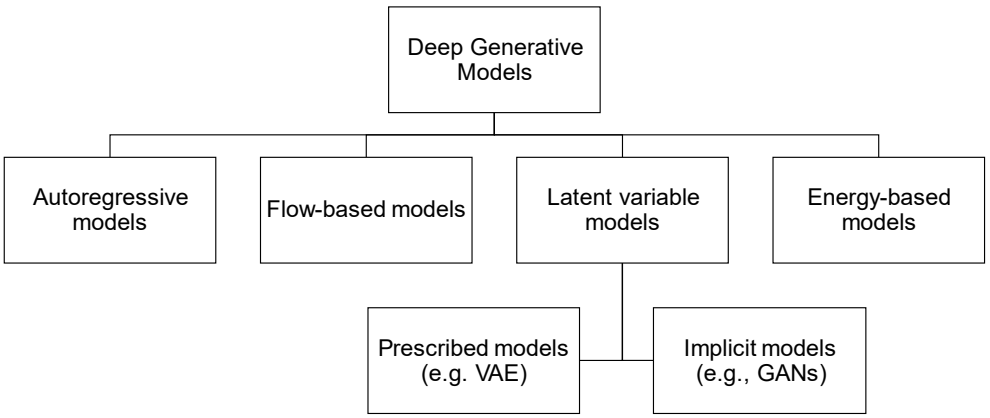

Figure 1: Taxonomy of DGMs based on Tomczak [12]

First introduced in 2014, the idea of a GAN [14] has attracted more and more attention as these models show strong results in image synthesis [15]. A GAN consists of two antagonist models, a generator G and a discriminator D, who work as binary classifiers [16]. D learns the data distribution from the training data and is able to classify new data as real (part of the training data) or fake (not part of the training data) after sufficient training. G generates random images using a random noise input signal, which are then classified by D as real or fake. If samples are classified as real, the model is capable of generating new samples that possess statistical similarity to the training data but were not part of the original data. The symbol E represents expected values for the training data $E_{x\sim p_{data}(x)}$ and the generates samples $E_{z\sim p_z(z)}$.

In other words, GANs can generate patterns that closely mimic the distribution of real data, as they are able to learn and follow the underlying rules and patterns present in the real data through the process of self-exploration [17]. The min-max optimization function for an exemplary GAN is displayed in (1) [14]:

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}[log\, D(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))] \quad (1)$$

Equation (1) comprises two logarithmic loss functions, one for the generator and one for the discriminator. The primary objective of the discriminator is to maximize its loss by applying penalties whenever it misclassifies real instances from the training data as fake or incorrectly identifies fake instances generated by the generator as real. On the other hand, the generator strives to minimize this function. The generator loss is determined based on the classification of the discriminator. It receives a reward when successfully fooling the discriminator and is penalized otherwise.

4.3. Usage of the Feature Term in GANs and Application in Product Creation

In contrast to feature-based product development, the different meaning of the term “feature” in the domain of DGMs needs to be clarified. According to Martin [4] “a feature is a semantic group (modeling atom), characterized by a set of parameters, used to describe an object which cannot be broken down, used in reasoning relative to one or more activities linked to the design and use of products and production systems” (p. 219) [4]. The term can be interpreted as a descriptive characteristic of an object to be described. In image recognition as a subfield of computer vision, feature recognition is widely used in the classification of images and object recognition on images. These features are differentiated into color features, texture and shape features [18].

However, not only image classifiers but also generative models are powerful for feature extraction and anomaly detection. Due to the fact that many generative models have their origin in computer vision, there are recent efforts to apply these approaches in engineering practice and to use the models in product creation. In engineering design practice, several potential applications of DGMs, specifically GANs, have emerged, which are exemplarily described below. The applications encompass topology optimization, generation of microstructures and metamaterials, as well as shape generation in 2D (e.g., images) and 3D (e.g., voxel or point clouds) [2].

CreativeGAN is a method introduced by Nobari et al. [13] for the automated design of novel bike frameworks using a GAN, novelty detection, and segmentation network. The objective of the model is to synthesize images of new bike frame designs with novel features that were previously detected in the training data.

RangeGAN, on the other hand, is a conditional design synthesis model for 3D airplane designs. It enables the generation of designs within a specific range, providing control over the characteristics of the generated designs. The authors demonstrate the feasibility of integrating multiple constraints into the model and conditioning the outcomes on several constraints, such as volume and length-to-wingspan ratio, allowing for potential overlaps [13].

Another approach is proposed by Gu et al. [5], where they combine a *Convolutional Neural Network* (CNN) with a GAN. Initially, the GAN generates images of springs, and their geometric properties are then predicted by the CNN. Subsequently, 3D-CAD models can be directly created from the generated data, allowing the determination of mechanical properties. This approach helps to accelerate the typical trial and error in the design process.

5. Conceptual Framework

Due to the rapid technological developments in AI, a need and opportunity have been identified to integrate these new methods into the process outlined in VDI 2221 [8] to support designers efficiently and show promising direction for the future. To offer a starting point to integrate these methods, the proposed framework in Figure 2 comprises three distinct meta-levels:

First, the *Knowledge Layer* (KL), which provides the underlying data basis as well as implicit knowledge about features and association rules. There exist various configurations of DGMs with different requirements for training data. For this purpose, the framework includes a KL responsible for data storage and retrieval. Data in product development can take various formats, such as tables, images, or higher dimensional structures such as meshes. The data is gathered from different sources, such as product specifications, user requirements, market research, and past designs. DGMs have special requirements for input data and formats, depending on their use case. Therefore, the primary task of the KL is not only data storage but also data selection, preprocessing (e.g., checking for completeness, errors etc.) and transformation.

Second, the *Processing Layer* (PL) encompasses the architectures and mechanisms of the DGM. It is important to emphasize that there are no one-size-fits-all solutions, and a differentiation according to the specific use case must be made.

Finally, the *Information Layer* (IL) includes the interface to human input. The IL serves as an interface between the designer and the assistance system. For the interaction with the user, different possibilities such as sliders or value tables are available to influence the solution in a targeted manner. The traditional development process starts with the initial requirements for the product or a detailed problem description. Derived from these requirements, the developer initiates the generation of solutions in the "Create" phase. The next decision point is the evaluation, which still requires the developer's expertise to select a solution that will then be further elaborated and optimized in "Further Detail Steps".

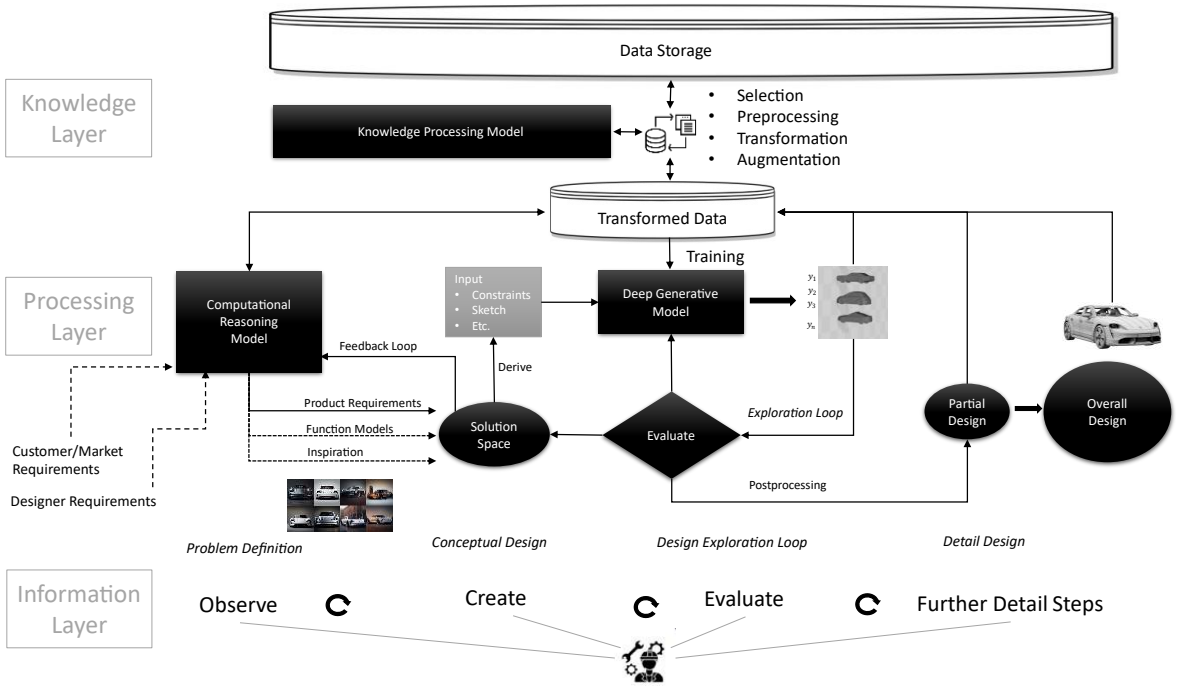

Figure 2: Knowledge-, Processing- and Information-Layer of the conceptual framework

The task of the central DGM is to generate solution proposals based on the requirements, which will subsequently be evaluated by the developer.

6. Prototypical Implementation

To realize the proposed approach and to demonstrate the general applicability of generative models, a prototypical implementation is performed in two steps. Leveraging the established framework, this implementation considers the central aspects encompassing data preprocessing, training of the DGM and evaluation of the generated results. The development of the computational reasoning model as well as the further derivation of partial design models and an overall design will not be part of the scope of this particular implementation.

Based on the idea of Macey [19] to classify vehicles according to their features using a classification map, this approach deals with the derivation of features from images. The used dataset consists of 1480 images depicting the side profiles of passenger cars from 37 distinct car brands in a resolution of 1024 x 1024 pixels [9]. This dataset is suitable because the pictured objects share a high degree of similarities and represent a true industrial design task as it is found in the real world when it comes to the design of early stage sideviews of cars. To provide a reasonable increase in the amount of used training data, the prototypical implementation uses adaptive discriminator augmentation to artificially expand the size of the dataset.

6.1. Vehicle Shape Generation Model with StyleGAN2

In order to build and train a model, the codebase of StyleGAN2 has been chosen. StyleGAN2 is an extension of StyleGAN, which has been published by Nvidia Corporation in 2020 and has achieved state-of-the-art results in generating unconditional images using GANs [20]. The basic GAN configuration assumes the existence of a probability distribution $p(z)$ from which latent vectors $z \in Z$ are sampled. Through a neural network, the generator function G is learned, which maps the latent vector to output, in this case, an image $I = G(z)$. StyleGAN introduces a mapping network including another set of latent vectors w and an additional latent space W, $w \in W$ which is another mapping from the Z space that is more disentangled and offers the possibility to separate high-level attributes of images [21].

The training takes place on a virtual runtime hosted by Google Colab [22] with a Nvidia Tesla T4 GPU and comparatively on a local runtime with a single Nvidia V100 GPU using the same Python code. The training of the StyleGAN2 is complex as well as computationally and time intensive and is therefore investigated on the basis of the two different setups. Over the entire runtime, the cloud-based environment achieves ≈507 sec/kimg while the local environment performs ≈153 sec/kimg, where the unit kimg stands for 1,000 generated images. Depending on the respective size of the used dataset, an optimal model must have seen about 25,000 kimg to reach convergence for an extensive dataset of e.g., 70,000 images [23]. Reasonable results are already achieved after approx. 5,000 kimg [24]. Therefore, the presented research is terminated after 5,000 kimg, resulting in a significant time advantage of ≈8,9 days of training on the local runtime compared to ≈29,3 days of training on the hosted runtime. For larger models with even more training data, there is a correspondingly significant need for more powerful hardware.

After the training, the trained model is able to generate images from a random seed (noise vector from the Z space distribution) which resemble the training dataset in style and motive and have the same resolution of 1024 x 1024 pixels.

6.2. Principal Component Analysis with GANSpace

In preparation for examining the generated latent space, the first step is to convert the file generated from TensorFlow (.pkl) into a PyTorch format (.pt). For this preprocessing step, the ”SG2-ADA-PT to Rosinality” implementation is used for converting [25].

Subsequently, a PCA is performed using the *GANSpace* implementation to find interpretable GAN controls [26]. The PCA processes the high-dimensional latent space of the remapped W space and reduces its dimensionality allowing visualization. Therefore, the exploration of the intermediate result contributes significantly to the explainability and consequently to the acceptability of the selected approach. The number of resulting principal components is at most equal to the degree of dimensionality of the latent space [20, 27].

A clustering of the principal components reveals that the associated images share similar features. Varying along such a principal component changes the manifestation of the corresponding feature on the image and forms the so-called feature vector w [20]. This can concern single (distinctive) or multiple features at once [20]. Figure 3 shows two of the 24 examined principal components. In this case, the generated principal components in descending order are causal for the intensity of the manifestation of the represented features. The first principal component apparently describes the driving direction of the vehicle as well as the background. The second represented principal component describes the vehicle's bodyshape and the color of the vehicle. Other identified features are for example the rims or the wheelbase of the vehicle. The complete interpretation of the identified features is still up to the individual and is complex and therefore prone to error, especially when interpreting multi-feature principal components. In particular, multi-feature principal components represent an area of further research. Ideally, each principal component corresponds to exactly one identified feature.

Figure 3: Varying two of the 24 examined principal components and the corresponding changes to the images

The possibility to examine the latent space with the help of a PCA is a special aspect of the combined use of StyleGAN2 and GANSpace due to the harmonized architecture and therefore constitutes a particularly symbiotic application.

7. Results & Discussion

The results of this paper are divided into a conceptual part, an analytical part and an applicational part. In the conceptual part, a contribution was made by the clarification of the feature term in the feature-based product development as well as with the general taxonomy of DGMs. The framework describes a conceptual method to integrate the use of generative models into a process model according to VDI 2221 [8]. In the analytical part, a PCA was presented to explore the latent space of a previously trained StyleGAN2 implementation. As a

result of this exploration, the principal components were identified as characteristic properties of the images used for training, which can be specifically modified for the generation of new images by the generator. The result of this modification is a manipulable image that can be morphed from identified design parameters and used as the basis of partial design in the early stages of product development.

Overall, the results demonstrate the general applicability of the proposed approach for the formulated purposes and for data-driven assistance systems of a developer. When analyzing the achieved quality of the images over the runtime, a fast increase with subsequently decreasing marginal returns is observed. A closer look at the generated images in Figure 4 reveals some errors so-called anomalies.

Figure 4: Four images generated by a random seed and the trained StyleGAN2 implementation

Improvements in the quality of the generated images are not a subject of this study, but presumably could be realized by a larger training dataset, a longer runtime or more powerful hardware.

8. Conclusion & Future Research

The presented approaches offer the potential to open the black box characteristic of AI methods and significantly increase the explainability of the presented solution. The presented concept exemplifies a possible use of data-driven methods in product development in order to create a foundation for the exploration of novel applications.

According to current research, the use of PCA is exclusively limited to BigGAN and StyleGAN based architectures, which is a restriction in the selection of further methods of generative models. Another restriction is the limitation of the generated solution within the solution space, which is defined by the training dataset. The generation of solutions outside the solution space remains the subject of future research. There is a further need for research in the integration of functions and in the consideration of additional constraints such as package space limitations as well as the detection and extrapolation of trends, for example, in the generation-spanning consideration of vehicle models. To address functions and other constraints, integrations between symbolic and sub-symbolic AI approaches should also be considered. Furthermore, time dependencies have to be included in order to incorporate planning processes. A focus on processing multi-feature principal components should be pursued in further work.

A more in-depth evaluation with an industrial partner, which goes beyond the prototypical implementation, is still pending. The aim of this evaluation with an industry partner could be the development and use of a user interface with sliders for the identified feature vectors. Using the sliders would give the user the interactive opportunity to try different linear combinations of feature values and evaluate the created overlay of morphed design parameters immediately.

Finally, the dependency of the quality of the generated images on technical factors such as the quantity of the training data, the used hardware and respectively the runtime requires further investigation.

References

[1] de Kleer, Johan; Feldman, Alexander; Matei, Ion: *Correcting Design Errors in Components and Connections Circuit Design.* Palo Alto Research Center, 2019.

[2] Regenwetter, Lyle; Nobari, Amin Heyrani; Ahmed Faez: *Deep Generative Models in Engineering Design: A Review*. In: Journal of Mechanical Design 144.7, DOI: 10.1115/1.4053859, 2022.

[3] Franceschelli, M. M. Giorgio; Musolesi, Mirco: *Creativity and Machine Learning: A Survey.* https://arxiv.org/abs/2104.02726, 2021.

[4] Martin, Patrick: *Some aspects of integrated production and manufacturing*. In: Advances in Integrated Design and Manufacturing in Mechanical Engineering (Bramley, A., Brissaud, D., Coutellier, D., & McMahon, C., Eds.), pp. 215–226. Amsterdam: Springer, 2005.

[5] Gu, Zewen et al.: *A Novel Self-Updating Design Method for Complex 3D Structures Using Combined Convolutional Neuron and Deep Convolutional Generative Adversarial Networks*. In: Advanced Intelligent Systems 4.6, p. 2100186, DOI: 10.1002/aisy.202100186, 2022.

[6] VDI: VDI-Richtlinie 2218 – *Informationsverarbeitung in der Produktentwicklung: Feature-Technologie*, 2003.

[7] Ehrlenspiel, Klaus et al.: *Kostengünstig Entwickeln und Konstruieren*. Berlin, Heidelberg: Springer Vieweg, DOI: 10.1007/978-3-642-41959-1, 2020.

[8] VDI: VDI-Richtlinie 2221 – *Entwicklung technischer Produkte und Systeme - Modell der Produktentwicklung*, 2019.

[9] Lee, Gyunpyo; Kim, Taesu; Suk, Hyeon-Jeong: *GP22: A Car Styling Dataset for Automotive Designers*. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, https://arxiv.org/pdf/2207.01760, 2022.

[10] Bossmann, Marc: *Feature-basierte Produkt- und Prozessmodelle in der integrierten Produktentstehung*. Dissertation Universität des Saarlandes, Band 38, 2007.

[11] Weber, Christian: *What is a Feature and What is its Use? – Results of FEMEX Working Group.* In: 29th International Symposium on Automotive Technology and Automation 1996 (ISATA 96), Florenz/Italien 03.–06.06.1996. Tagungsband S. 109/116, 1996.

[12] Tomczak, Jakub M.: *Deep Generative Modeling*. Springer eBook Collection, Cham: Springer, DOI: 10.1007/978-3-030-93158-2, 2022.

[13] Nobari, Amin; Heyrani, Wei Chen; Ahmed, Faez: *Range-GAN: Range-Constrained Generative Adversarial Network for Conditioned Design Synthesis*. URL: https://arxiv.org/pdf/2103.06230, 2021.

[14] Goodfellow, Ian J. et al.: *Generative Adversarial Networks.* https://arxiv.org/pdf/1406.2661, 2014.

[15] Karras, Terro; Laine, Samuli; Aila, Timo: *A Style-Based Generator Architecture for Generative Adversarial Networks*. URL: https://arxiv.org/pdf/1812.04948, 2018.

[16] Russell, Stuart J.; Norvig, Peter: *Artificial intelligence: A modern approach*. Fourth Edition, Global Edition. Pearson Series in Artificial Intelligence. Harlow: Pearson, 2022.

[17] Ganguly, Kuntal: *Learning Generative Adversarial Networks: Next generation deep learning simplified*. 1st ed. Birmingham: Packt Publishing, 2017.

[18] Tian, Dongpin: *A Review on Image Feature Extraction and Representation Techniques.* In: International Journal of Multimedia and Ubiquitous Engineering 8(4), p. 385-295, 2013.

[19] Macey, Stuart; Wardle, Geoff: *H-Point: The fundamentals of car design & packaging*. 1. ed. Culver City: Design Studio Press, URL: https://permalink.obvsg.at/AC07759341, 2009.

[20] Meng, Shengyu: *Exploring in the Latent Space of Design: A Method of Plausible Building Facades Images Generation, Properties Control and Model Explanation Base on StyleGAN2.* In: Yuan, P.F., Chai, H., Yan, C., Leach, N. (eds) Proceedings of the 2021 DigitalFUTURES. CDRF 2021. Springer, Singapore, https://doi.org/10.1007/978-981-16-5983-6_6, 2021.

[21] Karras, Terro et al.: *Analyzing and Improving the Image Quality of StyleGAN*. https://arxiv.org/pdf/1912.04958, 2019.

[22] Bisong, Ekaba: *Google Colaboratory.* In: Building Machine Learning and Deep Learning Models on Google Cloud Platform - A Comprehensive Guide for Beginners. Apress, Berkeley, CA. pp. 59-64, https://doi.org/10.1007/978-1-4842-4470-8_7, 2019.

[23] NVlabs: *FFHQ – Dataset*. https://github.com/NVlabs/ffhq-dataset, 2023

[24] Karras, Terro et al.: *Training Generative Adversarial Networks with Limited Data.* URL: https://arxiv.org/pdf/2006.06676, 2020.

[25] Google Colab Notebook: *PKL to PT Converter*: https://colab.research.google.com/github/dvschultz/stylegan2-ada-pytorch/blob/main/SG2_ADA_PT_to_Rosinality.ipynb, 2023.

[26] Härkönen, Erik et al.: *GANSpace: Discovering Interpretable GAN Controls*. In: Advances in Neural Information Processing Systems 33, URL: https://arxiv.org/pdf/2004.02546, 2020.

[27] Yonekura, Kazuo; Wada, Kazunari; Suzuki, Katsuyuki: *Generating various airfoil shapes with required lift coefficient using conditional variational autoencoders.* URL: http://arxiv.org/pdf/2106.09901v1, 2021.